МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Київський національний університет будівництва і архітектури

Web-програмування

Методичні вказівки
до виконання лабораторних робіт
для здобувачів першого (бакалаврського) рівня
вищої освіти за спеціальностями
F3 «Комп'ютерні науки»,
F6 «Інформаційні системи та технології»

Київ 2026

УДК 004.42:004.738.5
В26

Укладачі: Ю. О. Науменко, канд. техн. наук, доцент
А. Г. Хаддад, керівник ФОП

Рецензент О. В. Горда, канд. техн. наук, доцент

Відповідальна за випуск Т. А. Гончаренко, д-р техн. наук, професор

Затверджено на засіданні кафедри інформаційних технологій, протокол № 6 від 16 лютого 2026 року.

В авторській редакції.

Web-програмування [електронний ресурс]: методичні вказівки
В26 до виконання лабораторних робіт/ уклад.: Ю.О. Науменко, А.Г. Хаддад. – Київ : КНУБА, 2026. – 35 с.

Містять зміст, порядок оформлення і вказівки до виконання лабораторних робіт.

Призначено для здобувачів першого (бакалаврського) рівня вищої освіти, спеціальностей F3 «Комп'ютерні науки», F6 «Інформаційні системи та технології»

ЗМІСТ

Загальні положення

Дисципліна «Web-програмування» є наскрізною для програм підготовки фахівців з комп'ютерних наук та програмної інженерії, оскільки вона формує у здобувачів практичну здатність проєктувати, розробляти та підтримувати сучасні динамічні вебсистеми на засадах безпеки та ефективності. Актуальність курсу зумовлена домінуючою роллю мови PHP у сучасному вебі (близько 70% ринку), складністю сучасних інформаційних систем та потребою у валідованих даних і прозорій архітектурі коду.

Методичні вказівки з дисципліни охоплюють повний цикл розробки: від написання першого скрипта та налаштування локальних або хмарних середовищ (Replit, VS Code) – до вибору архітектурного дизайну, обробки даних та впровадження шаблону Model-View-Controller (MVC). Мета вказівок — сформувати цілісні знання та закріпити практичні навички роботи зі скалярними типами, керуючими структурами, файловими системами та клієнт-серверною комунікацією.

Окремий акцент у навчальному процесі зроблено на переході від базових мовних конструкцій до глибокої роботи з даними. Здобувачі вивчають не лише просту ітерацію, а й способи ефективного управління великими обсягами інформації через складні масивні структури та взаємодію з файловою системою сервера. Це закладає фундамент для розуміння того, як алгоритми та структури даних поєднуються для розв'язання реальних задач розробки.

Вказівки містять рекомендації й типові приклади, що у практичному зрізі відображають рівень засвоєння курсу, зокрема здобувач повинен уміти:

- ✓ налаштовувати середовище розробки та розуміти різницю між CLI та Web Server проєктами;
- ✓ ефективно оперувати змінними, константами та операторами, дотримуючись сучасних стандартів іменування;
- ✓ працювати з рядками та використовувати вбудовані функції для аналізу та трансформації тексту;
- ✓ реалізовувати складну логіку прийняття рішень за допомогою умовних операторів (if, switch, match) та логічних виразів;
- ✓ проєктувати багаторазовий код через створення користувацьких функцій, враховуючи типи параметрів та області видимості;
- ✓ використовувати цикли (while, for, foreach) для автоматизації повторюваних дій та уникнення дублювання коду;

✓ керувати колекціями даних за допомогою простих, асоціативних та багатовимірних масивів;

✓ взаємодіяти з файловою системою для збереження та зчитування даних, включаючи роботу з форматом JSON;

✓ розуміти цикл HTTP "запит-відповідь", маршрутизацію та механізми передачі даних через GET та POST;

✓ проєктувати та обробляти безпечні вебформи з впровадженням логіки валідації та використанням "липких" (sticky) форм.

Для полегшення засвоєння матеріалу наводяться інструкції з конфігурації інструментів, готові лістинги коду та практичні вказівки до реалізації проетів.

Завершивши вивчення дисципліни, здобувач повинен уміти самостійно: проєктувати базову логіку вебскриптів; впроваджувати керуючі структури та користувацькі функції для обробки даних; професійно маніпулювати масивами та взаємодіяти з файловою системою сервера. Окрім цього, здобувач навчиться розробляти функціональні вебформи, забезпечувати безпечне зчитування та валідацію користувацького вводу. Це забезпечує необхідну базу для подальшого вивчення складних архітектурних патернів та сучасних серверних технологій.

Лабораторна робота №1
Тема: Основи PHP. Синтаксис PHP та PHP шаблони у HTML. Коментарі у PHP

Мета: Ознайомитися з основами мови PHP, її синтаксисом та принципами роботи з офіційною документацією. Навчитися інтегрувати PHP-код у HTML-шаблони, використовувати змінні для персоналізації виводу та керувати виконанням коду за допомогою коментарів.

Короткі теоретичні відомості

PHP (Hypertext Preprocessor) – це скриптова мова програмування, де інтерпретатор (PHP engine) перетворює код у машинний безпосередньо під час виконання.

Синтаксис: PHP-код відокремлюється від статичного тексту (наприклад, HTML) за допомогою відкриваючого тегу <?php та закриваючого тегу ?>. Якщо файл містить лише PHP-код, закриваючий тег рекомендується опускати.

Оператори: Кожна окрема команда в PHP (наприклад, echo або print) називається оператором і обов'язково має завершуватися крапкою з комою (;).

Коментарі: Використовуються для пояснення коду або його тимчасового вимкнення. Однорядкові коментарі починаються з //, багаторядкові — охоплюються блоком /* ... */

Змінні: Використовуються для зберігання даних і завжди починаються зі знака долара ($). Імена змінних чутливі до регістру: $name та $Name — це різні змінні.

Шаблонізація: PHP дозволяє змішувати статичний текст і динамічний код, що є основою створення інтерактивних вебсайтів.

Порядок виконання:

Крок 1. Робота з джерелами та стандартами. Ознайомтеся з офіційним сайтом PHP (php.net). Зверніть увагу на стандарти кодування PHP-FIG (зокрема PSR-1), які рекомендують відокремлювати декларації від логіки, що спричиняє побічні ефекти.

Крок 2. Налаштування середовища. Запустіть середовище розробки (Replit або локальну інсталяцію з IDE, наприклад VS Code). Переконайтеся, що PHP працює, виконавши стандартний скрипт "Hello, world!".

Крок 3. Керування виводом за допомогою коментарів. Створіть скрипт, який містить перелік слів (Cat, Elephant, Dog, Helicopter, Bus, Spacecraft). Використовуючи однорядкові та багаторядкові коментарі, вимкніть вивід так, щоб на екрані з'явилися лише слова: Cat, Dog, Helicopter.

Крок 4. Робота зі змінними та персоналізація. Напишіть скрипт, який створює змінну $name, присвоює їй ваше ім'я та виводить персоналізоване повідомлення у форматі "Мене звати [ваше ім'я]".

Зауваження стосовно налаштування середовища Replit:

Для роботи з PHP в онлайн-середовищі Replit необхідно створити новий проєкт («Repl»), обравши офіційний шаблон PHP CLI для консольних скриптів або PHP Web Server для вебзастосунків (Рис.1).

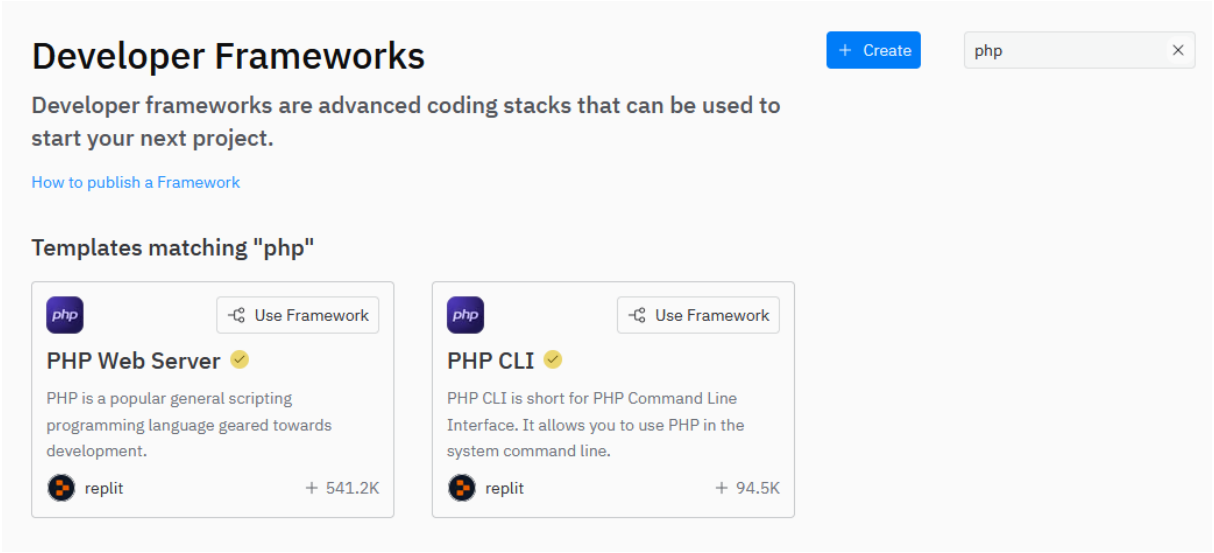

Рис.1. Створення проєкту у Replit

Контрольні запитання (самоконтроль)

1. Дайте відмінність між кодом PHP та шаблонним текстом (template text) на власному прикладі.
2. Як правильно використовувати коментарі (однорядкові та багаторядкові), аби вони не дублювали очевидний код?
3. Які критерії «гарного» імені для змінної (використання $, регістрозалежність, вибір між camelCase та snake_case)?
4. Чим константи відрізняються від змінних у PHP?
5. Які методи запуску PHP найдоцільніші для початкового етапу навчання і чому?

Лабораторна робота №2

Тема: Робота з різними типами даних

Мета: Вивчити скалярні типи даних PHP та спеціальний тип NULL. Навчитися використовувати механізми автоматичного жонглювання типами та явного приведення типів (casting) для коректної обробки інформації

Короткі теоретичні відомості

Скалярні типи: PHP має чотири типи даних, що можуть містити одне значення: цілі числа (int), числа з плаваючою комою (float), рядки (string) та логічні значення (bool)

Тип NULL: Спеціальний тип, що вказує на відсутність значення. Змінна оцінюється як NULL, якщо вона ніколи не мала значення, їй було присвоєно константу NULL або її було видалено функцією unset()

Жонглювання типами (Type Juggling): PHP автоматично перетворює типи залежно від контексту, наприклад, при математичних операціях із рядками.

Приведення типів (Type Casting): Явне перетворення значення шляхом вказання типу в дужках перед виразом, наприклад (int)55.9

Перевірка типів: Для перевірки приналежності до типу існують функції is_int(), is_float(), is_string(), is_bool(), is_null() та is_numeric()

var_dump(): Функція виводить детальну інформацію про тип та вміст змінної, що є критично важливим для налагодження.

Порядок виконання:

Крок 1. Явне приведення типів (Integer Casting). Напишіть скрипт, який створює змінну $scoreFloat зі значенням 55.9. Створіть другу змінну $scoreInt, присвоївши їй значення першої змінної, приведеної до цілого числа (int). Виведіть за допомогою var_dump() типи та значення обох змінних, щоб побачити результат відсікання дробової частини.

Крок 2. Дослідження NULL та функції unset(). Створіть скрипт, де змінній $age спочатку присвоюється число 21. Потім змініть її значення на NULL, а на останньому етапі видаліть змінну функцією unset(). Використовуйте var_dump() на кожному етапі та порівняйте результати (зокрема появу попередження "Undefined variable" після видалення).

Лістинг коду:

Console × index.php × Preview × +

Lab 2 > index.php

```php
<?php
$scoreFloat = 55.9;
$scoreInt = (int)$scoreFloat;

print "double scoreFloat = $scoreFloat\n";
print "integer scoreInt = $scoreInt\n";

$age = 21;
var_dump($age);

$age = NULL;
var_dump($age);

unset($age);
var_dump($age);
```

Контрольні запитання (самоконтроль)

1. Назвіть та коротко опишіть чотири скалярні типи даних, що використовуються в PHP.
2. Поясніть три конкретні ситуації, за яких змінна в PHP оцінюється як NULL
3. У чому полягає принципова різниця між «жонглюванням типами» (type juggling) та «приведенням типів» (type casting)?
4. Опишіть, як PHP виконує арифметичні операції в «числовому контексті», якщо один із операндів є рядком.
5. Яким буде результат порівняння рядка "1" та цілого числа 1 за допомогою операторів == та === відповідно? Поясніть чому.
6. Як працює оператор «космічний корабель» (<=>) і які три цілі числа він може повернути?
7. Назвіть функції, які використовуються для перевірки приналежності змінної до типів string, int, float та bool.
8. Як функція is_numeric() обробляє рядки, що містять лише цифри, порівняно з рядками, де цифри змішані з текстом?
9. Що відбувається з дробовою частиною числа (наприклад, 55.9) при його явному приведенні до типу int?
10. Чому для налагодження коду (debugging) рекомендується використовувати var_dump() замість print або echo?
11. Як PHP інтерпретує логічне значення false та порожній рядок при спробі виконати з ними арифметичне додавання?
12. Поясніть різницю у виводі var_dump() для змінної, якій присвоєно NULL, та змінної, яку було видалено функцією unset().
13. Які правила застосовує PHP при порівнянні двох рядків за допомогою операторів «більше» (>) або «менше» (<)?

Лабораторна робота №3

Тема: Робота з рядками та рядковими функціями

Мета: Освоїти методи оголошення рядків у PHP, навчитися виконувати конкатенацію та парсинг змінних. Закріпити навички використання вбудованих функцій для пошуку, заміни та форматування тексту.

Короткі теоретичні відомості

Типи рядків: Рядки в одинарних лапках (' ') сприймаються буквально (без парсингу змінних), тоді як рядки в подвійних лапках (" ") дозволяють інтерпретувати змінні та ескейп-послідовності (наприклад, \n — новий рядок).

Конкатенація: Об'єднання рядків виконується за допомогою оператора крапки (.). Оператор .= додає рядок до існуючого значення змінної.

Фігурні дужки {}: Використовуються в подвійних лапках для чіткого визначення імені змінної, якщо одразу після неї йдуть символи, які можуть бути частиною імені (наприклад, для створення множини: {$fruit}s).

Heredoc та Nowdoc: Конструкції для багаторядкових текстів. heredoc (починається з <<<EOT) виконує парсинг змінних, а nowdoc (починається з <<<'EOT') — ні.

Основні функції:

- strlen() — повертає довжину рядка.
- str_replace() — замінює всі входження підрядка.
- ucfirst(), strtolower(), strtoupper() — зміна регістру символів.
- trim() — видалення пробілів з країв рядка.
- strpos() — пошук позиції першого входження підрядка.

Порядок виконання:

Крок 1. Конкатенація рядків. Оголосіть змінну $name, яка містить ваше ім'я в одинарних лапках. Використовуючи оператор крапки (.), виведіть повідомлення у форматі: "[Ваше ім'я] is learning PHP".

Крок 2. Парсинг змінних та використання фігурних дужок. Створіть змінну $fruit зі значенням 'apple'. Використовуйте подвійні лапки та синтаксис фігурних дужок, щоб вивести речення: "apple juice is made from apples". Змініть значення на 'orange' і переконайтеся, що код працює коректно для множини.

Крок 3. Робота з багаторядковими текстами (Heredoc). Оголосіть змінну $happyMessage, використовуючи синтаксис heredoc. Текст повинен містити кілька рядків (наприклад, вірш або привітання). Виведіть вміст змінної.

Крок 4. Трансформація тексту та заміна підрядків. Створіть рядок "apple juice is made from apples.". Використайте str_replace(), щоб замінити "apple" на "grapefruit". До отриманого результату застосуйте ucfirst(), щоб речення починалося з великої літери.

Лістинг коду:

Console × index.php × Preview × +

Lab 3 > index.php

```php
<?php
$name = 'Demo';
print $name . ' is learning PHP' . "\n";

$fruit = 'apple';
print "{$fruit} juice is made from {$fruit}s.\n";

$happyMessage = <<<EOT
  This is a multi-line message,
  which was created using the heredoc syntax.
  It allows you to easily store text consisting
  of multiple lines in a single variable.
EOT;
print $happyMessage;

$phrase = 'apple juice is made from apples.';
$newPhrase = str_replace('apple', 'grapefruit', $phrase);
print ucfirst($newPhrase);
```

Контрольні запитання (самоконтроль)

1. Яка принципова різниця між рядками в одинарних та подвійних лапках?
2. Який оператор використовується в PHP для з’єднання рядків і чому це не символ +?
3. Поясніть призначення спеціальної константи PHP_EOL
4. У яких випадках необхідно використовувати фігурні дужки {} навколо імені змінної всередині рядка?
5. Що таке «ескейп-послідовність»? Наведіть приклади для табуляції та нового рядка.

6. Порівняйте синтаксис heredoc та nowdoc. Який із них дозволяє парсинг змінних?

7. Як працює індексація символів у рядку? Який індекс має перший символ?

8. Яка функція дозволяє дізнатися кількість символів у рядку?

9. Опишіть аргументи функції str_replace(). Чи змінює вона оригінальний рядок?

10. Чим відрізняються функції trim(), ltrim() та rtrim()?

11. Як за допомогою str_replace() видалити всі зайві пробіли або символи табуляції всередині рядка?

12. Поясніть призначення функцій ucfirst() та ucwords().

13. Що поверне функція strpos(), якщо шуканий підрядок не буде знайдено?

14. Як вивести символ долара ($) або подвійні лапки всередині рядка в подвійних лапках?

Лабораторна робота №4

Тема: Логічні оператори та логічні вирази

Мета: Закріпити навички написання динамічного коду, здатного приймати рішення на основі вхідних даних. Навчитися використовувати умовні структури (if, switch, match) та логічні оператори для реалізації складної алгоритмічної логіки.

Короткі теоретичні відомості

Логічні вирази: В основі будь-якого рішення лежить булевий вираз, що оцінюється як true (істина) або false (хиба).

Умовні оператори:

- if...else – виконує один блок коду, якщо умова істинна, та інший, якщо хибна.
- if...elseif...else – дає змогу перевіряти послідовно кілька умов.
- switch – порівнює змінну з набором значень (кейсами). Використовує нестроге порівняння (==).
- match – сучасна альтернатива switch (з PHP 8.x), що повертає значення та використовує строге порівняння (===)

Логічні оператори:

- AND (&&) – істинно, якщо обидва операнди істинні.
- OR (||) – істинно, якщо хоча б один операнд істинний.
- NOT (!) – заперечує значення виразу.
- XOR – істинно, якщо лише один із двох операндів є істинним

Спеціальні оператори:

- Тернарний оператор (? :) — скорочений запис if...else для присвоєння значень.
- Null-coalescing (??) — повертає перше значення, якщо воно не NULL, інакше — друге.

Порядок виконання:

Крок 1. Валідація довжини рядка. Напишіть скрипт, який присвоює значення змінній $name. Якщо довжина імені менша за чотири символи, виведіть повідомлення: "That is a short name".

Крок 2. Умовний вибір (Пральна машина). Створіть скрипт для вибору режиму прання. Якщо вага білизни ($laundryWeightKg) менша за 9 кг, виведіть "Fits in standard machine", інакше – "Needs medium to large machine".

Крок 3. Структури вибору (Транспорт). Використовуйте switch або match для виведення повідомлення залежно від типу транспортного засобу ($vehicle). Список значень: bus – "Beep beep", train — "Runs on tracks", car – "Has at least three wheels", helicopter – "Can fly", bicycle — "You never forget once you've learned". Для інших випадків – "You've chosen the road less traveled".

Крок 4. Перевірка облікових даних. Напишіть скрипт, який імітує вхід у систему. Використовуйте логічне AND, щоб вивести "You are now logged in" лише у випадку, якщо $userNameCorrect та $passwordCorrect обидва мають значення true.

Лістинг коду:

Console × index.php × Preview × +

Lab 4 > index.php

```php
<?php

$name = "Joe";
if (strlen($name) < 4) {
    print "That is a short name\n";
}

$laundryWeightKg = 10;
if ($laundryWeightKg < 9) {
    print "Fits in standard machine\n";
} else {
    print "Needs medium to large machine\n";
}

$vehicle = 'helicopter';
$message = match ($vehicle) {
    'bus' => "Beep beep",
    'train' => "Runs on tracks",
    'car' => "Has at least three wheels",
    'helicopter' => "Can fly",
    'bicycle' => "You never forget once you've learned",
    default => "You've chosen the road less traveled"
};
print $message . "\n";

$userNameCorrect = true;
$passwordCorrect = true;
if ($userNameCorrect && $passwordCorrect) {
    print "You are now logged in\n";
}
```

Контрольні запитання (самоконтроль)

1. Що таке булевий вираз і які значення він може приймати?
2. Яка різниця між операторами if...else та тернарним оператором ? :?

3. Поясніть призначення оператора elseif. Чим він відрізняється від вкладених if?

4. Яка роль ключового слова break у структурі switch? Що станеться, якщо його пропустити?

5. Чим match відрізняється від switch з точки зору типу порівняння значень?

6. Як працює логічний оператор XOR?

7. У чому різниця між використанням слів (and, or) та символів (&&, ||) у логічних операціях?

8. Як оператор заперечення ! впливає на логічний вираз?

9. Для чого використовується оператор нульового злиття (??)?

10. Як об’єднати кілька умов в одному операторі if?

11. Яка функція дозволяє дізнатися довжину рядка для використання в умовах?

12. Що таке "альтернативний синтаксис" для керуючих структур і коли його доцільно використовувати?

Лабораторна робота №5

Тема: Створення власних функцій та організація коду

Мета: Навчитися створювати власні функції для повторного використання коду (принцип DRY). Освоїти механізм розділення коду на декілька файлів, роботу з параметрами (зокрема необов'язковими), типами повернення та областю видимості змінних.

Короткі теоретичні відомості

Декларація функції: Починається з ключового слова function, після якого йде ім'я (зазвичай у snake_case), дужки з параметрами та тіло функції у фігурних дужках.

Розділення коду: Згідно зі стандартами (PSR-1), декларації (функції, класи) та побічні ефекти (вивід тексту, зміна змінних) краще тримати в різних файлах. Для підключення файлів використовується require_once.

Магічна константа __DIR__: Містить абсолютний шлях до поточної директорії файлу, що дозволяє будувати надійні шляхи до інших скриптів.

Параметри vs Аргументи: Параметри — це змінні в описі функції; аргументи — реальні значення, що передаються при виклику. Область видимості параметрів обмежена лише тілом функції (local scope).

Типи повернення:

- void – функція не повертає значення (але технічно повертає NULL).
- nullable (?int) – повертає вказаний тип або NULL.
- union (int|float) – дає змогу повертати декілька різних типів

Параметри за замовчуванням: Дають можливість робити аргументи необов'язковими під час виклику.

Порядок виконання:

Крок 1. Робота з декількома файлами. Створіть файл my_functions.php для збереження функцій. У головному файлі main.php підключіть його за допомогою require_once та магічної константи __DIR__.

Крок 2. Створення функції порівняння. У файлі my_functions.php оголосіть функцію which_is_larger(), яка приймає два цілих числа і повертає більше з них.

Крок 3. Гнучкість типів. Модифікуйте which_is_larger(), щоб вона приймала int або float (union types) та повертала NULL, якщо числа рівні.

Крок 4. Функція ASCII-арту. Створіть функцію (наприклад, для виводу великої літери вашого імені), яка використовує параметри за замовчуванням для символу малювання та символу пробілу.

Лістинг коду:

Console | main.php | my_functions....

Lab 5 > my_functions.php

```php
<?php

function which_is_larger(int|float $n1, int|float $n2): int|float|null
{
    if ($n1 > $n2) {
        return $n1;
    }
    if ($n2 > $n1) {
        return $n2;
    }
    return null;
}

function print_letter_a(string $char = 'A', string $spacer = ' '): void
{
    echo $spacer . $spacer . $char . $char . $spacer . $spacer . PHP_EOL;
    echo $spacer . $char . $spacer . $spacer . $char . $spacer . PHP_EOL;
    echo $char . $char . $char . $char . $char . $char . PHP_EOL;
    echo $char . $spacer . $spacer . $spacer . $spacer . $char . PHP_EOL;
}
```

Файл my_functions.php

Console | main.php | my_functions....

Lab 5 > main.php

```php
<?php

require_once __DIR__ . '/my_functions.php';

$res1 = which_is_larger(10, 20);
echo "Larger of 10 and 20: $res1" . PHP_EOL;

$res2 = which_is_larger(5.5, 5.5);
echo "Comparing 5.5 and 5.5: ";
var_dump($res2);

echo "Default letter:" . PHP_EOL;
print_letter_a();

echo "Letter with custom symbols (named arguments):" . PHP_EOL;
print_letter_a(char: '#', spacer: '.');
```

Файл main.php

Контрольні запитання (самоконтроль)

1. У чому полягає перевага використання require_once над require?
2. Яку проблему вирішує використання магічної константи __DIR__ при побудові шляхів?
3. Яку проблему вирішує використання магічної константи __DIR__ при побудові шляхів?
4. Яку проблему вирішує використання магічної константи __DIR__ при побудові шляхів?
5. Яку проблему вирішує використання магічної константи __DIR__ при побудові шляхів?
6. Поясніть різницю між термінами «параметр» та «аргумент».
7. Що таке «область видимості» (scope) змінних у контексті функцій?
8. Як працюють union types і у яких випадках їх доцільно використовувати?
9. Що означає тип повернення void?
10. У чому різниця між передачею аргументів за значенням та за посиланням (&)?
11. Яка роль ключового слова return у функціях, що повертають значення?
12. Чому рекомендується оголошувати функції в окремих файлах?

Лабораторна робота №6

Тема: Цикли while, do…while, for у PHP

Мета: Навчитися автоматизувати повторювані дії за допомогою циклічних конструкцій мови PHP. Освоїти механізми керування потоком виконання всередині циклів (оператори переривання та пропуску ітерацій), навчитися працювати з булевими прапорцями та обробляти інтерактивний ввід користувача.

Короткі теоретичні відомості

Цикл while: виконує блок коду доти, доки умова є істинною. Перевірка умови відбувається перед початком кожної ітерації. Якщо при першій перевірці умова хибна, тіло циклу не виконається жодного разу.

Цикл do...while: відрізняється тим, що умова перевіряється після виконання блоку коду. Це гарантує, що тіло циклу виконається принаймні один раз.

Цикл for: найчастіше використовується, коли кількість ітерацій відома заздалегідь. Його заголовок складається з трьох частин: ініціалізація лічильника, умова виконання та вираз оновлення (інкремент або декремент).

Керування циклами:

- break: негайно зупиняє виконання циклу та передає керування коду, що йде після нього. Часто використовується для виходу з нескінченних циклів while(true).
- continue: зупиняє поточну ітерацію та переходить до наступної (після перевірки умови або оновлення лічильника).

Булеві прапорці (Flags): це змінні, які використовуються як умова для циклу. Тіло циклу містить логіку, яка змінює значення прапорця на false для виходу.

Інтерактивність: Функція readline("prompt") дозволяє скрипту призупинити роботу та очікувати на ввід даних користувачем у консолі.

Порядок виконання:

Крок 1. Цикл із гарантованим виконанням (do...while). Напишіть скрипт, який повторно запитує у користувача слова за допомогою readline(). Цикл має зупинитися лише тоді, коли користувач введе слово, що починається з великої літери. Для перевірки використайте функцію ucfirst().

Крок 2. Нескінченний цикл та вихід за умовою (break). Створіть конструкцію while (true). Всередині циклу запитуйте будь-який текст. Якщо користувач введе числове значення (використайте перевірку is_numeric()), скрипт має вийти з циклу за допомогою break.

Крок 3. Фільтрація ітерацій у фіксованому циклі (continue). Реалізуйте цикл for, який ітерує числа від 1 до 21. Використовуйте оператор залишку від ділення % та команду continue, щоб надрукувати в консоль лише ті числа, які є кратними 3.

Лістинг коду:

Console × index.php × +

Lab 6 > index.php

```php
<?php

echo "--- Task 1: Enter capitalized word to stop ---\n";
do {
    $word = readline("Enter a word: ");
} while ($word !== ucfirst($word));

echo "Stopped! You entered: $word\n\n";

echo "--- Task 2: Enter any number to stop ---\n";
while (true) {
    $input = readline("Type something: ");
    if (is_numeric($input)) {
        echo "Numeric value detected. Exiting loop...\n";
        break;
    }
}
echo "Loop broken.\n\n";

echo "--- Task 3: Multiples of 3 (1 to 21) ---\n";
for ($i = 1; $i <= 21; $i++) {
    if ($i % 3 !== 0) {
        continue;
    }
    echo "Found multiple of 3: $i\n";
}
```

Контрольні запитання (самоконтроль)

1. У чому полягає основна відмінність між циклами while та do...while?
2. За яких умов тіло циклу while може не виконатися жодного разу?
3. Які три вирази складають заголовок циклу for?
4. Як уникнути виникнення «нескінченного циклу» при роботі з while?
5. Поясніть принцип дії оператора break. Як він змінює потік виконання програми?
6. Чим continue відрізняється від break при роботі з ітераціями?
7. Як працює функція readline() і яке значення вона повертає?
8. Що таке «булевий прапорець» (flag) і в яких випадках його зручно використовувати для керування циклом?
9. Чи можна використовувати декремент (зменшення лічильника) у циклі for? Наведіть приклад
10. Як реалізувати особливу логіку лише для останньої ітерації циклу for?
11. Який альтернативний синтаксис (без фігурних дужок) підтримує PHP для циклів у шаблонах?
12. Поясніть використання оператора % для фільтрації парних чи непарних чисел у циклі.
13. Що поверне функція is_numeric(), якщо користувач введе порожній рядок?

Лабораторна робота №7

Тема: Робота з масивами. Цикл foreach.

Мета: Опанувати навички створення та маніпулювання масивами різних типів (простими, асоціативними та багатовимірними). Навчитися ефективно ітерувати масиви за допомогою циклу foreach, використовувати вбудовані функції PHP для математичних обчислень, модифікації та сортування даних.

Короткі теоретичні відомості

Масив (Array): Це складений тип даних, який дозволяє зберігати групу пов’язаних значень під одним іменем змінної. В основі масиву лежить відображення значень на унікальні ключі.

Прості масиви (Simple Arrays): Використовують цілі числа як ключі, що автоматично призначаються починаючи з 0 (zero-based indexing).

Асоціативні масиви (Associative Arrays): Використовують рядки як ключі, що робить колекції даних більш осмисленими (наприклад, 'name' => 'John').

Багатовимірні масиви (Multidimensional Arrays): Це масиви, елементами яких є інші масиви. Це дає змогу створювати складні структури даних, такі як таблиці.

Цикл foreach: Спеціалізований цикл для ітерації колекцій. Синтаксис foreach ($array as $value) отримує лише значення, а foreach ($array as $key => $value) дозволяє працювати і з ключем, і зі значенням.

Оператор розпакування (Spread Operator ...): Дозволяє витягувати елементи одного масиву та вставляти їх індивідуально в інший масив.

Ключові функції:

- array_rand(): Повертає один або кілька випадкових ключів із масиву.
- array_pop(): Видаляє та повертає останній елемент масиву.
- count(): Повертає кількість елементів у масиві.
- sort(): Сортує значення масиву за зростанням.

Порядок виконання:

Крок 1. Випадковий вибір (Random Selection). Створіть масив кольорів і використайте array_rand(), щоб вибрати та надрукувати випадковий колір.

Крок 2. Ітерація Ключ/Значення (Key/Value Iteration). Напишіть цикл foreach, який виводить кожен колір з масиву разом із його числовим індексом.

Крок 3. Модифікація масиву (Array Modification). Використайте array_pop(), щоб видалити останній елемент масиву кольорів, і перевірте зміни за допомогою var_dump().

Крок 4. Математика з масивами (Math with Arrays). Створіть масив з віком кількох людей. Використайте count() та інші вбудовані підходи (наприклад, array_sum()), щоб обчислити та вивести кількість елементів та їхнє середнє значення.

Крок 5. Асоціативні масиви (Associative Arrays). Створіть масив, де ключами є імена людей, а значеннями — їхній зріст. Виведіть дані у зручному форматі.

Крок 6. Багатовимірні масиви (Multidimensional Arrays). Створіть вкладений масив для представлення даних про фільми (назва, тривалість, головний актор).

Крок 7. Комбінування та сортування (Combining and Sorting). Створіть два окремі масиви для непарних і парних чисел. Використайте оператор ... для їх об'єднання, а потім відсортуйте результат.

Лістинг коду:

Console × index.php Lab 6/ × index.php Lab 7/ ×

Lab 7 > index.php

```php
<?php

$colors = ['red', 'blue', 'green', 'yellow', 'purple'];
$randomKey = array_rand($colors);
echo "Random color picked: " . $colors[$randomKey] . PHP_EOL;

echo "\n--- Color List with Indices ---\n";
foreach ($colors as $index => $color) {
    echo "color $index = $color" . PHP_EOL;
}

echo "\n--- Array Modification (array_pop) ---\n";
$lastElement = array_pop($colors);
echo "Removed element: $lastElement" . PHP_EOL;
var_dump($colors);
```

```php

$ages = [19-21];
$count = count($ages);
$average = array_sum($ages) / $count;
echo "\nNumber of ages: $count" . PHP_EOL;
echo "Average age: " . number_format($average, 2) . PHP_EOL;

$heights = [
    'Fred' => 1.82,
    'Joelle' => 1.55,
    'Robin' => 1.70
];
echo "\n--- Heights List ---\n";
foreach ($heights as $name => $height) {
    echo "$name is $height meters tall" . PHP_EOL;
}

$movies = [
    ['title' => 'Back to the Future', 'duration' => 116, 'actor' => 'Michael J. Fox'],
    ['title' => 'The Fifth Element', 'duration' => 126, 'actor' => 'Bruce Willis'],
    ['title' => 'Alien', 'duration' => 117, 'actor' => 'Sigourney Weaver']
];
echo "\n--- Movie Details ---\n";
foreach ($movies as $movie) {
    echo "Movie: {$movie['title']}, Duration: {$movie['duration']} mins, Leading Actor: {$movie['actor']}" . PHP_EOL;
}

$odds = [14, 22-25];
$evens = [26-29];
$combined = [...$odds, ...$evens];
sort($combined);
echo "\n--- Sorted Combined Array ---\n";
echo implode(', ', $combined) . PHP_EOL;
```

Контрольні запитання (самоконтроль)

1. Дайте визначення масиву в PHP та поясніть концепцію відображення «ключ-значення»
2. У чому полягає різниця між простими та асоціативними масивами?
3. Який індекс має перший елемент у простому масиві за замовчуванням?
4. Поясніть роботу циклу foreach ($array as $key => $value). Для чого потрібна стрілка =>?
5. Що повертає функція array_rand() – значення чи ключ?
6. Як функція array_pop() змінює оригінальний масив? Чи передається масив у неї за посиланням?
7. Яка функція дозволяє дізнатися кількість елементів у масиві? Чи є у неї синоніми?
8. Як обчислити середнє арифметичне елементів масиву, використовуючи вбудовані функції?
9. Що таке багатовимірний масив і як отримати доступ до значень його внутрішніх елементів?
10. Поясніть призначення оператора розпакування (...). Що станеться, якщо додати масив до іншого масиву без цього оператора?

Лабораторна робота №8

Тема: Файлова система, файли та каталоги

Мета: Опанувати методи взаємодії PHP-скриптів із файловою системою сервера. Навчитися зчитувати та записувати дані у текстові файли, працювати з форматом JSON для обміну даними та автоматизувати пакетну обробку декількох файлів за допомогою шаблонів пошуку.

Короткі теоретичні відомості

Зчитування файлів:

- file_get_contents() – зчитує весь вміст файлу в один рядок.
- file() – зчитує файл у масив, де кожен елемент відповідає окремому рядку. Корисно використовувати прапорці FILE_IGNORE_NEW_LINES та FILE_SKIP_EMPTY_LINES для очищення даних.

Запис у файли:

- file_put_contents() – записує рядок у файл. За замовчуванням він перезаписує вміст. Щоб додати дані в кінець, використовується прапорець FILE_APPEND**.**

Робота з JSON:

- json_encode() — перетворює PHP-масив у формат JSON-рядка.
- json_decode($json, true) – перетворює JSON-рядок назад у асоціативний масив PHP.

Автоматизація (glob):

- Функція glob() повертає масив шляхів до файлів, що відповідають певному шаблону (наприклад, *.txt), що дозволяє обробляти сотні файлів у циклі.

Перевірка та безпека:

- Завжди слід перевіряти існування файлу через file_exists() або директорії через is_dir() перед початком роботи, щоб уникнути помилок виконання.
- __DIR__ – магічна константа, що вказує на абсолютний шлях до поточної папки скрипта, що робить роботу з файлами надійною.

Порядок виконання:

Крок 1. Запис масиву у текстовий файл. Оголосіть масив, де кожен елемент – це окремий рядок вірша. Використовуючи функцію implode() з

роздільником PHP_EOL та функцію file_put_contents(), запишіть цей вірш у новий файл poem.txt

Крок 2. Обробка JSON з віддаленого ресурсу. Напишіть скрипт, який зчитує JSON-рядок за допомогою file_get_contents() із тестового URL (наприклад, https://jsonplaceholder.typicode.com/todos/1). Перетворіть отриманий рядок у асоціативний масив та виведіть його структуру через var_dump()

Крок 3. Пакетна обробка файлів (Batch Processing). Створіть папку data та додайте туди 2-3 текстові файли, кожен з яких містить лише одне число (наприклад, бал гравця). Напишіть скрипт, який через glob() знаходить усі .txt файли в цій папці, зчитує їх вміст і обчислює загальну суму значень

Лістинг коду:

Console × index.php × Preview × +

Lab 8 > index.php

```php
<?php

$poemLines = [
    "What is with this code?",
    "Oh my, looks like I wrote it,",
    "What was I thinking?"
];
$filePath = __DIR__ . '/poem.txt';
$contentString = implode(PHP_EOL, $poemLines);
file_put_contents($filePath, $contentString);
echo "Task 1: Poem written to poem.txt\n\n";

$jsonUrl = 'https://jsonplaceholder.typicode.com/posts/1';
$jsonString = file_get_contents($jsonUrl);
if ($jsonString) {
    $dataArray = json_decode($jsonString, true);
    echo "Task 2: Data from JSON URL:\n";
    var_dump($dataArray);
}
echo "\n";
```

```php

$dataDir = __DIR__ . '/data/';
if (!is_dir($dataDir)) {
    mkdir($dataDir);
}
file_put_contents($dataDir . 'score1.txt', '55');
file_put_contents($dataDir . 'score2.txt', '99');
file_put_contents($dataDir . 'score3.txt', '101');

$files = glob($dataDir . '*.txt');
$grandTotal = 0;

foreach ($files as $file) {
    $content = file_get_contents($file);
    if (is_numeric(trim($content))) {
        $grandTotal += (int)$content;
    }
}
echo "Task 3: Processed " . count($files) . " files.\n";
echo "Grand Total Score: $grandTotal\n";
```

Контрольні запитання (самоконтроль)

1. Яка різниця між функціями file_get_contents() та file() при зчитуванні даних?
2. Як додати новий текст у кінець існуючого файлу, не видаляючи старі дані?
3. Навіщо використовувати магічну константу __DIR__ при вказанні шляхів до файлів?
4. Що повертає функція filesize() і в яких одиницях вимірюється результат?
5. Як перевірити, чи існує файл на диску, перш ніж намагатися його відкрити?
6. Опишіть призначення функцій json_encode() та json_decode().
7. Який другий аргумент потрібно передати в json_decode(), щоб отримати асоціативний масив замість об'єкта?
8. Як працює символ-wildcard * у функції glob()?
9. Для чого потрібні прапорці FILE_IGNORE_NEW_LINES та FILE_SKIP_EMPTY_LINES у функції file()??

Лабораторна робота №9

Тема: Маршрутизація. Обмін даними між клієнтом і сервером.

Мета: Ознайомитися з основами клієнт-серверної комунікації через протокол HTTP. Навчитися використовувати інструменти розробника браузера для аналізу заголовків і тіла запитів, а також опанувати еволюцію логіки шаблонізації в PHP: від чистого коду до використання коротких тегів виводу.

Короткі теоретичні відомості

Цикл HTTP «запит-відповідь»: Клієнт (браузер) надсилає запит на сервер, який обробляє його та повертає відповідь, що зазвичай містить статусний код, заголовки та тіло (наприклад, HTML-текст).

Методи GET та POST:

- GET використовується для отримання даних; змінні передаються відкрито в URL (query string).
- POST використовується для створення або зміни ресурсів; дані передаються приховано в тілі запиту.

Статусні коди: Сервер повертає тризначні коди, де 200 OK означає успіх, а 404 Not Found — відсутність ресурсу.

Маршрутизація (Routing): Процес, за якого сервер аналізує шлях запиту або змінні в URL, щоб вирішити, який скрипт виконати або який файл повернути.

Front Controller: Патерн проєктування, де єдиний скрипт (зазвичай index.php) обробляє всі вхідні запити та делегує їх іншим частинам системи.

Шаблонізація (Templating): PHP дозволяє змішувати статичний HTML з динамічним кодом. Використання коротких тегів виводу (<?= $var ?>) спрощує синтаксис, роблячи код чистішим і ближчим за виглядом до HTML

Порядок виконання:

Крок 1. Інспектування заголовків (Header Inspection). Відкрийте інструменти розробника браузера (Developer Tools) на вкладці "Network". Перейдіть на будь-який вебсайт та проаналізуйте заголовок HTTP GET запиту і тіло отриманої відповіді

Крок 2. Інспектування POST-даних. Знайдіть на будь-якому ресурсі вебформу (наприклад, форму пошуку або логіну), заповніть її та надішліть.

В інструментах розробника перегляньте змінні, надіслані методом POST у тілі запиту.

Лістинг коду

Console × index.php × Preview × +

Lab 9 > index.php

```php
<?php
    $pageTitle = 'Home Page';
    print '<!doctype html><html><head><title>';
    print $pageTitle;
    print '</title></head><body>';
    print '<h1>' . $pageTitle . '</h1>';
    print '<p>Welcome to pure PHP output.</p>';
    print '</body></html>';
?>

<?php
    $pageTitle = 'Home Page v2';
?>

<!doctype html>
<html>
<head>
    <title><?php print $pageTitle; ?></title>
</head>
<body>
    <h1><?php print $pageTitle; ?></h1>
    <p>Welcome to mixed PHP and HTML.</p>
</body>
</html>

<?php
    $pageTitle = 'Home Page v3';
?>

<!doctype html>
<html>
<head>
    <title><?= $pageTitle ?></title>
</head>
<body>
    <h1><?= $pageTitle ?></h1>
    <p>Welcome to short echo tags.</p>
</body>
</html>
```

Крок 3. Рефакторинг логіки шаблонізації. Реалізуйте один і той самий функціонал (вивід назви сторінки) трьома способами:

- Pure PHP: весь HTML виводиться через оператори print.
- Standard PHP blocks: стандартні блоки <?php ... ?> вперемішу з HTML-текстом.
- Short echo tags: використання <?= ... ?> для лаконічного виводу.

Контрольні запитання (самоконтроль)

1. Опишіть основні етапи циклу HTTP «запит-відповідь».
2. У чому полягає різниця між передачею даних методами GET та POST?
3. Що означає перша цифра в HTTP кодах статусів (наприклад, 2xx, 4xx)?
4. Для чого призначений статусний код 404?
5. Яка роль файлу index.php у патерні Front Controller?
6. Що таке «рядок запиту» (query string) і де він відображається?
7. Як за допомогою інструментів розробника переглянути заголовки HTTP?
8. Чому використання коротких тегів виводу <?= ?> вважається кращою практикою для шаблонів?
9. Поясніть концепцію «виводу без кешування» для POST-запитів
10. Чим статичний контент відрізняється від динамічного з точки зору сервера?
11. Як браузер вирішує, які додаткові файли (CSS, зображення) потрібно завантажити після отримання HTML?

Лабораторна робота №10

Тема: Побудова інтерактивних вебзастосунків. Робота з формами та валідація даних.

Мета: Синтезувати знання, отримані під час вивчення курсу. Навчитися проєктувати комплексні вебформи, обробляти різні типи вводу (текст, перемикачі, прапорці), впроваджувати логіку "липких" (sticky) форм та створювати надійну систему валідації помилок.

Короткі теоретичні відомості

Методи передачі (GET vs POST): GET використовується для запитів, що не змінюють стан сервера (наприклад, пошук), передаючи дані через URL. POST використовується для передачі конфіденційних даних або дій, що змінюють стан (реєстрація, замовлення), приховуючи дані в тілі запиту

Функція filter_input(): Сучасний стандарт отримання даних, який автоматично перевіряє існування змінної та дозволяє застосовувати фільтри санітизації (наприклад, FILTER_SANITIZE_SPECIAL_CHARS) для захисту від XSS-атак.

Логіка валідації: Процес перевірки даних на відповідність правилам (тип, довжина, обов'язковість). Найкраща практика — збір усіх повідомлень про помилки у масив для їх подальшого виведення.

Липкі форми (Sticky Forms): Техніка, при якій після невдалої валідації форма відображається знову, але вже заповнена введеними раніше даними, що покращує користувацький досвід

Масиви у формах: Прапорці (checkboxes) з однаковими іменами типу extras[] дозволяють PHP автоматично групувати вибрані значення у масив.

Порядок виконання:

Крок 1. Проєктування інтерфейсу. Створіть сторінку замовлення (наприклад, "Конфігуратор автомобіля"), яка включає:

- Текстове поле для імені.
- Поле для Email.
- Радіокнопки (Radio buttons) для вибору основного типу продукту.
- Набір прапорців (Checkboxes) для вибору додаткових опцій.

Крок 2. Реалізація Postback-логіки. Налаштуйте скрипт так, щоб він обробляв дані на тій же сторінці, де знаходиться форма.

Крок 3. Створення системи валідації. Впровадьте перевірку: ім'я має бути не коротшим за 3 символи, email має бути валідним, а числові поля (як-от вік) не повинні містити тексту.

Крок 4. Забезпечення "липкості" та виведення помилок. Додайте механізм, який зберігає введені дані в полях форми після натискання кнопки "Надіслати", та динамічно виводить список помилок червоним кольором угорі форми.

Рекомендації щодо реалізації проєкту

Архітектура обробки даних: Використовуйте змінну-прапорець (наприклад, $isSubmitted), яка перевіряє $_SERVER['REQUEST_METHOD'] === 'POST'. Це дозволить відокремити логіку першого завантаження сторінки від обробки надісланих даних.

Централізований збір помилок: Оголосіть порожній масив $errors = []. Для кожної перевірки (довжина імені, порожній email) додавайте новий рядок до цього масиву: $errors[] = 'Повідомлення...'. Якщо масив порожній після всіх перевірок, то дані вважаються валідними.

Реалізація "липкості" для різних полів:

- Для input type="text": виводьте значення змінної безпосередньо в атрибут value="<?= $name ?>". Не забудьте ініціалізувати змінну порожнім рядком на початку скрипта.
- Для radio та checkbox: використовуйте умову всередині тегу для додавання атрибута checked. Наприклад: <?php if($type === 'electric') echo 'checked'; ?>.

Безпека (Санітизація): Завжди використовуйте filter_input() з відповідними прапорцями. Наприклад, для Email використовуйте FILTER_SANITIZE_EMAIL.

Робота з масивами прапорців: При отриманні даних від групи прапорців використовуйте filter_input() з FILTER_REQUIRE_ARRAY. Це гарантує, що ви отримаєте масив, навіть якщо вибрано лише один пункт.

Зручність користувача: Використовуйте теги <label> з атрибутом for, що збігається з id поля вводу. Це дозволить активувати поле при натисканні на текст поруч із ним.

Контрольні запитання (самоконтроль)

1. У чому полягає основна відмінність між методами GET та POST при передачі даних форми?
2. Для чого в атрибуті name прапорців використовуються квадратні дужки (наприклад, options[])?
3. Що таке "санітизація" даних і чим вона відрізняється від "валідації"?
4. Яка перевага використання filter_input() над прямим зверненням до $_POST?
5. Як зробити текстове поле "липким" (щоб дані не зникали після помилки)?
6. Який атрибут тегу <input> дозволяє приховати символи пароля під час введення?
7. Що повертає функція empty() для рядка, який містить лише цифру "0"?
8. Як організувати виведення кількох повідомлень про помилки одночасно?
9. Яка роль атрибута action у тегу <form> і що станеться, якщо його не вказати?
10. Що таке суперглобальні масиви та які з них стосуються роботи з формами?

Список літератури

1. Smith M. *PHP Crash Course: The Complete, Modern, Hands-on Guide*. – San Francisco : No Starch Press, 2025. – 679 p.

2. PHP Manual : офіційна документація мови PHP [Електронний ресурс]. – Режим доступу: https://www.php.net/manual/ (дата звернення: 15.02.2026).

3. PHP the Right Way : збірник найкращих практик сучасного PHP-програмування [Електронний ресурс]. – Режим доступу: https://phptherightway.com/ (дата звернення: 15.02.2026).

4. PHP-FIG (PHP Framework Interop Group) : стандарти кодування PSR (PSR-1, PSR-3, PSR-4) [Електронний ресурс]. – Режим доступу: https://www.php-fig.org/psr/ (дата звернення: 15.02.2026).

5. OWASP (Open Web Application Security Project) : рекомендації з безпеки вебзастосунків [Електронний ресурс]. – Режим доступу: https://owasp.org (дата звернення: 15.02.2026).

Навчально-методичне видання

Web-програмування

Методичні вказівки
до виконання лабораторних робіт
для здобувачів першого (бакалаврського) рівня
вищої освіти за спеціальностями
F3 «Комп'ютерні науки»,
F6 «Інформаційні системи та технології»

Укладачі: **Науменко** Юрій Олександрович,
Хаддад Антон Георгійович

Випусковий редактор *Ю.М. Долгополова*
Комп’ютерне верстання *Ю.М. Долгополової*

Ум. друк. арк. 2,09. Обл.-вид. арк. 2,25
Електронний документ. Вид № 73/V-26.

Виконавець і виготовлювач

Київський національний університет будівництва і архітектури
Проспект Повітряних Сил, 31, Київ, Україна, 03037

Свідоцтво про внесення до Державного реєстру суб’єктів
видавничої справи ДК № 808 від 13.02.2002