**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**ДВНЗ «ПРИКАРПАТСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**
**ІМЕНІ ВАСИЛЯ СТЕФАНИКА»**

**І.М. ЛАЗАРОВИЧ, М. С. ДУТЧАК**

# Навчальний посібник

# з дисципліни «Програмування мовою PHP»

**Івано-Франківськ**
**2019 р**

УДК 004.43
ББК 32.973.2-018
Л17
*Рекомендовано до друку Вченою радою факультету математики та інформатики ДВНЗ «Прикарпатський національний університет імені Василя Стефаника» (протокол № 3 від 24 жовтня 2019 р.)*

***Рецензенти:***

**Кузь М. В.** – доктор технічних наук, професор кафедри інформаційних технологій факультету математики та інформатики ДВНЗ «Прикарпатський національний університет імені Василя Стефаника»;

**Пікуляк М.В.** – кандидат технічних наук, старший викладач кафедри інформаційних технологій факультету математики та інформатики ДВНЗ «Прикарпатський національний університет імені Василя Стефаника»

**Лазарович І.М.**

Л17 Навчальний посібник з дисципліни «Програмування мовою PHP» / Лазарович І.М., Дутчак М.С. – Івано-Франківськ: Видавництво ДВНЗ «Прикарпатський національний університет імені Василя Стефаника», 2019. – 282 с.

У навчальному посібнику наведено основні характеристики та тенденції розвитку сучасних технологій веб-програмування, загальний синтаксис мови PHP, методи для роботи з стандартними об’єктами, базові методи обробки числових масивів, рядків та хешів мовою PHP, роботу з файлами, методику взаємодії серверних сценарії та баз даних, механізми створення і роботи сесій. Теоретичні відомості супроводжуються значною кількістю прикладів, а завдання до лабораторних робіт спрямовані на закріплення отриманих теоретичних знань.

Навчальний посібник розроблено на основі вимог освітньої програми підготовки фахівців за спеціальністю «Інженерія програмного забезпечення». Наведений матеріал може бути використаний студентами інших спеціальностей, які вивчають веб-програмування і мають базові навики веб-дизайну та розробки сторінок на основі HTML, CSS та JavaScript.

**УДК 004. 43**
**ББК 32.973.2-018**
**Л17**

# Зміст

# ПЕРЕДМОВА

Всесвітня мережа Інтернет – це середовище спілкування і інформаційного обміну між мільйонами людей, що живуть в різних країнах, на різних півкулях. Вони розміщують в Інтернеті, на Web-серверах або пересилають по електронній пошті різну інформацію – текст, малюнки, відеозображення, звуковий супровід, мультимедійні чи будь-які інші дані і поступово ускладнюють свої Web-сторінки, роблячи їх більше інтерактивними і динамічними, з встановлення установки зворотного зв'язку.

Такий розвиток Інтернет став можливим завдяки технологіям веб-програмування, які передбачають процес створення веб-сайту або веб-додатку. Основними етапами процесу є веб-дизайн, верстання сторінок, програмування для веб на стороні клієнта і сервера.

Конспект містить десять тем, в межах даної роботи розглянуто архітектури типових веб-застосувань, описано програмування на стороні сервера з використанням мови PHP. Описано загальний синтаксис мови PHP, основні мовні конструкції, способи визначення функцій, забезпечення роботи з базами даних на прикладі СУБД MySQL. Також наведено відомості по роботі із сесіями та файлами cookies. Значну увагу приділено обробці масивів та рядків за допомогою стандартних методів PHP, також розглянуто особливості синтаксису та застосування регулярних виразів для формування шаблонів пошуку. Наведений теоретичний матеріал супроводжується значною кількістю прикладів. Кожна тема супроводжується контрольними питаннями для самоперевірки.

Наведений матеріал призначений для студентів, які вивчають веб-програмування, може бути використаний ними для самостійної роботи, а також для підготовки до виконання лабораторних і практичних робіт. Для розуміння та засвоєння поданих відомостей необхідно володіти навиками розробки веб-сторінок на основі HTML, CSS та Javascript.

# Тема 1. Архітектура типових веб-застосувань

## 1.1 Поняття архітектури ПЗ

Архітектура програмного забезпечення (англ. software architecture) — це структура програми або обчислювальної системи, яка містить програмні компоненти, видимі зовні властивості цих компонентів, а також відносини між ними. Цей термін також відноситься до документування архітектури програмного забезпечення. Документування архітектури ПЗ спрощує процес комунікації між зацікавленими особами, дозволяє зафіксувати прийняті на ранніх етапах проектування рішення про високорівневий дизайн системи і дозволяє використовувати компоненти цього дизайну і шаблони повторно в інших проектах.

Компоненти інформаційної системи по виконуваних функціях можна розділити на три шари: шар представлення, шар бізнес-логіки і шар доступу до даних (рис.1.1).

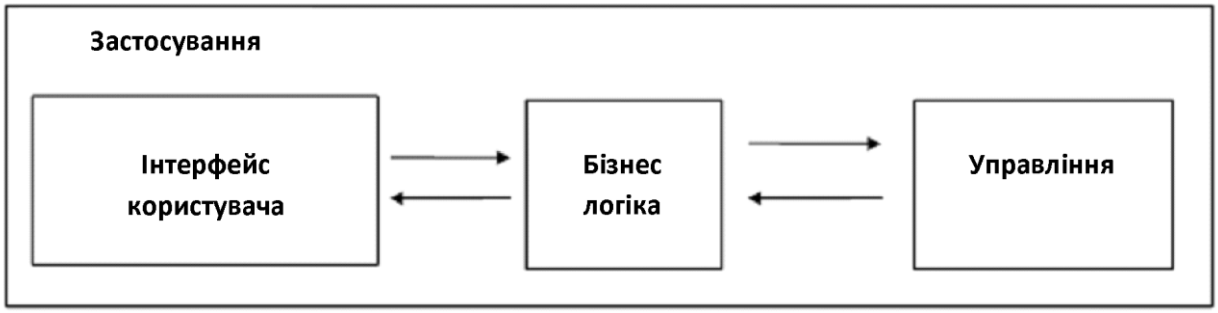



Рисунок 1.1 Компоненти інформаційної системи

Шар представлення – все, що пов’язано зі взаємодією з користувачем: натиснення кнопок, рух миші, отрисовка зображення, виведення результатів пошуку і так далі

Бізнес логіка – правила, алгоритми реакції додатка на дії користувача або на внутрішні події, правила обробки даних.

Шар доступу до даних – зберігання, вибірка, модифікація і видалення даних, пов’язаних з вирішуваною додатком прикладним завданням

## 1.2 Види архітектур сучасних програмних додатків

### 1.2.1 Централізована і файл-серверна архітектури

Історично першими з’явилися комп’ютерні системи з централізованою архітектурою додатків. При використанні цієї архітектури усе програмне забезпечення автоматизованої системи виконується централізовано на одному комп’ютері, що виконує одночасно багато завдань і що підтримує велику кількість користувачів. На цьому комп’ютері повністю здійснюється процес введення/ виведення інформації, а також її прикладна обробка. В якості центрального комп’ютера для такої системи може застосовуватися або велика ЕОМ(що називається також майнфреймом) або так звана МІНІ-ЕОМ. До подібного комплексу підключаються периферійні пристрої для введення/виведення інформації від кожного користувача.

Наступний етап розвитку архітектури ПЗ — файл-сервер — це виділений сервер, призначений для виконання файлових операцій введення-виводу і зберігаючий файли будь-якого типу (рис. 1.2). Як правило, має великий об’єм дискового простору, реалізованому у формі RAID— масиву для забезпечення безперебійної роботи і підвищеної швидкості запису і читання даних.

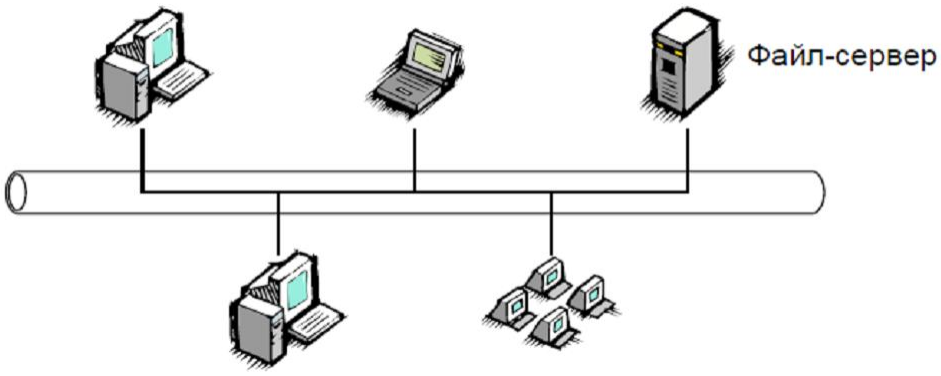



Рисунок 1.2 Файл-серверна архітектура

Файл-серверні додатки – додатки, схожі по своїй структурі з локальними застосуваннями що використовують мережевий ресурс для зберігання даних у вигляді окремих файлів. Функції сервера у такому разі зазвичай обмежуються зберіганням даних(можливо також зберігання виконуваних файлів), а обробка даних відбувається виключно на стороні клієнта (рисунок 1.3). Кількість клієнтів обмежена десятками зважаючи на неможливість одночасного доступу на запис до одного файлу. Проте клієнтів може бути в рази більше, якщо вони звертаються до файлів виключно в режимі читання.

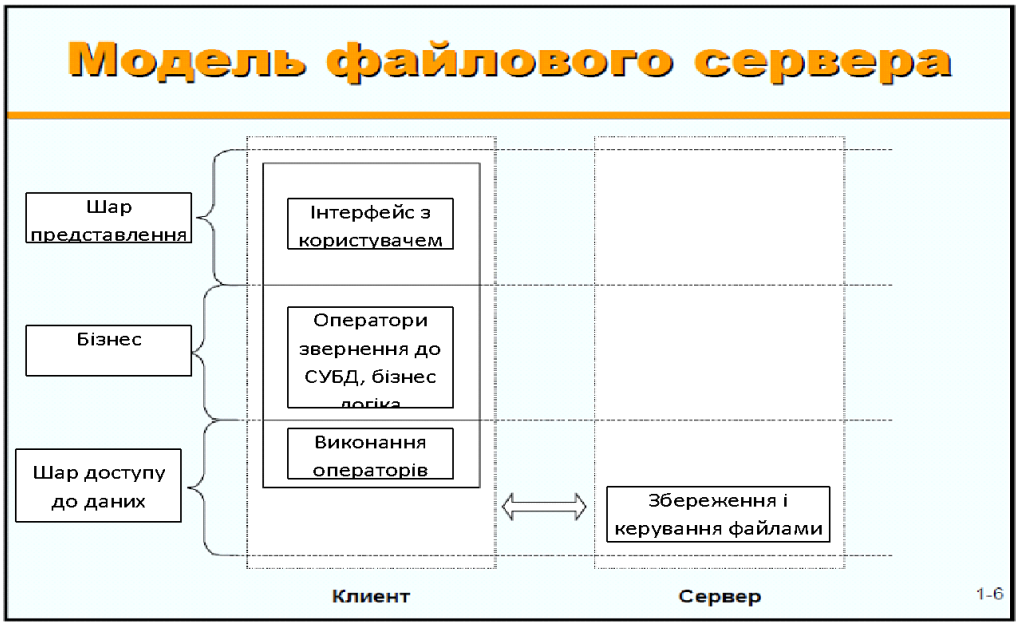



Рисунок 1.3 Модель файлового сервера

Переваги:

- низька вартість розробки;
- висока швидкість розробки;
- невисока вартість оновлення і зміни ПЗ.

Недоліки:

- зростання числа клієнтів різко збільшує об’єм трафіку і навантаження на мережі передачі даних;
- високі витрати на модернізацію і супровід сервісів бізнес-логіки на кожній клієнтській робочій станції;
- низька надійність системи.

### 1.2.2 Архітектура клієнт-сервер

Архітектура є одним із архітектурних шаблонів програмного забезпечення та є домінуючою концепцією у створенні розподілених мережних застосунків і передбачає взаємодію та обмін даними між ними (рисунок 1.4). Вона передбачає такі основні компоненти:

- набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;
- набір клієнтів, які використовують сервіси, що надаються серверами;
- мережа, яка забезпечує взаємодію між клієнтами та серверами.

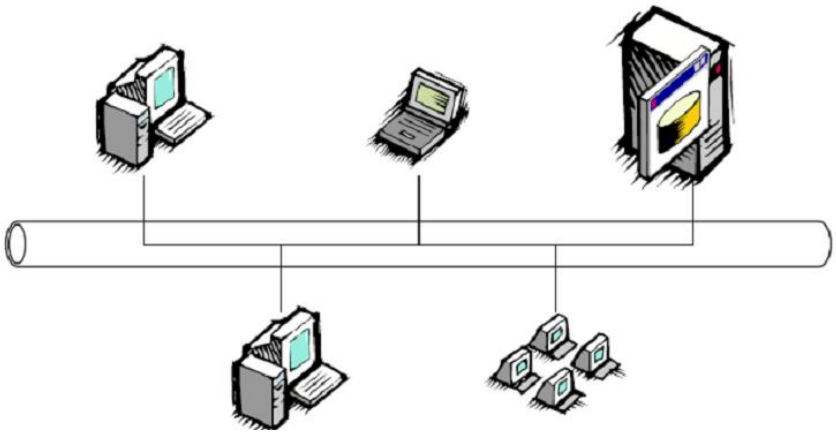


Рисунок 1.4 Архітектура клієнт-сервер

Сервери є незалежними один від одного. Клієнти також функціонують паралельно і незалежно один від одного. Немає жорсткої прив'язки клієнтів до серверів. Більш ніж типовою є ситуація, коли один сервер одночасно обробляє запити від різних клієнтів; з другого боку, клієнт може звертатися то до одного сервера, то до іншого. Клієнти мають знати про доступні сервери, але можуть не мати жодного уявлення про існування інших клієнтів.

Загальноприйнятим є положення, що клієнти та сервери – це перш за все програмні модулі. Найчастіше вони знаходяться на різних комп'ютерах, але бувають ситуації, коли обидві програми – і клієнтська, і серверна, фізично розміщуються на одній машині; в такій ситуації сервер часто називається локальним.

Модель клієнт-серверної взаємодії визначається перш за все розподілом обов'язків між клієнтом та сервером (рисунок 1.5). Логічно можна виокремити три рівні операцій:

- рівень представлення даних, який по суті являє собою інтерфейс користувача і відповідає за представлення даних користувачеві і введення від нього керуючих команд;
- прикладний рівень, який реалізує основну логіку застосунку і на якому здійснюється необхідна обробка інформації;
- рівень управління даними, який забезпечує зберігання даних та доступ до них.

Дворівнева клієнт-серверна архітектура передбачає взаємодію двох програмних модулів – клієнтського та серверного. В залежності від того, як між ними розподіляються наведені вище функції, розрізняють:

- модель тонкого клієнта, в рамках якої вся логіка застосунку та управління даними зосереджена на сервері. Клієнтська програма

забезпечує тільки функції рівня представлення;

- модель товстого клієнта, в якій сервер тільки керує даними, а обробка інформації та інтерфейс користувача зосереджені на стороні клієнта. Товстими клієнтами часто також називають пристрої з обмеженою потужністю: кишенькові комп'ютери, мобільні телефони та ін.

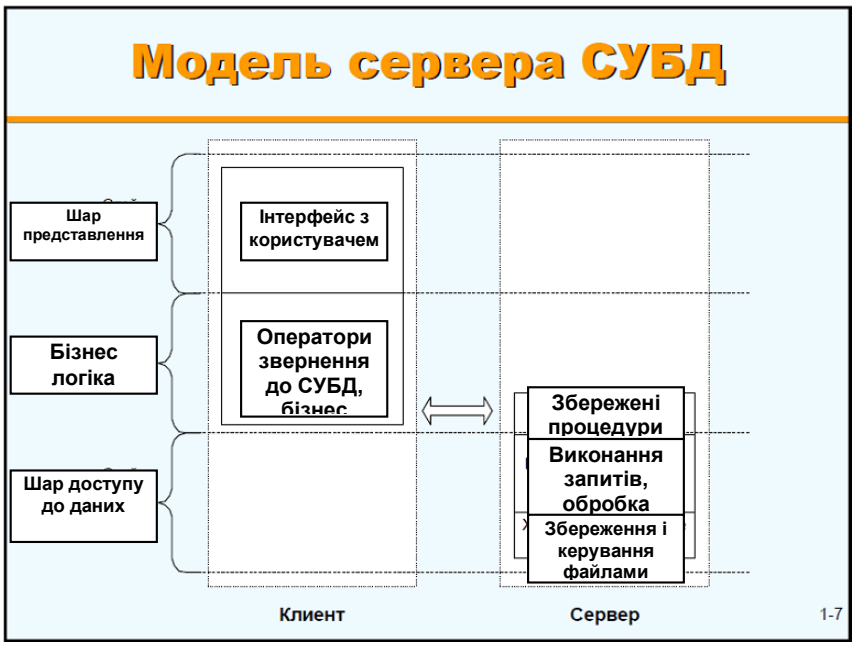



Рисунок 1.5 Модель архітектури клієнт-сервер

Переваги:

- повна підтримка розрахованої на багато користувачів роботи;
- гарантія цілісності даних.

Недоліки:

- бізнес логіка додатків залишилася в клієнтському ПЗ. При будь-якій зміні алгоритмів, потрібно оновлювати призначене для користувача ПЗ на кожному клієнтові;
- високі вимоги до пропускної спроможності комунікаційних каналів з сервером, що перешкоджає використання клієнтських станцій інакше як в локальній мережі;
- слабкий захист даних від злому, особливо від недобросовісних користувачів системи;
- висока складність адміністрування і налаштування робочих місць користувачів системи;
- необхідність використати потужні ПК на клієнтських місцях;
- висока складність розробки системи через необхідність

виконувати бізнес-логіку і забезпечувати призначений для користувача інтерфейс в одній програмі.

### 1.2.3 Трьохрівнева архітектура

Трьохрівнева архітектура, або триланкова архітектура (англ. three-tier або англ. Multitier architecture) — архітектурна модель програмного комплексу, що припускає наявність в нім трьох компонентів : клієнтського застосування(що зазвичай називається «тонким клієнтом» або терміналом), сервера додатків, до якого підключено клієнтське застосування, і сервера бази даних, з яким працює сервер додатків (рисунок 1.6).


Рисунок 1.6 Модель трьохрівневої архітектури

Клієнт — це інтерфейсний(зазвичай графічний) компонент, який представляє перший рівень, власне додаток для кінцевого користувача. Перший рівень не повинен мати прямих зв’язків з базою даних(за вимогами безпеки), бути навантаженим основною бізнес-логікою(за вимогами масштабованості) і зберігати стан додатка(за вимогами надійності). На перший рівень може бути винесена і зазвичай виноситься проста бізнес-логіка: інтерфейс авторизації, алгоритми шифрування, перевірка значень, що вводяться, на допустимість і відповідність формату, нескладні операції(сортування, угрупування, підрахунок значень) з даними, вже завантаженими на термінал.

Сервер додатків розташовується на другому рівні. На другому рівні

зосереджена бóльшая частина бізнес-логіки. Поза ним залишаються фрагменти, що експортуються на термінали (див. вище), а також занурені в третій рівень процедури, що зберігаються, і тригери.

*Сервер бази даних* забезпечує зберігання даних і виноситься на третій рівень. Звичайно це стандартна реляційна або об'єктно-орієнтована СУБД. Якщо третій рівень є базою даних разом з процедурами, що зберігаються, тригерами і схемою, що описує додаток в термінах реляційної моделі, то другий рівень будується як програмний інтерфейс, що зв'язує клієнтські компоненти з прикладною логікою бази даних.

У простій конфігурації фізично сервер додатків може бути поєднаний з сервером бази даних на одному комп'ютері, до якого по мережі підключається один або декілька терміналів.

У «правильній»(з крапки зору безпеки, надійності, масштабування) конфігурації сервер бази даних знаходиться на виділеному комп'ютері(чи кластері), до якого по мережі підключені один або декілька серверів додатків, до яких, у свою чергу, по мережі підключаються термінали.

В порівнянні з клієнт-серверною або файл-серверной архітектурою можна виділити наступні переваги трирівневої архітектури :

- масштабованість
- конфігурується — ізольованість рівнів один від одного дозволяє(при правильному розгортанні архітектури) швидко і простими засобами переконфігурувати систему при виникненні збоїв або при плановому обслуговуванні на одному з рівнів
- висока безпека
- висока надійність
- низькі вимоги до швидкості каналу(мережі) між терміналами і сервером додатків
- низькі вимоги до продуктивності і технічних характеристик терміналів, як наслідок зниження їх вартості. Терміналом може виступати не лише комп'ютер, але і, наприклад, мобільний телефон.

Недоліки витікають з переваг. По порівнянню с клієнт-серверної або файл-серверной архітектурою можна виділити наступні недоліки трирівневої архітектури :

- більш висока складність створення додатків;
- складніше в розгортанні і адмініструванні;
- високі вимоги до продуктивності серверів додатків і сервера бази

даних, а, означає, і висока вартість серверного устаткування;

- високі вимоги до швидкості каналу(мережі) між сервером бази даних і серверами додатків.

### 1.2.4 Розподілені інформаційні системи

У літературі можна знайти різні визначення розподілених систем, причому жодне з них не є задовільним і не узгоджується з іншими. Для наших завдань вистачить досить вільної характеристики.

Розподілена система — це набір незалежних обчислювальних машин, що представляється їх користувачам єдиною об'єднаною системою (рисунок 1.7). У цьому визначенні обмовляються два моменти. Перший відноситься до апаратури: усі машини автономні.

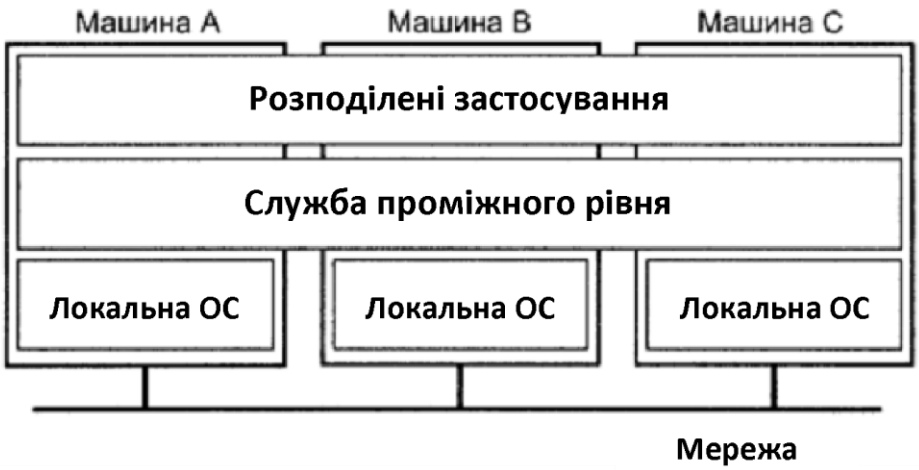



Рисунок 1.7 Модель розподіленої системи

Другий торкається програмного забезпечення: користувачі думають, що мають справу з єдиною системою. Важливі обидва моменти. Пізніше в цій главі ми до них повернемося, але спочатку розглянемо деякі базові питання, що стосуються як апаратного, так і програмного забезпечення.

Характеристики розподілених систем :

- від користувачів приховані відмінності між комп'ютерами і способи зв'язку між ними. Те ж саме відноситься і до зовнішньої організації розподілених систем.
- користувачі і додатки однаково працюють в розподілених системах, незалежно від того, де і коли відбувається їх взаємодія.

Розподілені системи повинні також відносно легко піддаватися розширенню, або масштабуванню. Ця характеристика є прямим наслідком наявності незалежних комп'ютерів, але в той же час не вказує, яким чином ці

комп’ютери насправді об’єднуються в єдину систему.

Розподілені системи зазвичай існують постійно, проте деякі їх частини можуть тимчасово виходити з ладу. Користувачі і додатки не повинні повідомлятися про те, що частини системи замінені або полагоджені або, що додані нові для підтримки додаткових користувачів.

Для того, щоб підтримати представлення системи в єдиному виді, організація розподілених систем часто включає додатковий рівень програмного забезпечення, що знаходиться між верхнім рівнем, на якому знаходяться користувачі і додатки, і нижнім рівнем, що складається з операційних систем.

### 1.2.5 Сервісно-орієнтована архітектура

Сервісно-орієнтована архітектура (англ. Service-oriented architecture, SOA) — архітектурний шаблон програмного забезпечення, модульний підхід до розробки програмного забезпечення, заснований на використанні розподілених, слабко пов’язаних замінних компонентів, оснащених стандартизованими інтерфейсами для взаємодії за стандартизованими протоколам.

Дуже часто становлення того чи іншого підходу супроводжується появою невірних або хибних трактувань, як це було, наприклад, з концепцією федеративного сховища даних. Не оминуло стороною це і сервіс-орієнтовану архітектуру. Так вважає представник компанії BEA Джерімі Уестерман (Jeremy Westerman). Саме тому в одній із своїх статей, присвячених SOA, він спеціально зупиняється на «проблемних місцях» і наводить докладні пояснення:

SOA не є чимось новим: IT-відділи компаній успішно створювали і розгортали додатки, що підтримують сервіс-орієнтовану архітектуру, вже багато років — задовго до появи XML і Web-сервісів.

SOA — це не технологія, а спосіб проектування і організації інформаційної архітектури та бізнес-функціональності.

Купівля найновіших продуктів, що реалізують XML і Web-сервіси, не означає побудову додатків згідно з принципами SOA.

Джерімі Уестерман дає наступне визначення SOA: це парадигма, призначена для проектування, розробки та управління дискретних одиниць логіки (сервісів) в обчислювальному середовищі. Застосування цього підходу вимагає від розробників проектування додатків як набору сервісів, навіть якщо переваги такого рішення відразу неочевидні. Розробники повинні вийти за

межі своїх додатків і подумати, як скористатися вже існуючими сервісами, або вивчити, як їх сервіси можуть бути використані їх колегами.

SOA підштовхує до використання альтернативних технологій і підходів (таких як обмін повідомленнями) для побудови додатків за допомогою зв'язування сервісів, а не за допомогою написання нового програмного коду. У цьому випадку, при належному проектуванні, застосування повідомлень дозволяє компаніям своєчасно реагувати на зміну ринкових умов — настроювати процес обміну повідомленнями, а не розробляти нові програми.

Ще до недавніх пір термін «сервіс-орієнтована архітектура» був синонімом «Web-сервіс». SOA — виклик Web-сервісів за допомогою засобів і мов управління бізнес-процесами. SOA — це термін, який з'явився для опису виконуваних компонентів — таких як Web-сервіси — які можуть викликатися іншими програмами, які виступають у якості клієнтів або споживачів цих сервісів. Ці сервіси можуть бути повністю сучасними — або навіть застарілими — прикладними програмами, які можна активізувати як чорний ящик. Від розробника не потрібно знати, як працює програма, необхідно лише розуміти, які вхідні та вихідні дані потрібні, і як викликати ці програми для виконання. У найзагальнішому вигляді SOA припускає наявність трьох основних учасників: постачальника сервісу, споживача сервісу та реєстру сервісів. Взаємодія учасників виглядає досить просто: постачальник сервісу реєструє свої сервіси в реєстрі, а споживач звертається до реєстру із запитом.

Для використання сервісу необхідно дотримуватися угоди про інтерфейс для звернення до сервісу — інтерфейс повинен не залежати від платформи. SOA реалізує масштабованість сервісів — можливість додавання сервісів, а також їх модернізацію. Постачальник сервісу і його споживач виявляються непов'язаними — вони спілкуються за допомогою повідомлень. Оскільки інтерфейс повинен не залежати від платформи, то і технологія, використовувана для визначення повідомлень, також повинна не залежати від платформи. Тому, як правило, повідомлення є XML-документами, які відповідають XML-схемі.

Дійсно, відкриті стандарти, що описують XML і Web-сервіси, дозволяють застосовувати SOA до всіх технологій і додатків, встановлених в компанії. Як відомо, Web-сервіси базуються на широко поширених і відкритих протоколах: HTTP, XML, UDDI, WSDL і SOAP. Саме ці стандарти реалізують основні вимоги SOA — по-перше, сервіс повинен піддаватися динамічному виявленню і викликом (UDDI, WSDL і SOAP), по-друге, повинен використовуватися незалежний від платформи інтерфейс (XML). Нарешті,

HTTP забезпечує функціональну сумісність.

Переваги використання SOA

Перш ніж, перерахувати переваги використання SOA, буде доречним нагадати, що переваги бувають різні: стратегічні і тактичні. SOA має ряд переваг як стратегічних, так і тактичних.

Стратегічна цінність SOA:

- скорочення часу реалізації проектів, або «часу виходу на ринок»;
- підвищення продуктивності;
- більш швидка і менш дорога інтеграція додатків і інтеграція B2B — зупинимося більш детально на цьому пункті.

Тактичні переваги SOA:

- простіша розробка і впровадження додатків;
- використання поточних інвестицій;
- зменшення ризику, пов’язаного з впровадженням проектів в області автоматизації послуг і процесів;
- можливість безперервного поліпшення наданої послуги;
- скорочення числа звернень за технічною підтримкою;
- підвищення показника повернення інвестицій (roi);
- перспективи.
-

**Контрольні питання до теми 1**

1. Що таке архітектура програмного забезпечення
2. З чого складається інформаційна система
3. Які є види сучасного ПЗ?
4. Порівняйте файл-серверну та клієнт-серверну архітектуру.
5. Переваги і недоліки трирівневої архітектури.
6. Принцип роботи розподілених систем.
7. SOA, її суть і особливості.

# Тема 2. Загальні відомості про мову PHP

## 2.1 Вступ в PHP

Мова PHP є однією з найпоширеніших мов, що використовуються у сфері веб-розробок (разом із Java, .NET, Perl, Python, Ruby). PHP (від англ. PHP: Hypertext Preprocessor — PHP: гіпертекстовий препроцесор), попередня назва: Personal Home Page Tools — скриптова мова програмування, була створена для генерації HTML-сторінок на стороні веб-сервера.

PHP інтерпретується веб-сервером у HTML-код, який передається на сторону клієнта. На відміну від скриптової мови JavaScript, користувач не бачить PHP-коду, бо браузер отримує готовий html-код. Це є перевага з точки зору безпеки, але погіршує інтерактивність сторінок. Але ніхто не забороняє використовувати PHP для генерування JavaScript-кодів, які виконуються вже на стороні клієнта.

Мова PHP була розроблена як інструмент для вирішення чисто практичних завдань. Його творець, Расмус Лердорф, хотів знати, скільки чоловік читають його online-резюме, і написав для цього простеньку CGI – оболонку на мові Perl, тобто це був набір Perl – скриптів, призначених виключно для певної мети – збору статистики відвідувань. Для довідки – CGI (Common Gateway Interface – загальний інтерфейс шлюзів) є стандартом, який призначений для створення серверних застосувань, працюючих по протоколу HTTP. Такі застосування (їх називають шлюзами або CGI – програмами ) запускаються сервером в режимі реального часу. Сервер передає запити користувача CGI – програмі, яка їх обробляє і повертає результат своєї роботи на екран користувача. Таким чином, відвідувач отримує динамічну інформацію, яка може змінюватися в результаті впливу різних чинників. Сам шлюз ( скрипт CGI ) може бути написаний на різних мовах програмування – Ci/C++, Fortran, Perl, TCL, UNIX Shell, Visual Basic, Python та ін.

Незабаром з’ясувалося, що оболонка має невелику продуктивність, і довелося переписати її наново, але вже на мові Ci. Після цього первинники були викладені на загальний огляд для виправлення помилок і доповнення. Користувачі сервера, де розташовувався сайт з першою версією PHP, зацікавилися інструментом, з’явилися охочі його використати. Так що скоро PHP перетворився на самостійний проект, і на початку 1995 року вийшла перша відома версія продукту, Personal Home Page Tools (засоби для персональної домашньої сторінки), що називалася.

До середини 1995 року після грунтовної переробки з’явилася друга

версія продукту, названа PHP/FI (Personal Home Page / Forms Interpreter – персональна домашня сторінка/ інтерпретатор форм).

У 1997 вийшла друга версія Сі –реализація PHP – PHP/FI 2.0. До того моменту PHP використали вже декілька тисяч чоловік по всьому світу, приблизно з 50 тис. доменів, що складало близько 1% усього числа доменів Internet.

PHP 3.0 була першою версією, PHP, що нагадує, яким ми знаємо його сьогодні. Він дуже сильно відрізнявся від PHP/FI 2.0 і з'явився знову ж таки як інструмент для вирішення конкретного прикладного завдання. Його творці, Енді Гутманс (Andi Gutmans) і Зів Сураски (Zeev Suraski), в 1997 році переписали наново код PHP/FI, оскільки він здався ним непридатним для розробки додатка електронної комерції, над яким вони працювали. Однією з сильних сторін PHP 3.0 була можливість розширення ядра. Саме властивість розширюваності PHP 3.0 притягнула увагу безлічі розробників, бажаючих додати свій модуль розширення. Крім того, PHP 3.0 надавала широкі можливості для взаємодії з базами даних, різними протоколами і API.

До кінця 1998 року число користувачів PHP зросло до десятків тисяч. Сотні тисяч web-сайтів повідомляли про те, що вони працюють з використанням цієї мови. Майже на 10% серверів Internet був встановлений PHP 3.0. Нове ядро було назване «Zend Engine» (від імен творців : Zeev і Andi) і уперше представлено в середині 1999 року. PHP 4.0, заснований на цьому ядрі і такий, що приніс з собою набір додаткових функцій, офіційно вийшов в травні 2000 року, майже через два роки після свого попередника, PHP 3.0.

Нині ведуться роботи по поліпшенню Zend Engine і впровадженню нововведень в PHP 5.0. Одна з істотних змін сталася в об'єктній моделі мови, її грунтовно підлатали і додали багато нових можливостей.

Шоста версія PHP розроблялася з жовтня 2006 року. Було зроблено безліч нововведень, як, наприклад, виключення з ядра регулярних виразів POSIX і «довгих» суперглобальних масивів, видалення директив safe_mode, magic_quotes_gpc і register_globals з конфігураційного файлу php.ini. Одним з основних нововведень повинна була стати підтримка Юнікода. Проте у березні 2010 року розробка PHP6 була визнана безперспективною із-за складнощів з підтримкою Юнікоду. Початковий код PHP6 переміщений на гілку, а основною лінією розробки стала версія 5.4.)

Сьогодні PHP використовується сотнями тисяч розробників. Декілька мільйонів сайтів написані на PHP, що складає більше 20% доменів Internet.

«PHP може усе», - заявляють його творці. В першу чергу PHP

використовується для створення скриптів, працюючих на стороні сервера, для цього його, власне, і придумали. PHP здатний вирішувати ті ж завдання, що і будь-які інші CGI – скрипти, у тому числі обробляти дані html –форм, динамічно генерувати html сторінки і тому подібне. Але є і інші області, де може використовуватися PHP . Всього виділяють три основні сфери застосування PHP.

Перша область, як вже говорилося, - це створення додатків ( скриптів ), які виконуються на стороні сервера. PHP найширше використовується саме для створення такого роду скриптів. Для того, щоб працювати таким чином, знадобиться PHP –парсер (тобто обробник php – скриптів ) і web –сервер для обробки скрипта, браузер для перегляду результатів роботи скрипта, ну, і, звичайно, який-небудь текстовий редактор для написання самого php –кода.

Друга область – це створення скриптів, що виконуються в командному рядку. Тобто за допомогою PHP можна створювати такі скрипти, які виконуватимуться, незалежно від web –сервера і браузеру, на конкретній машині. Для такої роботи знадобиться лише парсер PHP (в цьому випадку його називають інтерпретатором командного рядка (cli, command line interpreter)).

І остання область – це створення GUI –додатків (графічних інтерфейсів), що виконуються на стороні клієнта. В принципі це не самий кращий спосіб використати PHP, особливо для початківців, але якщо ви вже досконально вивчили PHP, то такі можливості мови можуть виявитися дуже корисні. Для застосування PHP в цій області знадобиться спеціальний інструмент – PHP – GTK, який є розширенням PHP.

Отже, сфера застосування PHP досить велика і різноманітна. Проте існує безліч інших мов програмування, здатних вирішувати схожі завдання. Чому варто вивчати PHP? Що це нам дає? По-перше, PHP дуже простий у вивченні. Досить ознайомитися лише з основними правилами синтаксису і принципами його роботи, і можна починати писати власні програми, причому братися за такі завдання, рішення яких на іншій мові вимагало б серйозної підготовки.

По-друге, PHP підтримується майже на усіх відомих платформах, майже в усіх операційних системах і на самих різних серверах. Це теж дуже важливо. Навряд чи комусь захочеться переходити, наприклад, від роботи під Windows до роботи під Linux або від сервера IIS до сервера Apache тільки для того, щоб вивчити ще одну мову програмування.

**Перша PHP –программа**

Розглянемо приклад.

```
<html> <head>
 <title>Приклад</title>
 </head> <body>
 <?php
 echo «<p>Привіт, я – скрипт PHP!</p>»;
 ?>
</body> </html>
```

Це простий HTML –файл, в який вбудований за допомогою спеціальних тегів код, написаний на мові PHP.

PHP –скрипти – це програми, які виконуються і обробляються сервером. Так що порівнювати його із скриптовими мовами типу JavaScript неможливо, тому що написані на них скрипти виконуються на машині клієнта. У чому відмінність скриптів, що виконуються на сервері і на клієнтові? Якщо скрипт обробляється сервером, клієнтові посилаються тільки результати роботи скрипта . Наприклад, якщо на сервері виконувався скрипт, подібний до приведеного вище, клієнт отримає згенеровану HTML –сторінку виду :

```
<html>
 <head>
 <title>Приклад</title>
 </head>
 <body>
 <p>Привіт, я – скрипт PHP!</p>
 </body>
</html>
```

В цьому випадку клієнт не знає, який код виконується. Можна навіть конфігурувати свій сервер так, щоб HTML –файли оброблялися процесором PHP, так що клієнти навіть не зможуть дізнатися, чи отримують вони звичайний HTML-файл або результат виконання скрипта. Якщо ж скрипт обробляється клієнтом (наприклад, це програма на мові JavaScript), то клієнт отримує сторінку, що містить код скрипта.

Ми відмічали вище, що PHP – скрипти вбудовуються в HTML –код. Виникає питання, яким чином? Є декілька способів. Один з них приведений в найпершому прикладі – за допомогою відкриваючого тега <?php і закриваючого тега ?> . Такого виду спеціальні теги дозволяють перемикатися між режимами HTML і PHP. Цей синтаксис найбільш прийнятний, оскільки дозволяє задіяти PHP в XML-сумісних програмах (наприклад, написаних на мові XHTML), але проте можна використати наступні альтернативні варіанти (команда echo «Some text» ; виводить на екран текст «Some text».) :

```
<? echo «Це проста інструкція для обробки PHP»; ?>
<script language=»php»>
 echo «Текст на сорінці»;  </script>
<% echo «Можна використати теги у стилі ASP «; %>
```

Перший з цих способів не завжди доступний. Щоб їм користуватися, треба включити короткі теги або за допомогою функції short_tags () для PHP 3, або включивши установку short_open_tag в конфігураційному файлі PHP, або скомпілювавши PHP з параметром –enable-short-tags. Навіть якщо це включено по замовчуванню в php.ini –dist, використання коротких тегів не рекомендується. Другий спосіб аналогічний вставці, наприклад, JavaScript – коду і використовує для цього відповідний html тег. Тому використати його можна завжди, але це робиться рідко із-за його громіздкості. Третій спосіб можна застосувати, тільки якщо теги в стилі ASP були включені, використовуючи конфігураційну установку asp_tags.

Коли PHP обробляє файл, він просто передає його текст, поки не зустріне один з перерахованих спеціальних тегів, який повідомляє його про необхідність почати інтерпретацію тексту як коду PHP. Потім він виконує увесь знайдений код до закриваючого тега, що говорить інтерпретатору, що далі знову йде просто текст. Цей механізм дозволяє впроваджувати PHP –код в HTML – усе за межами тегів PHP залишається незмінним, тоді як усередині інтерпретується як код. Помітимо також, що php –файл не схожий на CGI – скрипт. Php-файл не має бути здійснимим або ще яким-небудь чином поміченим.

Для того, щоб відправити php-файл на обробку серверу, треба в рядку браузеру набрати шлях до цього файлу на сервері. Скрипти php повинні розташовуватися там, де дозволений доступ через www, наприклад там же, де лежить домашня сторіночка. Якщо php –файл лежить на локальній машині, то його можна обробити за допомогою інтерпретатора командного рядка.

### 2.3 Загальний синтаксис PHP

Перше, що треба знати відносно синтаксису PHP, - це те, як він вбудовується в HTML-код, як інтерпретатор дізнається, що це код на мові PHP. У попередньому пункті лекції ми вже говорили про це. Повторюватися не будемо, відмітимо тільки, що в прикладах ми найчастіше використовуватимемо варіант <?php ?>, і іноді скорочений варіант <? ?>.

Програма на PHP (та і на будь-якій іншій мові програмування) – це набір команд (інструкцій). Обробникові програми (парсеру) необхідно якось

відрізняти одну команду від іншої. Для цього використовуються спеціальні символи – роздільники. У PHP інструкції розділяються так само, як і в Си або Perl, - кожен вираз закінчується крапкою з комою.

Закриваючий тег «?> « також має на увазі кінець інструкції, тому перед ним крапку з комою не ставлять. Наприклад, наступні фрагменти коду еквівалентні:

```
<?php
echo «Hello, world»!; // крапка з комою
 // у кінці команди
 // обов’язкова
?>
<?php
echo «Hello, world»! ?>
<!-крапка з комою
 опускається із-за «?>» 
```

Часто при написанні програм виникає необхідність робити які-небудь коментарі до коду, які ніяк не впливають на сам код, а тільки пояснюють його. Це важливо при створенні великих програм і у разі, якщо декілька чоловік працюють над однією програмою. За наявності коментарів в програмі в її коді розібратися набагато простіше. Крім того, якщо вирішувати задачу по частинах, недороблені частини рішення також зручно коментувати, щоб не забути про них надалі. У усіх мовах програмування передбачена можливість включати коментарі в код програми. PHP підтримує декілька видів коментарів : в стилі Си, C++ і оболонки Unix. Символи // і # означають початок однорядкових коментарів, /* і */ - відповідно початок і кінець багаторядкових коментарів.

```
<?php
echo «Мене звуть Василь»;
 // Це однорядковий коментар
 // у стилі C++
echo «Моє прізвище Петренко»;
/* Це багаторядковий коментар.
При виконанні програми усе, що
знаходиться тут (закоментовано)
буде ігноровано. */
echo «Я вивчаю PHP «;
 # Це коментар в стилі
 # оболонки Unix
?>
```

Важливим елементом кожної мови є змінні, константи і оператори, що застосовуються до цих змінних і констант. Розглянемо, як виділяються і обробляються ці елементи в PHP.

Змінна в PHP позначається знаком долара, за яким йде її ім'я. Наприклад:

```
$my_var
```

Ім'я змінної чутливе до регістра, тобто змінні $my_var і $My_var різні.

Імена змінних відповідають тим же правилам, що і інші найменування в PHP: правильне ім'я змінної повинне розпочинатися з букви або символу підкреслення з подальшими у будь-якій кількості буквами, цифрами або символами підкреслення.

У PHP 3 змінні завжди присвоювалися за значенням. Тобто коли ви присвоюєте вираз зміній, усі значення оригінального виразу копіюються в цю змінну. Це означає, наприклад, що після присвоєння однієї змінної значення інший, зміна однієї з них не впливає на значення іншої.

```
<?php
$first = ' Text ';    // Присвоюється $first
                      // значення ' Text '
$second = $first;     // Присвоюється змінній
                  // $second значення $first
?>
```

PHP 4, окрім цього, пропонує ще один спосіб присвоєння значень змінним: присвоєння по посиланню. Для того, щоб присвоїти значення змінної по посиланню, це значення повинне мати ім'я, тобто воно має бути представлене якою-небудь змінною. Щоб вказати, що значення однієї змінної присвоюється іншій змінній по посиланню, треба перед ім'ям першої змінної поставити знак амперсанд &.

Розглянемо той же приклад, що і вище, тільки присвоюватимемо значення змінної first змінної second по посиланню:

```
<?php
$first = 'Text'; //Присвоюємо $first
                 // значення 'Text'
$second = &$first;
/* Робимо посилання на $first через $second. Тепер
значення цих змінних завжди співпадатимуть */
```

```
?>
```

Для зберігання постійних величин, тобто таких величин, значення яких не міняється в ході виконання скрипта, використовуються константи. Такими величинами можуть бути математичні константи, паролі, шляхи до файлів і тому подібне. Основна відмінність константи від змінної полягає в тому, що їй не можна присвоїти значення більше одного разу і її значення не можна анулювати після її оголошення. Крім того, у константи немає приставки у вигляді знаку долара і її не можна визначити простим наданням значення. Як же тоді можна визначити константу? Для цього існує спеціальна функція define(). Її синтаксис такий:

```
define(«Ім'я_константи», «Значення_константи»,
 [Нечутливість_до_регістра])
```

По замовчуванню імена констант чутливі до регістра. Для кожної константи це можна змінити, вказавши в якості значення аргументу Нечутливість_до_регістра значення True. Існує угода, по якій імена констант завжди пишуться у верхньому регістрі.

Отримати значення константи можна, вказавши її ім'я. На відміну від змінних, не треба упереджати ім'я константи символом $. Крім того, для набуття значення константи можна використати функцію constant() з ім'ям константи в якості параметра.

```
<?php
// визначаємо константу PASSWORD
define(«PASSWORD», «qwerty»);
//визначаємо регістронезалежну
//константу PI зі значенням 3.14
define(«PI», «3.14», True);
// виведемо значення константи PASSWORD,
// тобто qwerty
echo (PASSWORD);
// теж виведе qwerty
echo constant(«PASSWORD»);
?>
```

Окрім констант, що оголошуються користувачем, про яких ми тільки що розповіли, в PHP існує ряд констант, визначуваних самим інтерпретатором. Наприклад, константа __FILE__ зберігає ім'я файлу програми (і шлях до нього), яка виконується в даний момент, __FUNCTION__ містить ім'я функції,

__CLASS__ - ім’я класу, PHP_VERSION – версія інтерпретатора PHP. Повний список обумовлених констант можна отримати, прочитавши керівництво по PHP.

### 2.3.1 Оператори в PHP

Оператори дозволяють виконувати різні дії зі змінними, константами і виразами. Ми ще не згадували про те, що таке вираз. Вираз можна визначити як усе, що завгодно, що має значення. Змінні і константи – це основні і найбільш прості форми виразів. Існує безліч операцій (і операторів, що відповідають їм ), які можна робити з виразами. В таблицях 2.1-2.6 детально розглянуто основні види операцій в PHP.

Таблиця 2.1. Арифметичні оператори

| Позначення | Назва | Приклад |
|---|---|---|
| + | Додавання | $a + $b |
| - | Віднімання | $a - $b |
| * | Множення | $a * $b |
| / | Ділення | $a / $b |
| % | Залишок від ділення | $a % $b |

Таблиця 2.2. Оператори роботи з рядками

| Позначення | Назва | Приклад |
|---|---|---|
| . | Конкатенація (додавання рядків ) | $c = $a . $b (це рядок, що складається з $a і $b ) |

Таблиця 2.3. Оператори присвоєння

| Позначення | Назва | Опис | Приклад |
|---|---|---|---|
| = | Присвоєння | Змінній зліва від оператора буде присвоєно значення, отримане в результаті виконання яких-небудь операцій або змінної / константи з правого боку | $a = ($b = 4) +5; ( $a дорівнюватиме 9, $b дорівнюватиме 4 ) |
| += | Додавання з присвоєнням | Скорочення. Додає до змінної число і потім привласнює їй отримане значення | $a += 5; (еквівалентно $a = $a + 5; ) |

| | | | |
|---|---|---|---|
| .= | Конкатенація з присвоєнням | Скорочено означає комбінацію операцій конкатенації і присвоєння (спочатку додається рядок, потім отриманий рядок записується в змінну ) | $b = «Привіт «; $b .= «усім»; (еквівалентно $b = $b . «усім»; ) В результаті: $b=»Привіт усім» |

Таблиця 2.4. Логічні оператори

| **Позначення** | Назва | **Опис** | **Приклад** |
|---|---|---|---|
| and | І | $a і $b істинні ( True ) | $a and $b |
| && | І | | $a && $b |
| or | АБО | Хоч би одна зі змінних $a або $b істинна (можливо, що і обоє) | $a or $b |
| \|\| | АБО | | $a \|\| $b |
| xor | Виключне АБО | Одна зі змінних істинна. Випадок, коли вони обоє істинні, виключається | $a xor $b |
| ! | Інверсія (NOT) | Якщо $a=True, то !$a=False і навпаки | ! $a |

Таблиця 2.5. Оператори порівняння

| **Позначення** | **Назва** | **Опис** | **Приклад** |
|---|---|---|---|
| == | Рівність | Значення змінних рівні | $a == $b |
| === | Еквівалентність | Рівні значення і типи змінних | $a === $b |
| != | Нерівність | Значення змінних не рівні | $a != $b |
| <> | Нерівність | | $a <> $b |
| !== | Нееквівалентність | Змінні не еквівалентні | $a !== $b |
| < | Менше | | $a < $b |
| > | Більше | | $a > $b |
| <= | Менше або рівно | | $a <= $b |
| >= | Більше або рівно | | $a >= $b |

Таблиця 2.6. Оператори інкременту і декременту

| **Позначення** | **Назва** | **Опис** | **Приклад** |
|---|---|---|---|
| ++$a | Пре-інкремент | Збільшує $a на одиницю і повертає $a | <? $a=4; echo «Має бути 4:» .$a++; echo «Має бути 6:» .++$a; |

| | | | |
|---|---|---|---|
| | | | ?> |
| $a++ | Пост-інкремент | Повертає $a, потім збільшує $a на одиницю | |
| --$a | Пре-декремент | Зменшує $a на одиницю і повертає $a | |
| $a-- | Пост-декремент | Повертає $a, потім зменшує $a на одиницю | |

### 2.3.2 Прості типи даних PHP

PHP підтримує вісім простих типів даних.

Чотири скалярні типи:

- boolean(логічний);
- integer(цілий);
- float(з плаваючою крапкою);
- string(строковий).

Два змішані типи:

- array(масив);
- object(об’ єкт).

І два спеціальні типи:

- resource(ресурс);
- NULL.

У PHP не прийнято явне оголошення типів змінних. Прийнятніше, щоб це робив сам інтерпретатор під час виконання програми залежно від контексту, в якому використовується змінна. Розглянемо по порядку усі перераховані типи даних.

Тип boolean (булевий або логічний тип).

Цей простий тип виражає істинність значення, тобто змінна цього типу може мати тільки два значення – істина TRUE або брехня FALSE .

Щоб визначити булевий тип, використовують ключове слово TRUE або FALSE. Обидві регістонезалежні.

```
<?php
$test = True;
?>
```

Логічні змінні використовуються в різних керівниках конструкціях (циклах, умовах і тому подібне, детальніше мова про них піде в одній з наступних лекцій). Мати логічний тип, тобто набувати тільки два значення, істину або брехню, можуть також і деякі оператори (наприклад, оператор рівності). Вони також використовуються в конструкціях, що управляють, для перевірки яких-небудь умов. Наприклад, в умовній конструкції перевіряється істинність значення оператора або змінної і залежно від результату перевірки виконуються ті або інші дії. Тут умова може бути істинна або неправдива, що якраз і відбиває змінна і оператор логічного типу.

```
<?php
// Оператор '==' перевіряє рівність
// і повертає булеве значення
if ($know == False){ // якщо $know має
                     //значення false
echo «Вивчай PHP»!;
}
?>
```

Тип integer (цілі) задає число з множини цілих чисел Z ={…, - 2, - 1, 0, 1, 2, …}. Цілі можуть бути вказані в десятковій, шістнадцятиричній або вісімковій системі числення, за бажанням з попереднім знаком « – « або « + «.

Якщо ви використовуєте вісімкову систему числення, ви повинні упередити число 0 (нулем), для використання шістнадцятиричної системи треба поставити перед числом 0x.

```
<?php
# десяткове число
$a = 1234;
# від'ємне число
$a = - 123;
# вісімкове число (екв. 83 у десятковій системі)
$a = 0123;
# шістнадцяткове число (екв. 26 у десятк.системі)
$a = 0x1A;
?>
```

Розмір цілого залежить від платформи, хоча, як правило, максимальне значення близько двох мільярдів (це 32-бітове знакове). Беззнакові цілі PHP не підтримує.

Якщо ви визначите число, що перевищує межі цілого типу, воно буде

інтерпретовано як число з плаваючою крапкою. Також якщо ви використовуєте оператор, результатом роботи якого буде число, що перевищує межі цілого, замість нього буде повернено число з плаваючою крапкою.

У PHP не існує оператора ділення цілих. Результатом ½ буде число з плаваючою крапкою 0.5. Ви можете привести значення до цілого, що завжди округлює його в меншу сторону, або використати функцію round(), що округлює значення за стандартними правилами. Для перетворення змінної до конкретного типу треба перед змінною вказати в дужках потрібний тип. Наприклад, для перетворення змінної $a=0.5 до цілого типу необхідно написати(0.5) або (integer) $a або використати скорочений запис(0.5). Можливість явного приведення типів за таким принципом існує для усіх типів даних (звичайно, не завжди значення одного типу можна перевести в інший тип ). Ми не поглиблюватимемося в усі тонкощі приведення типів, оскільки PHP робить це автоматично залежно від контексту.

Тип float (числа з плаваючою крапкою) (вони ж числа подвійної точності або дійсні числа) можуть бути визначені за допомогою будь-якого з наступних синтаксисів :

```
<?php
$a = 1.234;
$b = 1.2e3;
$c = 7E-10;
?>
```

Розмір числа з плаваючою крапкою залежить від платформи, хоча максимум, як правило, ~1.8e308 з точністю близько 14 десяткових цифр.

Тип string (рядки)

Рядок – це набір символів. У PHP символ – це те ж саме, що байт, це означає, що існує рівно 256 різних символів. Це також означає, що PHP не має вбудованої підтримки Unicode. У PHP практично не існує обмежень на розмір рядків, тому немає абсолютно ніяких причин турбуватися про їх довжину.

Рядок в PHP може бути визначений трьома різними способами:

- за допомогою одинарних лапок ;
- за допомогою подвійних лапок ;
- heredoc – синтаксисом.

Простий спосіб визначити рядок – це взяти її в одинарних лапок « ‘ «. Щоб використати одинарну лапку усередині рядка, як і у багатьох інших

мовах, перед нею необхідно поставити символ оберненої косої риски « \ «, т. е. екранувати її. Якщо обернена коса риска повинна йти перед одинарною лапкою або бути у кінці рядка, необхідно продублювати її « \\’ «.

Якщо усередині рядка, поміщеного в одинарні лапки, зворотний слеш « \ « зустрічається перед будь-яким іншим символом (відмінним від « \ « і « ‘ « ), то він розглядається як звичайний символ і виводиться, як і усі інші. Тому обернену косу риску необхідно екранувати, тільки якщо вона знаходиться у кінці рядка, перед закриваючою лапкою.

У PHP існує ряд комбінацій символів, що розпочинаються з символу оберненої косої риски. Їх називають послідовностями, що управляють, і вони мають спеціальні значення, про які ми розповімо трохи пізніше. Так от, на відміну від двох інших синтаксисів, змінні і такі, що управляють послідовності для спеціальних символів, що зустрічаються в рядках, поміщених в одинарні лапки, не обробляються .

```
<?php
// Виведе: Щоб вивести ‘ потрібно
// перед нею поставити \
echo ‘Щоб вивести \’ потрібно перед
нею поставити \\’;?>
```

Якщо рядок поміщений в подвійні лапки « «, PHP розпізнає більшу кількість послідовностей, що управляють, для спеціальних символів. Деякі з них приведені в таблиці 2.7.

Таблиця 2.7. Керуючі послідовності

| Послідовність | Значення |
|---|---|
| \n | Новий рядок ( LF або 0x0A (10) в ASCII) |
| \r | Повернення каретки ( CR або 0x0D (13) в ASCII) |
| \t | Горизонтальна табуляція ( HT або 0x09 (9) в ASCII) |
| \\ | Обернена коса риска |
| \$ | Знак долара |
| \» | Подвійна лапка |

Повторюємо, якщо ви захочете екранувати будь-який інший символ, обернена коса риска також буде надрукована!

Найважливішою властивістю рядків в подвійних лапках є обробка змінних.

```
‹ ?php
```

```
$str = «Іван»;
// виведе: Привіт, Іван!
echo «Привіт, $str»!;
?›
```

В даному випадку замість підрядка $str підставляється значення змінної $str. Відбувається це за наступною схемою. Підрядок вважатиметься змінній, якщо вона утворює правильне ім’я змінної. Наприклад, в рядку Hello, $strs! змінною вважатиметься $strs, а не $str як в попередньому прикладі. Якщо все-таки вимагається розпізнати саме змінну $str в цьому рядку, то потрібно застосувати фігурні дужки.

```
‹?php
$str = ‘Ivan’;
// виведе: Hello, Ivans!
echo «Hello, {$str}s»!;
?›
```

У тому разі якщо вам знадобиться вивести знак долара (наприклад, для виведення імені змінної), можна використати одинарні лапки або екранувати його.

```
‹?php
$str = «Hello»;
// виведе: Змінна має ім’я $str
echo ‘Змінна має ім’я $str’;
echo ««br –«;
// виїде: Змінна має ім’я $str
echo ‘Змінна має ім’я \$str’;
?›
```

У обох випадках змінна в рядках не визначається.

Інший спосіб визначення рядків – це використання heredoc –синтаксиса. В цьому випадку рядок повинен розпочинатися з символу <<<, після якого йде ідентифікатор. Закінчується рядок цим же ідентифікатором. Закриваючий ідентифікатор повинен починатися в першому стовпці рядка. Крім того, ідентифікатор повинен відповідати тим же правилам іменування, що і усі інші мітки в PHP: містити тільки буквено-цифрові символи і знак підкреслення і розпочинатися не з цифри або знаку підкреслення.

Heredoc –текст поводиться так само, як і рядок в подвійних лапках, при цьому їх не маючи. Це означає, що вам немає необхідності екранувати лапки

в heredoc, але ви як і раніше можете використати перелічені вище послідовності, що управляють. Змінні усередині heredoc теж обробляються.

```
<?php
$str = <<<EOD
Приклад рядка, що охоплює декілька
рядків, з використанням heredoc – синтаксису
EOD;
// Тут ідентифікатор – EOD. Нижче
// ідентифікатор EOD
$name = 'Василь';
echo <<<EOD
Мене звуть «$name». EOD;
// виведе: Мене звуть «Василь».
?>
```

Зауваження: Підтримка heredoc була додана в PHP 4.

### 2.3.3 Тип array (масив)

Масив в PHP є впорядкованою картою – типом, який перетворить значення в ключі. Цей тип оптимізований в декількох напрямах, тому ви можете використати його як власне масив, список (вектор), хеш-таблицю (що є реалізацією карти), стек, чергу і так далі. Оскільки ви можете мати в якості значення інший масив PHP, можна також легко емулювати дерева.

Визначити масив можна за допомогою конструкції array() або безпосередньо задаючи значення його елементам.

Визначення за допомогою array()

```
array (key => value
 key1 => value1, … )
```

Мовна конструкція array () приймає як параметри пари ключ => значення, розділені комами. Символ => встановлює відповідність між значенням і його ключем. Ключ може бути як цілим числом, так і рядком, а значення може бути будь-якого наявного в PHP типу. Числовий ключ масиву часто називають індексом. Індексування масиву в PHP розпочинається з нуля. Значення елементу масиву можна отримати, вказавши після імені масиву в квадратних дужках ключ шуканого елементу. Якщо ключ масиву є стандартним записом цілого числа, то він розглядається як число, інакше – як рядок. Тому запис $a[«1»] рівносильна запису $a[1], так само як і a[«-1»] рівносильно $a[- 1].

```
<?php
$books = array («php» =>»PHP guide», 12 => true);
echo $books[«php»];
//виведе «PHP guide»
echo $books[12];
//виведе 1
?>
```

Якщо для елементу ключ не заданий, то в якості ключа береться максимальний числовий ключ, збільшений на одиницю. Якщо вказати ключ, якому вже було присвоєно якесь значення, то воно буде перезаписано. Починаючи з PHP 4.3.0, якщо максимальний ключ – від'ємне число, то наступним ключем масиву буде нуль ( 0 ).

Якщо використати в якості ключа TRUE або FALSE, то його значення переводиться відповідно в одиницю і нуль типу integer. Якщо використати NULL, то замість ключа отримаємо порожній рядок. Можна використати і сам порожній рядок в якості ключа, при цьому її потрібно брати в лапки. Так що це не те ж саме, що використання порожніх квадратних дужок. Не можна використати в якості ключа масиви і об'єкти.

Створити масив можна, просто записуючи в нього значення. Як ми вже говорили, значення елементу масиву можна отримати за допомогою квадратних дужок, усередині яких треба вказати його ключ, наприклад, $book[«php»]. Якщо вказати новий ключ і нове значення, наприклад, $book[«new_key»]=»new_value», то в масив додасться новий елемент. Якщо ми не вкажемо ключ, а тільки присвоїмо значення $book[]=»new_value», то новий елемент масиву матиме числовий ключ, на одиницю більший максимального існуючого. Якщо масив, в який ми додаємо значення, ще не існує, то він буде створений.

```
<?
$books[«key»]= value; // додали в масив $books
 // значення value з ключем key
$books[] = value1; /* додали в масив значення
                      value1 з ключем 13,
                      оскільки максимальний ключ у
 нас був 12 */
?>
```

Для того, щоб змінити конкретний елемент масиву, треба просто присвоїти йому з його ключем нове значення. Змінити ключ елементу не можна, можна тільки видалити елемент (пару ключ / значення ) і додати нову.

Щоб видалити елемент масиву, треба використати функцію unset ().

```
<?php
$books = array («php» => «PHP users guide»,
 12 => true);
unset($books[12]); //Видаляє елемент з
                   // ключем 12 з масиву
unset ($books); // видаляє масив повністю
?>
```

Помітимо, що, коли використовуються порожні квадратні дужки, максимальний числовий ключ шукається серед ключів, існуючих в масиві з моменту останнього переиндексирования. Переіндексувати масив можна за допомогою функції array_values ().

### 2.3.4 Тип object (об'єкти)

Об'єкти – тип даних, що прийшов з об'єктно-орієнтованого програмування (ООП). Згідно з принципами ООП, клас – це набір об'єктів, що мають певні властивості і методи роботи з ним, а об'єкт відповідно – екземпляр класу. Наприклад, програмісти – це клас людей, які пишуть програми, вивчають комп'ютерну літературу і, крім того, як усі люди, мають ім'я і прізвище. Тепер, якщо узяти одного конкретного програміста, Васю Іванова, то можна сказати, що він є об'єктом класу програмістів, має ті ж властивості, що і інші програмісти, теж має ім'я, пише програми і тому подібне

У PHP для доступу до методів об' єкту використовується оператор ->. Для ініціалізації об' єкту використовується вираз new, що створює в змінній екземпляр об' єкту.

```
<?php
class Person //створюємо клас людей
{    // метод, який навчає людину PHP
function know_php()
 {
 echo «Тепер я знаю PHP»;
 }
}
$bob = new Person; // створюємо об' єкт
 // класу людина
$bob -> know_php(); // навчаємо його PHP
?>
```

### 2.3.5 Інші типи даних в PHP

Тип resource Ресурс – це спеціальна змінна, що містить посилання на зовнішній ресурс (наприклад, з'єднання з базою даних). Ресурси створюються і використовуються спеціальними функціями (наприклад, mysql_connect(), pdf_new() і тому подібне).

Тип Null. Спеціальне значення NULL говорить про те, що змінна не має значення.

Змінна вважається NULL, якщо:

- їй була присвоєна константа NULL ( $var = NULL );
- їй ще не було присвоєно яке-небудь значення;
- вона була видалена за допомогою unset ().

Існує тільки одне значення типу NULL – регістронезалежне ключове слово NULL.

Задача. Потрібно скласти листа різним людям з приводу різних подій. Спробуємо використати для вирішення цього завдання вивчені засоби – змінні, оператори, константи, рядки і масиви. Залежно від одержувача змінюється подія і звернення, вказані в листі, тому природно винести ці величини в змінні. Більше того, оскільки подій і людей багато, зручно використати змінні типу масив. Підпис в листі залишається постійним завжди, тому логічно задати її як константу. Щоб не писати занадто довгі і громіздкі рядки, використовуваний оператор конкатенації.

```
<?
// нехай наш підпис буде константою
define(«SIGN», «З повагою, Василь»);
// задамо масиви людей і подій
$names = array(«Іван Іванович», «Петро Петренкоич», «Семен Семенович»);
$events = array(«f» => «день відкритих дверей», «o» => «відкриття виставки»
 «p» => «бал випускників»);
// складемо текст запрошення
$str = «Шановний (а), $names[0]».;
$str .= «<br> Запрошуємо Вас на «. $events[“f”];
$str .= “<br>” . SIGN;
echo $str; // виведемо текст на экран
?>
```

### 2.4 Основні операції в PHP

#### 2.4.1 Оператори розгалудження і вибору

Оператор if – це один з найважливіших операторів багатьох мов, включаючи PHP. Він дозволяє виконувати фрагменти коду залежно від умови. Структуру оператора if можна представити таким чином:

```
if (вираз) блок_виконання
```

Тут вираз є будь-який правильний PHP-вираз (тобто усе, що має значення). В процесі обробки скрипта вираз перетвориться до логічного типу. Якщо в результаті перетворення значення виразу істинне ( True ), то виконується блок_виконання. Інакше блок_виконання ігнорується. Якщо блок_виконання містить декілька команд, то він має бути поміщений у фігурні дужки { }.

Правила перетворення виразу до логічного типу:

У FALSE перетворяться наступні значення:

- логічне False;
- цілий нуль ( 0 );
- дійсний нуль ( 0.0 );
- порожній рядок і рядок «0» ;
- масив без елементів;
- об’єкт без змінних (детально про об’єкти буде розказано в одній з наступних лекцій);
- спеціальний тип NULL ;

Усі інші значення перетворяться в TRUE.

```
<?
$names = array(«Іван», «Петро», «Семен»);
if ($names[0]==»Іван»){
 echo «Привіт, Ваня»!;
 $num = 1;
 $account = 2000;
}
if ($num) echo «Іван перший в списку»!;
$bax = 30;
if ($account > 100*$bax+3)
 echo «Цей рядок не з’явиться
 на екрані, оскільки умова не виконана»;
?>
```

Ми розглянули тільки одну, основну частину оператора if . Існує декілька розширень цього оператора. Оператор else розширює if на випадок,

якщо що перевіряється в if вираз є невірним, і дозволяє виконати які-небудь дії за таких умов.

Структуру оператора if, розширеного за допомогою оператора else, можна представити таким чином:

```
if (вираз) блок_виконання
else блок_виконання1
```

Цю конструкцію if…else можна інтерпретувати приблизно так: якщо виконана умова (тобто вираз=true ), то виконуємо дії з блоку_виконання, інакше – дії з блоку_виконання1. Використати оператор else не обов'язково.

Подивимося, як можна змінити попередній приклад, враховуючи необхідність здійснення дій і у разі невиконання умови.

```
<?
$names = array(«Іван», «Петро», «Семен»);
if ($names[0]==»Іван»){
 echo «Привіт, Ваня»!;
 $num = 1;
 $account = 2000;
} else {
 echo «Привіт, $names[0].
А ми чекали Ваню😉«;
}
if ($num) echo «Іван перший в списку»!;
else echo «Іван НЕ перший в списку»?!;
$bax = 30;
if ($account > 100*$bax+3)
 echo «Цей рядок не з'явиться на екрані
 оскільки умова не виконана»;
 else echo «Зате з'явиться цей рядок»!;
?>
```

Ще один спосіб розширення умовного оператора if – використання оператора elseif . elseif – це комбінація else і if . Як і else, він розширює if для виконання різних дій у тому випадку, якщо умова, що перевіряється в if, невірна. Але на відміну від else, альтернативні дії будуть виконані, тільки якщо elseif-умова є вірна. Структуру оператора if, розширеного за допомогою операторів else і elseif, можна представити таким чином:

```
if (вираз) блок_виконання
elseif(вираз1) блок_виконання1
```

```
…
else блок_виконанняN
```

Операторів elseif може бути відразу декілька в одному if –блоці. Elseif – твердження буде виконане, тільки якщо попередня if-умова є False, усі передуючі elseif –умови являються False, а дана elseif –умова – True.

```
<?
$names = array(«Іван», «Петро», «Семен»);
if ($names[0]==»Іван»){
 echo «Привіт, Ваня»!;
}elseif ($names[0] == «Петро»){
 echo «Привіт, Петя»!;
}elseif ($names[0] == «Семен»){
 echo «Привіт, Сеня»!;
}else {
 echo «Привіт, $names[0]. А ти хто такий»?;
}
?>
```

Альтернативний синтаксис. PHP пропонує альтернативний синтаксис для деяких своїх структур, що управляють, а саме для if, while, for, foreach і switch . У кожному випадку відкриваючу дужку треба замінити на двокрапку ( 💻, а що закриває – на endif;, endwhile; і так далі відповідно.

Наприклад, синтаксис оператора if можна записати таким чином:

if(вираз) : блок_виконання endif;

Сенс залишається тим же : якщо умова, записана в круглих дужках оператора if, виявилася істиною, виконуватиметься увесь код, від двокрапки « :» до команди endif;. Використання такого синтаксису корисне при вбудовуванні php в html –код.

Ще одна конструкція, що дозволяє перевіряти умову і виконувати залежно від цього різні дії, - це switch . На російську мову назву цього оператора можна перевести як «перемикач». І сенс у нього саме такий. Залежно від того, яке значення має змінна, він перемикається між різними блоками дії. switch дуже схожий на оператор if…elseif…else або набір операторів if . Структуру switch можна записати таким чином:

```
switch (вираз або змінна){
case значення1:
 блок_дій1
break;
```

```
case значення2:
 блок_дій2
break;
…
default:
 блок_дій_по_замовчуванню
}
```

На відміну від if, тут значення вираз не приводиться до логічного типу, а просто порівнюється зі значеннями, перерахованими після ключових слів case ( значення1, значення2 і так далі). Якщо значення виразу співпало з якимсь варіантом, то виконується відповідний блок_дій – від двокрапки після значення, що співпало, до кінця switch або до першого оператора break, якщо такий знайдеться. Якщо значення виразу не співпало ні з одним з варіантів, то виконуються дії по замовчуванню (блок_дій_по_замовчуванню), що знаходяться після ключового слова default. Вираз в switch обчислюється тільки один раз, а в операторі elseif – кожного разу, тому, якщо вираз досить складний, то switch працює швидше.

```
<?
$names = array(«Іван», «Петро», «Семен»);
switch($names[0]){
case «Іван»:
 echo «Привіт, Ваня»!;
break;
case «Петро»:
 echo «Привіт, Петя»!;
break;
case «Семен»:
 echo «Привіт, Сеня»!;
break;
default:
 echo «Привіт, $names[0].
А як Вас звуть»?;
}
?>
```

Якщо в даному прикладі опустити оператор break, наприклад, в case «Петро»:, то, якщо змінна виявиться рівному рядку «Петро», після виводу на екран повідомлення «Привіт, Петя»! програма піде далі і виведе також повідомлення «Привіт, Сеня»! і тільки потім, зустрівши break, продовжить своє виконання за межами switch .

Для конструкції switch, як і для if, можливий альтернативний синтаксис, де відкриваюча switch фігурна дужка замінюється двокрапкою, а що закриває – endswitch; відповідно.

### 2.4.2 Цикли в PHP

У PHP існує декілька конструкцій, що дозволяють виконувати дії, що повторюються, залежно від умови. Це цикли while, do…while, foreach і for . Розглянемо їх детальніше.

Оператор while. Структура:

```
while (вираз){ блок_виконання }
або
while (вираз) : блок_виконання endwhile;
```

while – простий цикл. Він пропонує PHP виконувати команди блоку_виконання до тих пір, поки вираз обчислюється як True (тут, як і в if, відбувається приведення виразу до логічного типу). Значення виразу перевіряється кожного разу на початку циклу, так що, навіть якщо його значення змінилося в процесі виконання блоку_виконання, цикл не буде зупинений до кінця ітерації (тобто доки усі команди блоку_виконання не будуть виконані).

```
<?
//ця програма надрукує усі парні цифри
 $i = 1;
 while ($i < 10){
 if ($i % 2 == 0) print $i;
 // друкуємо цифру, якщо вона парна
 $i++; // і збільшуємо $i на одиницю
 }
?>
```

Цикли do.while дуже схожі на цикли while, з тією лише різницею, що істинність виразу перевіряється у кінці циклу, а не на початку. Завдяки цьому блок_виконання циклу do…while гарантовано виконується хоч би один раз.

Структура:

```
do {блок_виконання} while (вираз);
<?
// ця програма надрукує число 12, незважаючи на те
// що умова циклу не виконана
```

```
$i = 12;
 do{
 if ($i % 2 == 0) print $i;
 // якщо число парне, то друкуємо його
 $i++;
 // збільшуємо число на одиницю
 }while ($i<10);
?>
```

Оператор for. Це найскладніші цикли в PHP. Вони нагадують відповідні цикли мови C. Структура оператора:

```
for (вираз1; вираз2; вираз3)
    { блок_виконання }
або
for (вираз1; вираз2; вираз3):
    блок_виконання
endfor;
```

Тут, як ми бачимо, умова складається відразу з трьох виразів. Перший вираз вираз1 обчислюється безумовно один раз на початку циклу. На початку кожної ітерації обчислюється вираз2. Якщо він являється True, то цикл триває і виконуються усі команди блоку_виконання. Якщо вираз2 обчислюється як False, то виконання циклу зупиняється. У кінці кожної ітерації (тобто після виконання усіх команд блоку_виконання ) обчислюється вираз3.

Кожен з виразів 1, 2, 3 може бути порожнім. Якщо вираз2 є порожнім, то це означає, що цикл повинен виконуватися невизначений час (в цьому випадку PHP вважає цей вираз завжди істинним). Це не так безглуздо, як здається, адже цикл можна зупиняти, використовуючи оператор break .

Наприклад, усі парні цифри можна вивести з використанням циклу for таким чином:

```
<?php
for ($i=0; $i<10; $i++){
 if ($i % 2 == 0) print $i;
 // друкуємо парні числа
}
?>
```

Якщо опустити другий вираз (умова $i<10 ), то таке ж завдання можна вирішити, зупиняючи цикл оператором break .

```
<?php
for ($i=0; ; $i++){
 if ($i>=10) break;
 // якщо $i більше або рівне 10,
 // то припиняємо роботу циклу
 if ($i % 2 == 0) print $i;
 // якщо число парне,
 // то друкуємо його
}
?>
```

Можна опустити усі три вирази. В цьому випадку просто не буде задано початкове значення лічильника $i і воно не змінюватиметься кожного разу у кінці циклу. Усі ці дії можна записати у вигляді окремих команд або у блоці_виконання, або перед циклом:

```
<?php
$i=2; // задаємо початкове значення лічильника
for ( ; ; ){
 if ($i>=10) break;
 // якщо $i більше або рівне 10,
 // то припиняємо роботу циклу
 if ($i % 2 == 0) print $i;
 // якщо число парне,
 // то друкуємо його
$i++;}// збільшуємо лічильник на одиницю
?>
```

У третьому виразі конструкції for можна записувати через кому відразу декілька простих команд. Наприклад, якщо ми хочемо просто вивести усі цифри, то програму можна записати зовсім просто:

```
<?php
for ($i=0; $i<10; print $i, $i++)
/* Якщо блок_виконання не містить команд
чи містить тільки одну команду,
фігурні дужки, в які він поміщений,
можна опускати*/
?>
```

Оператор foreach – це ще одна корисна конструкція. Вона з’явилася тільки в PHP4 і призначена виключно для роботи з масивами.

Синтаксис:

```
foreach ($array as $value){блок_виконання}
або
foreach ($array as $key => $value)
 {
    блок_виконання
    }
```

У першому випадку формується цикл по усіх елементах масиву, заданого змінній $array. На кожному кроці циклу значення поточного елементу масиву записується в змінну $value, і внутрішній лічильник масиву пересувається на одиницю (так що на наступному кроці буде видний наступний елемент масиву). Усередині блоку_виконання значення поточного елементу масиву може бути отримане за допомогою змінної $value. Виконання блоку_виконання відбувається стільки разів, скільки елементів в масиві $array.

Друга форма запису на додаток до переліченого вище на кожному кроці циклу записує ключ поточного елементу масиву в змінну $key, яку теж можна використати у блоці_виконання.

Коли foreach починає виконання, внутрішній вказівник масиву автоматично встановлюється на перший елемент.

```
<?php
$names = array(«Іван», «Петро», «Семен»);
foreach ($names as $val){
 echo «Привіт, $val <br>»;
 // виведе усім вітання
}
foreach ($names as $k => $val){
 // окрім вітання,
 // виведемо номери в списку, тобто ключі
 echo «Привіт, $val !
 Ти в списку під номером $k <br>»;
}
?>
```

### 2.4.3 Оператори передачі керування

Іноді вимагається негайно завершити роботу циклу або окремої його ітерації. Для цього використовують оператори break і continue .

Оператор break закінчує виконання поточного циклу, будь то for, foreach, while, do…while або switch. break може використовуватися з числовим аргументом, який говорить, роботу скількох керівників структур, що містять його, треба завершити.

```
<?php
$i=1;
while ($i){
 $n = rand(1,10);
 // генеруємо довільне число від 1 до 10
 echo «$i:$n «;
 // виводимо номер ітерації і згенероване число
 if ($n==5) break;
    /* Якщо було згенеровано число 5, то припиняємо
    роботу циклу. В цьому випадку усе, що
    знаходиться після цього рядка усередині циклу,
    не буде виконано */
 echo «Цикл працює <br>»;
 $i++;
}
echo «<br>Число ітерацій циклу $i «;
?>
```

Результатом роботи цього скрипта буде приблизно наступне:

```
1:7 Цикл працює
2:2 Цикл працює
3:5
Число ітерацій циклу 3
```

Якщо після оператора break вказати число, то урветься саме така кількість тих, що містять цей оператор циклів. У наведеному вище прикладі це неактуально, оскільки вкладених циклів немає. Трохи змінимо наш скрипт:

```
<?php
$i=1;
while ($i){
 $n = rand(1,10);
 // генеруємо довільне число від 1 до 10
 switch ($n){
 case 5:
 echo «<font color=blue>
 Вихід з switch (n=$n)</font>»;
 break 1;
// припиняємо роботу switch
 // (першого break циклу, що містить)
 case 10:
 echo «<font color=red>
 Вихід з switch і
    while (n=$n)</font>»;
```

```
 break 2;
// припиняємо роботу switch і while
 // (двох break циклів, що містять)
 default:
 echo «switch працює (n=$n) «;
}
echo « while працює – крок $i <br>»;
$i++;
}
echo «<br>Число ітерацій циклу $i «;
?>
```

Іноді треба не повністю припинити роботу циклу, а тільки почати його нову ітерацію. Оператор continue дозволяє пропустити подальші інструкції з блоку_виконання будь-якого циклу і продовжити виконання з нового круга. continue можна використати з числовим аргументом, який вказує, скільки його управляють конструкцій, що містять, повинні завершити роботу.

Замінимий в прикладі попереднього параграфа оператор break на continue . Крім того, обмежимо число кроків циклу трьома.

```
<?php
$i=1;
while ($i<=4){
 $n = rand(1,10);
// генеруємо довільне число від 1 до 10
 echo «$i:$n «;
// виводимо номер ітерації і згенероване число
 if ($n==5){
 echo «Нова ітерація <br>»;
 continue;
        /* Якщо було згенеровано число 5,
        те починаємо нову ітерацію циклу,
        $i не збільшується */
 }
 echo «Цикл працює <br>»;
 $i++;
}
--$i;
echo «<br>Число ітерацій циклу $i «;
?>
```

Результатом роботи цього скрипта буде

```
1:10
```

```
Цикл працює
2:5 Нова ітерація
2:1 Цикл працює
3:1 Цикл працює
4:1 Цикл працює
Число ітерацій циклу 4
```

Помітимо, що після виконання оператора continue робота циклу не закінчується. У прикладі лічильник циклу не міняється у разі отримання числа 5, оскільки він знаходиться після оператора continue . Фактично за допомогою continue ми намагаємося уникнути ситуації, коли буде згенеровано число 5. Тому можна було просто написати, замінивши оператор continue на перевірку істинності вираз :

```
<?php
$i=1;
while ($i<4){
 $n = rand(1,10);
// генеруємо довільне число від 1 до 10
 if ($n!==5){
 echo «$i:$n <br>»;
// виводимо номер ітерації і згенероване число
 $i++;
 }
}
?>
```

У PHP існує одна особливість використання оператора continue – в конструкціях switch він працює так само, як і break . Якщо switch знаходиться усередині циклу і треба почати нову ітерацію циклу, слід використати continue 2.

### 2.4.4 Оператори під’єднання

Оператор include дозволяє включати код, що міститься у вказаному файлі, і виконувати його стільки разів, скільки програма зустрічає цей оператор. Включення може робитися будь-яким з перерахованих способів :

```
include ‘ім’я_файлу’;
include $file_name;
include («ім’я_файлу»);
```

Приклад. Нехай у файлі params.inc у нас зберігається набір якихось

параметрів і функцій. Кожного разу, коли нам треба буде використати ці параметри (функції), ми вставлятимемо в текст нашої основної програми команду include ‘params.inc’.

Файл params.inc:

```
<?php
$user = «Василь»;
$today = date(«d.m.y»);
/* функція date() повертає дату і час
 (тут – дату у форматі день.місяць.рік) */
?>
```

Файл include.php:

```
<?php
include («params.inc»);
/* змінні $user і $today задані у файлі params.inc.
Тут ми теж можемо ними користуватися
завдяки команді include(«params.inc») */
echo «Привіт, $user!<br>»;
 // виведе «Привіт, Василь»!
echo «Сьогодні $today»;
 // виведе, наприклад, «Сьогодні 7.07.05»
?>
```

Помітимо, що використання оператора include еквівалентно простій вставці змістовної частини файлу params.inc в код програми include.php. Можливо, тоді можна було в params.inc записати простий текст без всяких тегів, що вказують на те, що це php –код? Не можна! Річ у тому, що у момент вставки файлу відбувається перемикання з режиму обробки PHP в режим HTML. Тому код усередині файлу, що включається, який треба обробити як PHP-скрипт, має бути поміщений у відповідні теги.

Пошук файлу для вставки відбувається за наступними правилами.

Спочатку ведеться пошук файлу в include_path відносно поточної робочої директорії.

Якщо файл не знайдений, то пошук здійснюється в include_path відносно директорії поточного скрипта.

Параметр include_path, визначуваний у файлі налаштувань PHP, задає імена директорій, в яких треба шукати файли, що включаються.

Наприклад, ваш include_path це . (тобто поточна директорія) поточна робоча директорія це /www/. В основний файл include.php ви включаєте файл

my_dir/a.php, який у свою чергу включає b.php. Тоді парсер насамперед шукає файл b.php в директорії /www/, і якщо такого немає, то в директорії /www/my dir/.

Якщо файл включений за допомогою include, то код, що міститься в ньому, наслідує зону видимості змінних рядка, де з'явився include . Будь-які змінні викликаного файлу будуть доступні в файлі з цього рядка і далі. Відповідно, якщо include з'являється усередині функції викликаю чого файлу, то код, що міститься у файлі, що викликається, поводитиметься так, як ніби він був визначений усередині функції. Таким чином, він успадкує зону видимості цієї функції. Хоча ми і не знайомилися ще з поняттям функції, все ж приводимо тут ці відомості з розрахунку на інтуїтивне його розуміння.

Приклад. Нехай файл для вставки params.inc залишиться таким же, а include.php буде наступним:

```
<?php
function Footer(){ // оголош.функцію з ім'ям Footer
include («params.inc»);
/* включаємо файл params.inc. Тепер його
змінними можна користуватися але тільки
усередині функції */
$str = «Сьогодні: $today <br>»;
$str .= «<a
href='mailto:help@intuit.ua'>Сторінку створив
     $user</a>»;
 echo «$str»;
}
Footer();
 // викликаємо функцію Footer(). Отримаємо:
    //Сьогодні: 08.07.05
    //Сторінку створив Василь

echo «$user, $today»;
 // виведе кому, оскільки
 // ці змінні видно тільки
// усередині функції
?>
```

Окрім локальних файлів, за допомогою include можна включати і зовнішні файли, вказуючи їх url-адресу. Ця можливість контролюється директивою url_fopen_wrappers у файлі налаштувань PHP і по замовчуванню, як правило, включена. Але у версіях PHP для Windows до PHP 4.3.0 ця можливість не підтримується зовсім, незалежно від url_fopen_wrappers.

include() – це спеціальна мовна конструкція, тому при використанні усередині умовних блоків її треба брати у фігурні дужки.

```
<?php
/* Це невірний запис. Отримаємо помилку.
 Ми ж вставляємо не одну команду, а декілька,
вони тільки записані у іншому файлі */
if ($condition) include(«first.php»);
else include(«second.php»);
// А ось так правильно.
if ($condition){ include(«first.php»); }
else { include(«second.php»); }
?>
```

При використанні include можливі два види помилок – помилка вставки (наприклад, не можна знайти вказаний файл, невірно написана сама команда вставки і тому подібне) або помилка виконання (якщо помилка міститься у файлі, що вставляється). У будь-якому випадку при помилці в команді include виконання скрипта не завершується.

Оператор require діє приблизно так само, як і #include в C++. Усе, що ми говорили про include, лише за деякими виключеннями, справедливо і для require . require також дозволяє включати в програму і виконувати який-небудь файл. Основна відмінність require і include полягає в тому, як вони реагують на виникнення помилки. Як вже говорилося, include видає попередження, і робота скрипта триває. Помилка в require викликає фатальну помилку роботи скрипта і припиняє його виконання.

Умовні оператори на require() не впливають. Хоча, якщо рядок, в якому з’являється цей оператор, не виконується, то жоден рядок коду з файлу, що вставляється, теж не виконується. Цикли також не впливають на require(). Хоча код, що міститься у файлі, що вставляється, є об’єктом циклу, але вставка сама по собі відбувається тільки одного разу.

**Контрольні питання до теми 2**

1. Яка мова програмування є основою для PHP?
2. Для яких задач призначена PHP?
3. Де розміщується код PHP? Що потрібно для виконання PHP програм?
4. Як оголошуються змінні в PHP?
5. Назвіть типи даних в PHP.
6. Назвіть основні оператори та операціїв PHP.

7. Які в PHP є оператори розгалудження, вибору, їх синтаксис?
8. Які в PHP є оператори циклів, їх синтаксис?
9. Яка роль операторів передачі керування?
10. Для чого використовуються оператори під’єднання в PHP?
11.

# Тема 3. Реалізація технології клієнт-сервер на PHP

Internet побудований за багаторівневим принципом, від фізичного рівня, пов'язаного з фізичними аспектами передачі двійкової інформації, і до прикладного рівня, що забезпечує інтерфейс між користувачем і мережею.

HTTP (HyperText Transfer Protocol, протокол передачі гіпертексту ) – це протокол прикладного рівня, розроблений для обміну гіпертекстовою інформацією в Internet.

HTTP надає набір методів для вказівки цілей запиту, що відправляється серверу. Ці методи засновані на дисципліні посилань, де для вказівки ресурсу, до якого має бути застосований цей метод, використовується універсальний ідентифікатор ресурсів (Universal Resource Identifier) у вигляді місцезнаходження ресурсу (Universal Resource Locator, URL ) або у вигляді його універсального імені (Universal Resource Name, URN ).

Повідомлення по мережі при використанні протоколу HTTP передаються у форматі, схожому з форматом поштового повідомлення Internet (RFC – 822) або з форматом повідомлень MIME (Multipurpose Internet Mail Exchange).

HTTP використовується для комунікацій між різними призначеними для користувача програмами і програмами-шлюзами, що надають доступ до існуючим Internet-протоколам, таким як SMTP (протокол електронної пошти), NNTP (протокол передачі новин), FTP (протокол передачі файлів), Gopher і WAIS. HTTP розроблений для того, щоб дозволяти таким шлюзам через проміжні програми- сервери (proxy) передавати дані без втрат.

Протокол реалізує принцип запит/відповідь. Програма-клієнта ініціює взаємодію з програмою, що відповідає серверам, і надсилає запит, що містить:

- метод доступу;
- адреса URL;
- версію протоколу;
- сполучення (схоже за формою на MIME) з інформацією про тип передаваних даних, інформацією про клієнта, що послав запит, і, можливо, зі змістовною частиною (тілом) повідомлення.

Відповідь сервера містить:

- рядок стану, в який входить версія протоколу і код повернення (успіх або помилка);
- повідомлення (у формі, схожій на MIME), в яке входить інформація сервера, метаінформація (тобто інформація про зміст повідомлення)

і тіло повідомлення.

У протоколі не вказується, хто повинен відкривати і закривати з'єднання між клієнтом і сервером. На практиці з'єднання, як правило, відкриває клієнт, а сервер після відправки відповіді ініціює його розрив.

Давайте розглянемо детальніше, в якій формі вирушають запити на сервер.

### 3.1 Форма запиту клієнта

Клієнт посилає серверу запит в одній з двох форм : в повній або скороченій. Запит в першій формі називається відповідно до повним запитом, а в другій формі – простим запитом.

Простий запит містить метод доступу і адресу ресурсу. Формально це можна записати так:

```
<Простий-запит> :=<Метод> <символ пропуск>
 <Запитаний-URL> <символ нового рядка>
```

В якості методу можуть бути вказані GET, POST, HEAD, PUT, DELETE і інші. Про найбільш поширених з них ми поговоримо трохи пізніше. Як прошеного URL найчастіше використовується URL –адрес ресурсу.

Приклад простого запиту :

```
GET http://phpbook.info/
```

Тут GET – це метод доступу, тобто метод, який має бути застосований до запитаного ресурсу, а http://phpbook.info/ - це URL –адрес запитаного ресурсу.

Повний запит містить рядок стану, декілька заголовків (заголовок запиту, загальний заголовок або заголовок змісту) і, можливо, тіло запиту. Формально загальний вигляд повного запиту можна записати так:

```
<Повний запит> :=<Рядок Стану>
 (<Загальний заголовок>|<Заголовок запиту>|
 <Заголовок змісту>)
 <символ нового рядка>
 [<зміст запиту>]
```

Квадратні дужки тут означають необов'язкові елементи заголовка, через вертикальну риску перераховані альтернативні варіанти. Елемент <Рядок

стану> містить метод запиту і URL ресурсу (як і простий запит ) і, крім того, використовувану версію протоколу HTTP. Наприклад, для виклику зовнішньої програми можна задіяти наступний рядок стану :

```
POST http://phpbook.info/cgi-bin/test HTTP/1.0
```

В даному випадку використовується метод POST і протокол HTTP версії 1.0.

У обох формах запиту важливе місце займає URL запитаного ресурсу. Частіше усього URL використовується у вигляді URL –адреси ресурсу. При зверненні до сервера можна застосовувати як повну форму URL, так і спрощену.

Повна форма утримує тип протоколу доступу, адресу сервера ресурсу і адресу ресурсу на сервері (рисунок 3.1).

У скороченій формі опускають протокол і адресу сервера, вказуючи тільки місце розташування ресурсу від кореня сервера. Повну форму використовують, якщо можлива пересилка запиту іншому серверу. Якщо ж робота відбувається тільки з одним сервером, то частіше застосовують скорочену форму.

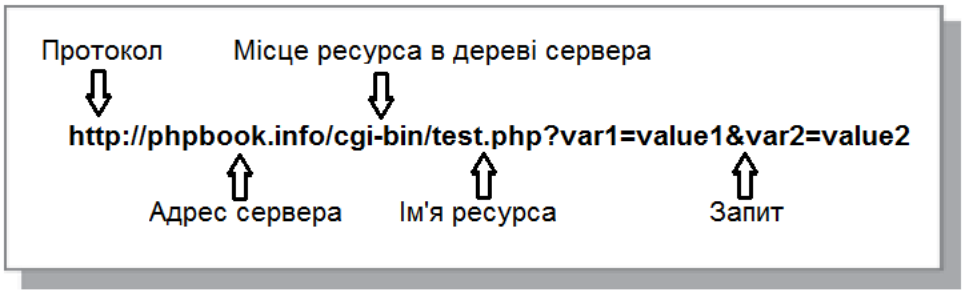



Рисунок 3.1. Повна форма URL

Далі ми розглянемо найбільш поширені методи відправки запитів.

Як вже говорилося, будь-який запит клієнта до сервера повинен розпочинатися з вказівки методу. Метод повідомляє про мету запиту клієнта. Протокол HTTP підтримує досить багато методів, але реально використовуються тільки три: POST, GET і HEAD . Метод GET дозволяє отримати будь-які дані, ідентифіковані за допомогою URL в запиті ресурсу. Якщо URL вказує на програму, то повертається результат роботи програми, а не її текст (якщо, звичайно, текст не є результат її роботи). Додаткова інформація, необхідна для обробки запиту, вбудовується в сам запит (у адресний рядок). При використанні методу GET в поле тіла ресурсу

повертається інформація (текст HTML –документа, наприклад), що власне зажадалася.

Існує різновид методу GET – умовний GET . Цей метод повідомляє сервер про те, що на запит треба відповісти, тільки якщо виконана умова, що міститься в полі if, - Modified – Since заголовка запиту. Якщо говорити точніше, то тіло ресурсу передається у відповідь на запит, якщо цей ресурс змінювався після дати, вказаної в if, - Modified – Since.

Метод HEAD аналогічний методу GET, тільки не повертає тіло ресурсу і не має умовного аналога. Метод HEAD використовують для отримання інформації про ресурс. Це може згодитися, наприклад, при рішенні задачі тестування гіпертекстових посилань.

Метод POST розроблений для передачі на сервер такої інформації, як анотації ресурсів, новинні і поштові повідомлення, дані для додавання у базу даних, тобто для передачі інформації великого об'єму і досить важливої. На відміну від методів GET і HEAD, в POST передається тіло ресурсу, яке і є інформацією, що отримується з полів форм або інших джерел введення.

Досі ми тільки теоретизували, знайомилися з основними поняттями. Тепер пора навчитися використати усе це на практиці. Далі в лекції ми розглнемо, як посилати запити серверу і як обробляти його відповіді.

### 3.2 Використання HTML–форм для передачі даних на сервер

Як передавати дані серверу? Для цього в мові HTML є спеціальна конструкція – форми. Форми призначені для того, щоб отримувати від користувача інформацію. Наприклад, вам треба знати логін і пароль користувача для того, щоб визначити, на які сторінки сайту його можна допускати. Чи вам потрібні особисті дані користувача, щоб була можливість з ним зв'язатися. Форми якраз і застосовуються для введення такої інформації. У них можна вводити текст або вибирати відповідні варіанти зі списку. Дані, записані у форму, вирушають для обробки спеціальній програмі (наприклад, скрипту на PHP) на сервері. Залежно від введених користувачем даних ця програма може формувати різні web-сторінок, відправляти запити до бази даних, запускати різні застосування і тому подібне

Розберемося з синтаксисом HTML –форм. Можливо, багато хто з ним знайомий, але ми все ж повторимо основні моменти, оскільки це важливо.

Отже, для створення форми в мові HTML використовується тег FORM . Усередині нього знаходиться одна або декілька команд INPUT . За допомогою атрибутів action і method тега FORM задаються ім'я програми, яка

оброблятиме ці форми, і метод запиту, відповідно. Команда INPUT визначає тип і різні характеристики прошеної інформації. Відправка даних форми відбувається після натиснення кнопки input типу submit . Створимо форму для реєстрації учасників заочної школи програмування.

```
<h2>Форма для реєстрації учасників</h2>
<form action=»1.php» method=POST> <!-створюємо
  форму. Ці форми оброблятиме файл 1.php, при
  відправці запиту буде використаний метод POST
Ім’я <br><input type=text name=»first_name»
 value=»Введіть Ваше ім’я»><br>
Прізвище <br><input type=text name=»last_name»><br>
E - mail <br><input type=text name=»email»><br><p>
Виберіть курс, який ви б хотіли відвідувати :<br>
<input type=radio name=»kurs» value=»PHP»>PHP<br>
<input type=radio name=»kurs» value=»Lisp»>Lisp<br>
<input type=radio name=»kurs» value=»Perl»>Perl<br>
<input type=radio name=»kurs» value=»Unix»> Unix
   <br> </p>
<p>Що ви хочете, щоб ми знали про вас? <BR>
<textarea name=»comment» cols=32 rows=5> </textarea>
    </p>
<p><input name=»confirm» type=checkbox
 checked>Підтвердити отримання <br>
<input type=submit value=»Відправити»>
<input type=reset value=»Відмінити»></p>
</form>
```

Після обробки браузером цей файл виглядатиме приблизно так, як показано на рисунку 3.2

Ось так створюються і виглядають HTML –форми. Вважатимемо, що ми навчилися або згадали, як їх створювати. Як ми бачимо, у формі можна вказувати метод передачі даних. Подивимося, що відбуватиметься, якщо вказати метод GET або POST, і в чому буде різниця.

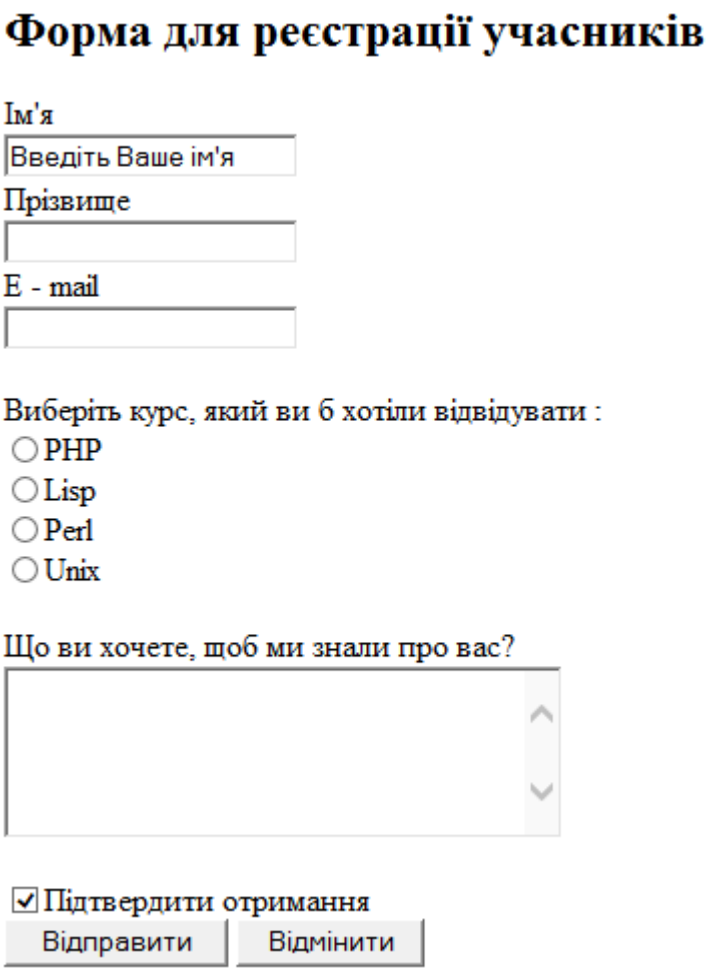



Рисунок 3.2. Приклад html –форми

### 3.3 Передавання методом GET

При відправці даних форми за допомогою методу GET вміст форми додається до URL після знаку питання у вигляді пар ім’я=значення, об’єднаних за допомогою амперсанда &:

```
action?name1=value1&name2=value2&name3=value3
```

Тут action – це URL –адрес програми, яка повинна обробляти форму (це або програма, задана в атрибуті action тега form, або сама поточна програма, якщо цей атрибут опущений). Імена name1, name2, name3 відповідають іменам елементів форми, а value1, value2, value3 – значенням цих елементів. Усі спеціальні символи, включаючи = і &, в іменах або значеннях цих параметрів будуть закодовані. Тому не варто використати в назвах або значеннях елементів форми ці символи і символи кирилиці в ідентифікаторах.

Якщо в полі для введення ввести який-небудь службовий символ, то він буде переданий в його шістнадцятковому коді, наприклад, символ $

заміниться на %24. Так само передаються і кириличні букви.

Для полів введення тексту і пароля (це елементи input з атрибутом type=text і type=password ), значенням буде те, що введе користувач. Якщо користувач нічого не вводить в таке поле, то в рядку запиту буде присутнім елемент name=, де name відповідає імені цього елементу форми.

Для кнопок типу checkbox і radio button значення value визначається атрибутом VALUE у тому випадку, коли кнопка відмічена. Не відмічені кнопки при складанні рядка запиту ігноруються цілком. Декілька кнопок типу checkbox можуть мати один атрибут NAME (і різні VALUE ), якщо це необхідно. Кнопки типу radio button призначені для одного з усіх запропонованих варіантів і тому повинні мати однаковий атрибут NAME і різні атрибути VALUE.

В принципі створювати HTML –форму для передачі даних методом GET не обов'язково. Можна просто додати в рядок URL потрібні змінні і їх значення.

```
http://phpbook.info/test.php?id=10&user=pit
```

У зв'язку з цим у передачі даних методом GET є один істотний недолік – будь-хто може підробити значення параметрів. Тому не радимо використати цей метод для доступу до захищених паролем сторінок, для передачі інформації, що впливає на безпеку роботи програми або сервера. Крім того, не варто застосовувати метод GET для передачі інформації, яку не дозволено змінювати користувачеві.

Незважаючи на усі ці недоліки, використати метод GET досить зручно при відладці скриптів (тоді можна бачити значення і імена передаваних змінних) і для передачі параметрів, що не впливають на безпеку.

### 3.4 Передавання методом POST

Вміст форми кодується точно так, як і для методу GET (див. вище), але замість додавання рядка до URL вміст запиту посилається блоком даних як частину операції POST . Якщо є присутнім атрибут ACTION, то значення URL, яке там знаходиться, визначає, куди посилати цей блок даних. Цей метод, як вже відзначалося, рекомендується для передачі великих за об'ємом блоків даних.

Інформація, введена користувачем і відправлена серверу за допомогою методу POST, подається на стандартне введення програмі, вказаній в атрибуті action, або потоковому скрипту, якщо цей атрибут опущений. Довжина

посиланого файлу передається в змінній оточення CONTENT_LENGTH, а тип даних – в змінній CONTENT_TYPE.

Передати дані методом POST можна тільки за допомогою HTML – форми, оскільки дані передаються в тілі запиту, а не в заголовку, як в GET . Відповідно і змінити значення параметрів можна, тільки змінивши значення, введене у форму. При використанні POST користувач не бачить передавані серверу дані.

Основна перевага POST запитів – це їх велика безпека і функціональність в порівнянні з GET –запросами. Тому метод POST частіше використовують для передачі важливої інформації, а також інформації великого об'єму. Проте не варто цілком покладатися на безпеку цього механізму, оскільки дані POST запиту також можна підробити, наприклад створивши html –файл на своїй машині і заповнивши його потрібними даними. Крім того, не усі клієнти можуть застосовувати метод POST, що обмежує варіанти його використання.

При відправці даних на сервер будь-яким методом передаються не лише самі дані, введені користувачем, але і ряд змінних, що називаються змінними оточення, характеризують клієнта, історію його роботи, шляхи до файлів і тому подібне. Ось деякі зі змінних оточення :

CONTENT_LENGTH - Довжина файлу, що надсилається;

CONTENT_TYPE – тип даних, які надсилаються;

REMOTE_ADDR – IP – адреси хоста(комп'ютера), що відправляє запит;

REMOTE_HOST – ім 'я хоста, з якого відправлений запит;

HTTP_REFERER – адреси сторінки, що посилається на поточний скрипт;

REQUEST_METHOD – метод, який був використаний при відправці запит;

QUERY_STRING – інформація, що знаходиться в URL після знаку питання;

SCRIPT_NAME – віртуальний шлях до програми, яка повинна виконуватися;

HTTP_USER_AGENT – інформація про браузер, який використовує клієнт .

### 3.5 Обробка запитів за допомогою PHP

Досі ми згадували тільки, що запити клієнта обробляються на сервері за допомогою спеціальної програми. Насправді цю програму ми можемо

написати самі, у тому числі і на мові PHP, і вона робитиме з отриманими даними усе, що ми захочемо. Для того, щоб написати цю програму, необхідно познайомитися з деякими правилами і інструментами, пропонованими для цих цілей PHP.

Усередині PHP –скрипта є декілька способів діставання доступу до даних, переданих клієнтом по протоколу HTTP. До версії PHP 4.1.0 доступ до таких даних здійснювався по іменах переданих змінних (нагадаємо, що дані передаються у вигляді пар «ім'я змінної, символ «=», значення змінної»). Таким чином, якщо, наприклад, було передано first_name=Nina, то усередині скрипта з'являлася змінна $first_name зі значенням Nina. Якщо вимагалося розрізняти, яким методом були передані дані, то використовувалися асоціативні масиви $HTTP_POST_VARS і $HTTP_GET_VARS, ключами яких були імена переданих змінних, а значеннями – відповідно значення цих змінних. Таким чином, якщо пара first_name = Nina передана методом GET, то $HTTP_GET_VARS[«first_name»]=»Nina».

Використати в програмі імена переданих змінних безпосередньо небезпечно. Тому було вирішено починаючи з PHP 4.1.0 задіяти для звернення до змінних, переданих за допомогою HTTP –запитів, спеціальний масив - $_REQUEST . Цей масив містить дані, передані методами POST і GET, а також за допомогою HTTP cookies. Це суперглобальний асоціативний масив, тобто його значення можна отримати у будь-якому місці програми, використовуючи як ключ ім'я відповідної змінної (елементу форми).

Приклад. Припустимо, ми створили форму для реєстрації учасників заочної школи програмування, як в наведеному вище прикладі. Тоді у файлі 1.php, оброблювальному цю форму, можна написати наступне:

```
<?php
$str = «Вітаємо,
 «.$_REQUEST[«first_name»]. «
 «.$_REQUEST[«last_name»]».! <br>»;
$str .=» Ви вибрали для вивчення курс по
 «.$_REQUEST[«kurs»];
echo $str;
?>
```

Тоді, якщо у форму ми ввели ім'я «Василь», прізвище «Петренко» і вибрали серед усіх курсів курс по PHP, на екрані браузеру отримаємо таке повідомлення:

```
Вітаємо, Василь Петренко!
Ви вибрали для вивчення курс по PHP
```

Після введення масиву $_REQUEST масиви $HTTP_POST_VARS і $HTTP_GET_VARS для однорідності були перейменовані в $_POST і $_GET відповідно, але самі вони з ужитку не зникли з міркувань сумісності з попередніми версіями PHP. На відміну від своїх попередників, масиви $_POST і $_GET стали суперглобальними, тобто доступними безпосередньо і усередині функцій і методів.

Наведемо приклад використання цих масивів. Припустимо, нам треба обробити форму, що містить елементи введення з іменами first_name, last_name, kurs (наприклад, форму form.html, приведену вище). Дані були передані методом POST, і дані, передані іншими методами, ми обробляти не хочемо. Це можна зробити таким чином:

```
<?
php
$str = «Вітаємо, .$_POST [«first_name»]».
 «.$_POST [«last_name»] «.! <br>»;
$str .= «Ви вибрали для вивчення курс по «.
 $_POST[«kurs»];
echo $str;
?>
```

Тоді на екрані браузеру, якщо ми ввели ім’я «Василь», прізвище «Петренко» і вибрали серед усіх курсів курс по PHP, побачимо повідомлення, як в попередньому прикладі:

```
Вітаємо, Василь Петренко!
Ви вибрали для вивчення курс по PHP
```

Для того, щоб зберегти можливість обробки скриптів більше ранніх версій, ніж PHP 4.1.0, була введена директива register_globals, що дозволяє або забороняє доступ до змінних безпосередньо по їх іменах. Якщо у файлі налаштувань PHP параметр register_globals=On, то до змінних, переданих серверу методами GET і POST, можна звертатися просто по їх іменах (тобто можна писати $first_name ). Якщо ж register_globals=Off, то треба писати $_REQUEST[«first_name»] чи $_POST[«first_name»], $_GET[«first_name»], $HTTP_POST_VARS[«first_name»], $HTTP_GET_VARS[«first_name»]. З точки зору безпеки цю директиву краще відключати (тобто

register_globals=Off ). При включеній директиві register_globals перелічені вище масиви також міститимуть дані, передані клієнтом.

Іноді виникає необхідність упізнати значення якої-небудь змінної оточення, наприклад метод, що використався при передачі запиту або IP – адрес комп'ютера, що відправив запит. Отримати таку інформацію можна за допомогою функції getenv(). Вона повертає значення змінної оточення, ім'я якої передане їй в якості параметра.

```
<?
getenv('REQUEST_METHOD');
 // поверне використаний метод
echo getenv ('REMOTE_ADDR');
 // виведе IP –адрес користувача,
 // що надіслав запит
?>
```

Як ми вже говорили, якщо використовується метод GET, то дані передаються додаванням рядка запиту у вигляді пар «ім'я_змінної=значення до URL –адресу ресурсу». Усе, що записано в URL після знаку питання, можна отримати за допомогою команди

```
getenv('QUERY_STRING');
```

Завдяки цьому можна по методу GET передавати дані в якому-небудь іншому виді. Наприклад, вказувати тільки значення декількох параметрів через знак плюс, а в скрипті розбирати рядок запиту на частини або можна передавати значення усього одного параметра. В цьому випадку в масиві $_GET з'явиться порожній елемент з ключем, рівним цьому значенню (усьому рядку запиту), причому символ « + «, що зустрівся в рядку запиту, буде замінений на підкреслення «_».

Методом POST дані передаються тільки за допомогою форм, і користувач ( клієнт ) не бачить, які саме дані вирушають серверу. Щоб їх побачити, хакер повинен підмінити нашу форму своїй. Тоді сервер відправить результати обробки неправильної форми не туди, куди треба. Щоб цього уникнути, можна перевіряти адресу сторінки, з якою були послані дані. Це можна зробити знову ж таки за допомогою функції getenv() :

```
getenv('HTTP_REFERER');
```

Тепер самий час вирішити завдання, сформульоване на початку лекції.

Наведемо приклад обробки запиту за допомогою PHP. Нагадаємо, в чому полягало завдання, і уточнимо її формулювання. Треба написати форму для реєстрації учасників заочної школи програмування і після реєстрації відправити учасникові повідомлення. Ми назвали це повідомлення універсальним листом, але воно трохи відрізнятиметься від того листа, який ми склали на попередній лекції. Тут ми також не відправлятимемо що-небудь по електронній пошті, щоб не уподібнюватися спамерам, а просто згенеруємо це повідомлення і виведемо його на екран браузеру. Початковий варіант форми реєстрації ми вже приводили вище. Змінимо його так, щоб кожен той, що реєструється міг вибрати скільки завгодно курсів для відвідування, і не підтверджуватимемо отримання реєстраційної форми.

```
<h2>Форма для реєстрації студентів</h2>
<form action="1.php" method=POST>
Ім'я <br><input type=text name="first_name"
 value="Введіть Ваше ім'я"><br>
Прізвище <br><input type=text name="last_name"><br>
E-mail <br><input type=text name="email"><br>
<p>Виберіть курс, який ви б хотіли відвідувати: <br>
<input type=checkbox name='kurs[]' value='PHP'>PHP<br>
<input  type=checkbox  name='kurs[]'  value='Lisp'>  Lisp
<br>
<input  type=checkbox  name='kurs[]'  value='Perl'>Perl
<br>
<input  type=checkbox  name='kurs[]'  value='Unix'>Unix
<br>
<P>Що ви хочете, щоб ми знали про вас? <BR>
<textarea name="comment" cols=32 rows=5></textarea>
<input type=submit value="Відправити">
<input type=reset value="Відмінити">
</form>
```

Тут усе досить просто і зрозуміло. Єдине, що можна відмітити, - це спосіб передання значень елементу checkbox. Коли ми пишемо в імені елементу kurs[], це означає, що перший відмічений елемент checkbox буде записаний в перший елемент масиву kurs, другий відмічений checkbox, - в другий елемент масиву і так далі. Можна, звичайно, просто дати різні імена елементам checkbox, але це ускладнить обробку даних, якщо курсів буде багато.

Скрипт, який усе це розбиратиме і оброблятиме, називається 1.php (форма посилається саме на цей файл, що записано в її атрибуті action ). По

замовчуванню використовується для передачі метод GET, але ми вказали POST . За отриманими даними від людини, що зареєструвалася, скрипт генерує відповідне повідомлення. Якщо людина вибрала якісь курси, то йому виводиться повідомлення про час їх проведення і про лекторів, які їх читають. Якщо людина нічого не вибрала, то виводиться повідомлення про наступні збори заочної школи програмістів (ЗШП).

Скрипт 1.php, що обробляє форму form_final.html

```
<? // створимо масиви відповідностей курс-час його
   // проведення і курс-лектор
$times = array("PHP"=>"14.30", "Lisp"=>"12.00",
 "Perl"=>"15.00", "Unix"=>"14.00");
$lectors = array("PHP"=>"Василь Васильович", "Lisp"=>
     "Іван  Іванович",  "Perl"=>"Петро  Петренкоич",
     "Unix"=>"Семен Семенович");
define("SIGN", "З повагою, адміністрація");
 // визначаємо підпис листа як константу
define("MEETING_TIME", "18.00");
 // задаємо час зборів студентів
$date = "12 травня"; // задаємо дату проведення лекцій
 //починаємо складати текст повідомлення
$str = "Доброго дня, шановий ".$_POST["first_name"]
 ." ".$_POST["last_name"].!<br>";
$str .= "<br>Повідомляємо Вас, що;
$kurses = $_POST["kurs"]; // збережемо в цій змінній
 // список вибраних курсів
if (!isset($kurses)){ // якщо не вибраний жоден курс
 $event = "наступні збори студентів";
 $str.="$event відбудеться $date ";
    $str.=MEETING_TIME. "<br>";
} else { // якщо хоч би один курс вибраний
$event="вибрані Вами лекції відбудуться $date <ul>";
//функція count обчислює число елементів в масиві
 $lect = "";
 for ($i=0;$i<count($kurses);$i++){
     // для кожного вибраного курсу
     // запам'ятовуємо назву курсу
 $k = $kurses[$i];
     // складаємо повідомлення
 $lect = $lect."<li>лекція з $k в $times[$k]";
 $lect .= " (Ваш лектор, $lectors[$k])";
 }
 $event = $event . $lect . "</ul>";
 $str .= "$event";
}
```

```
$str .= "<br>". SIGN; // додаємо підпис
echo $str; // виводим повідомлення на екран
?>
```

**Контрольні питання до теми 3**

1. Яка роль протоколу HTTP в архітектурі клієнт-сервер?
2. Назвіть складові запиту клієнта.
3. З яких частин складається повна фонма URL?
4. Які засоби HTML використовуються для передавання даних від клієнта на сервер?
5. Які найбільш вживані методи передавання даних від клієнта до сервера? Яка між ними різниця?
6. Який із методів передавання є безпечніший, поясніть чому.
7. Які вбудовані засоби має PHP для отримання і обробки переданих даних?

# Тема 4. Функції в PHP

## 4.1 Прості функції, визначені користувачем

Для чого потрібні функції? Щоб відповісти на це питання, треба зрозуміти, що взагалі є функції. У програмуванні, як і в математиці, функція є відображення безлічі її аргументів на безліч її значень. Тобто функція для кожного набору значень аргументу повертає якісь значення, що є результатом її роботи. Навіщо потрібні функції, спробуємо пояснити на прикладі. Класичний приклад функції в програмуванні - це функція, що обчислює значення факторіалу числа. Тобто ми задаємо їй число, а вона повертає нам його факторіал. При цьому не треба для кожного числа, факторіал якого ми хочемо отримати, повторювати один і той же код - досить просто викликати функцію з аргументом, рівним цьому числу.

Функція обчислення факторіалу натурального числа

```
<?php
function fact($n){
 if ($n==0) return 1;
 else return $fact = $n * fact($n - 1);
}
echo fact(3);
 // можна було б написати echo (3*2);
 // але якщо число велике
echo fact(50);
 // то зручніше користуватися функцією
 // чим писати echo (50*49*48*...*3*2);
?>
```

Таким чином, коли ми здійснюємо дії, в яких простежується залежність від якихось даних, і при цьому, можливо, нам знадобиться виконувати такі ж дії, але з іншими початковими даними, зручно використати механізм функцій - оформити блок дій у вигляді тіла функції, а дані, що міняються, - в якості її параметрів.

Подивимося, як в загальному вигляді виглядає завдання (оголошення) функції. Функція може бути визначена за допомогою наступного синтаксису:

```
function Ім 'я_функції (параметр1, параметр2
 ... параметрN){
 Блок_команд
 return "значення, що повертає функція";
}
```

Якщо прямо так написати в php -програмі, то працювати нічого не буде. По-перше, Ім'я_функції і імена параметрів функції ( параметр1, параметр2 і так далі) повинні відповідати правилам іменування в PHP (і кириличних символів в них краще не використовувати). Імена функцій нечутливі до регістра. По-друге, параметри функції - це змінні мови, тому перед назвою кожного з них повинен стояти знак $. Ніяких трикрапок ставити в списку параметрів не можна. По-третє, замість слів блок_дій в тілі функції повинен знаходитися будь-який правильний PHP-код (не обов'язково залежний від параметрів). І нарешті, після ключового слова return повинно йти коректний php -вираз (що-небудь, що має значення). Крім того, у функції може і не бути параметрів, як і повертаного значення. Приклад правильного оголошення функції - функція обчислення факторіалу, приведена вище.

Як відбувається виклик функції? Вказується ім'я функції і в круглих дужках список значень її параметрів, якщо такі є:

```
<?php
Ім'я_функції ("значення_для_параметра1"
 "значення_для_параметра2",...);
?>
```

Коли можна викликати функцію? Здавалося б, дивне питання. Функцію можна викликати після її визначення, тобто у будь-якому рядку програми нижче блоку function f_name(){...}. У PHP3 це було дійсно так. Але вже в PHP4 такої вимоги немає. Вся річ у тому, як інтерпретатор обробляє отримуваний код. Єдиний виняток становлять функції, визначувані умовно (усередині умовних операторів або інших функцій). Коли функція визначається таким чином, її визначення повинне передувати її виклику.

Якщо функція одного разу визначена в програмі, то перевизначити або видалити її пізніше не можна. Попри те, що імена функцій нечутливі до регістра, краще викликати функцію по тому ж імені, яким вона була задана у визначенні.

Розглянемо детальніше аргументи функцій, їх призначення і використання.

У кожної функції може бути, як ми вже говорили, список аргументів. За допомогою цих аргументів у функцію передається різна інформація (наприклад, значення числа, факторіал якого потрібно підрахувати). Кожен аргумент є змінною або константою.

За допомогою аргументів дані у функцію можна передавати трьома

різними способами. Це передача аргументів за значенням (використовується по замовчуванню), по посиланню і завдання значення аргументів по замовчуванню. Розглянемо ці способи детальніше.

Коли аргумент передається у функцію за значенням, зміна значення аргументу усередині функції не впливає на його значення поза функцією. Щоб дозволити функції змінювати її аргументи, їх треба передавати по посиланню. Для цього у визначенні функції перед ім'ям аргументу слід написати знак амперсанд "&".

```
<?php
function add_label(&$data_str){
 $data_str .= "checked";
}
$str = "<input type=radio name=article ";
// нехай є такий рядок
echo $str ".><br>";
// виведе елемент форми - не відмічену радіо кнопку
add_label($str);
// викличемо функцію
echo $str ".><br>";
// це виведе вже відмічену радіо кнопку
?>
```

У функції можна визначати значення аргументів, використовувані по замовчуванню. Саме значення по замовчуванню має бути константним вираженням, а не змінною і не представником класу або викликом іншої функції.

У нас є функція, що створює інформаційне повідомлення, підпис до якого міняється залежно від значення переданого їй параметра. Якщо значення параметра не задане, то використовується підпис "Оргкомітет".

```
<?php
function Message($sign="Оргкомітет".){
// тут параметр sign має за
// умовчанням значення "Оргкомітет"
 echo "Наступні збори відбудуться завтра.<br>";
 echo $sign . "<br>"; }
Message();
 // викликаємо функцію без параметра.
 // В цьому випадку підпис - це Оргкомітет
Message("З повагою, Василь".);
 // В цьому випадку підпис
```

```
 // буде "З повагою, Василь".
?>
```

Якщо у функції декілька параметрів, то ті аргументи, для яких задаються значення по замовчуванню, мають бути записані після усіх інших аргументів у визначенні функції. Інакше з'явиться помилка, якщо ці аргументи будуть опущені при виклику функції.

### 4.2 Функції із списками аргументів змінної довжини

У PHP4 можна створювати функції зі змінним числом аргументів. Тобто ми створюємо функцію, не знаючи заздалегідь, із скількома аргументами її викличуть. Для написання такої функції ніякого спеціального синтаксису не потрібно. Усе робиться за допомогою вбудованих функцій func_num_args(), func_get_arg(), func_get_args().

Функція func_num_args() повертає число аргументів, переданих в поточну функцію. Ця функція може використовуватися тільки усередині визначення призначеної для користувача функції. Якщо вона з'явиться поза функцією, то інтерпретатор видасть попередження.

```
<?php
function DataCheck(){
 $n = func_num_args();
 echo "Число аргументів функції $n";
}
DataCheck();
 // виведе рядок "Число аргументів функції 0"
DataCheck(1,2,3);
 // виведе рядок "Число аргументів функції 3"
?>
```

Функція func_get_arg (ціле номер_аргументу ) повертає аргумент зі списку переданих у функцію аргументів, порядковий номер якого заданий параметром номер_аргументу. Аргументи функції вважаються починаючи з нуля. Як і func_num_args(), ця функція може використовуватися тільки усередині визначення якої-небудь функції.

Номер_аргументу не може перевищувати число аргументів, переданих у функцію. Інакше буде згенеровано попередження, і функція func_get_arg() поверне False.

Створимо функцію для перевірки типу цих її аргументів. Вважаємо, що перевірка пройшла успішно, якщо перший аргумент функції - ціле число,

другий - рядок.

```
<?
function DataCheck(){
 $check =true;
 $n = func_num_args();
 // число аргументів переданих у функцію
 /* перевіряємо, чи являється перший
 переданий аргумент цілим числом */
 if ($n>=1) if (!is_int(func_get_arg(0)))
 $check = false;
 /* перевіряємо, чи являється другий
 переданий аргумент рядком */
 if ($n>=2)
 if (!is_string(func_get_arg(1)))
 $check = false;
return $check;
}
if (DataCheck(123, "text"))
 echo "Перевірка пройшла успішно<br>";
else echo "Дані не задовольняють
 умовам<br>";
if (DataCheck(324))
 echo "Перевірка пройшла успішно<br>";
else echo "Дані не задовольняють умовам<br>";
?>
```

Функція func_get_args() повертає масив, що складається зі списку аргументів, переданих функції. Кожен елемент масиву відповідає аргументу, переданому функції. Якщо функція використовується поза визначенням призначеної для користувача функції, то генерується попередження.

Перепишемо попередній приклад, використовуючи цю функцію. Перевірятимемо, чи є цілим числом кожен парний аргумент, що передається функції :

```
<?
function DataCheck(){
 $check =true;
 $n = func_num_args();
 // число аргументів, переданих у функцію
 $args = func_get_args();
 // масив аргументів функції
 for ($i=0;$i<$n;$i++){
 $v = $args[$i];
```

```
 if ($i % 2 == 0){
 if (!is_int($v)) $check = false;
 // перевіряємо, чи є парний аргумент цілим
 }
 }
return $check;
}
if (DataCheck("text", 324))
 echo "Перевірка пройшла успішно<br>";
else echo "Дані не задовольняють
 умовам<br>";
?>
```

Як бачимо, комбінації функцій func_num_args(), func_get_arg() і func_get_args() використовується для того, щоб функції могли мати змінний список аргументів. Ці функції були додані тільки в PHP 4. У PHP3 для того, щоб добитися подібного ефекту, можна використати в якості аргументу функції масив. Наприклад, ось так можна написати скрипт, перевіряючий, чи є кожен непарний параметр функції цілим числом:

```
<?
function DataCheck($params){
 $check =true;
 $n = count($params);
 // число аргументів переданих у функцію

 for ($i=0;$i<$n;$i++){
 $v = $params[$i];
 if ($i % 2 != 0){
 // перевіряємо, чи являється непарний
 // аргумент цілим
if (!is_int($v)) $check = false;
 }
 }
return $check;
}
if (DataCheck("text", 324))
echo "Перевірка пройшла успішно<br>";
else echo "Дані не задовольняють умовам<br>";
?>
```

### 4.3 Використання глобальних і статичних змінних усередині функції

Щоб використати усередині функції змінні, задані поза нею, ці змінні

треба оголосити як глобальні. Для цього в тілі функції слід перерахувати їх імена після ключового слова global :

global $var1, $var2;

```
<?
$a=1;
function Test_g(){
global $a;
 $a = $a*2;
 echo 'в результаті роботи функції $a=',$a;
}
echo 'поза функцією $a=',$a';
Test_g();
echo "<br>";
echo 'поза функцією $a=',$a';
Test_g();
?>
```

В результаті роботи цього скрипта отримаємо:

```
поза функцією $a=1, в результаті роботи функції $a=2
поза функцією $a=2, в результаті роботи функції $a=4
```

Коли змінна оголошується як глобальна, фактично створюється посилання на глобальну змінну. Тому такий запис еквівалентний наступним (масив $GLOBALS містить усі змінні, глобальні відносно поточній зоні видимості) :

```
$var1 = &$GLOBALS["var1"];
$var2 = &$GLOBALS["var2"];
```

Це означає, наприклад, що видалення змінної $var1 не видаляє глобальної змінної $GLOBALS["var1"].

Щоб використати змінні тільки усередині функції, при цьому зберігаючи їх значення і після виходу з функції, треба оголосити ці змінні як статичні. Статичні змінні видно тільки усередині функції і не втрачають свого значення, якщо виконання програми виходить за межі функції. Оголошення таких змінних робиться за допомогою ключового слова static :

```
static $var1, $var2;
```

Статичною змінною може бути присвоєне будь-яке значення, але не посилання.

```
<?
function Test_s(){
static $a = 1;
// не можна привласнювати вираження або посилання
 $a = $a*2;
 echo $a;
}
Test_s(); // виведе 2
echo $a; // нічого не виведе, оскільки $a доступна тільки
     // усередині функції
Test_s(); // усередині функції $a=2, тому
 // результатом роботи функції
     // буде число 4
?>
```

### 4.4 Визначення результатів, що повертаються функціями

Усі функції, приведені вище в якості прикладів, виконували які-небудь дії. Окрім подібних дій, будь-яка функція може повертати як результат своєї роботи яке-небудь значення. Це робиться за допомогою затвердження return. Повертане значення може бути будь-якого типу, включаючи списки і об'єкти. Коли інтерпретатор зустрічає команду return в тілі функції, він негайно припиняє її виконання і переходить на той рядок, з якого була викликана функція.

Наприклад, складемо функцію, яка повертає вік людини. Якщо людина не померла, то вік вважається відносно поточного року.

```
<?php
/* якщо другий параметр обчислюється
як true, то він розглядається як дата смерті, */

function Age($birth, $is_dead){
 if ($is_dead) return $is_dead -$birth;
 else return date("Y") - $birth;
}
echo Age(1971, false); // для 2009 року виведе 38
echo Age(1971, 2001); // виведе 30
?>
```

В даному прикладі можна було і не використати функцію return, а просто

замінити її функцією виведення echo. Проте якщо ми все ж робимо так, що функція повертає якесь значення (в даному випадку вік людини), то в програмі ми можемо присвоїти будь-якій змінній значення цієї функції :

$an_age = Age(1981, 2004);

В результаті роботи функції може бути повернено тільки одне значення. Декілька значень можна отримати, якщо повертати список значень (одновимірний масив).

Коли функція повертає декілька значень для їх обробки в програмі, зручно використати мовну конструкцію list(), яка дозволяє однією дією присвоїти значення відразу декільком змінним. Наприклад, в попередньому прикладі, залишивши без зміни функцію, обробити повертані їй значення можна було так:

```
<?
// завдання функції Full_age()
list($day,$month,$year) = Full_age("07"
 "08","1974");
echo "Вам $year років, $month місяців і
 $day днів";
?>
```

Взагалі конструкцію list() можна використати для присвоєння змінним значень елементів будь-якого масиву.

```
<?
$arr = array("first", "second");
list($a,$b) = $arr;
 // змінною $a присвоюється перше
 // значення масиву, $b - друге
echo $a," ",$b; // виведе рядок "first second"
?>
```

### 4.5 Посилання як результат функції

В результаті своєї роботи функція також може повертати посилання на яку-небудь змінну. Це може згодитися, якщо вимагається використати функцію для того, щоб визначити, якою змінною має бути присвоєне посилання. Щоб отримати з функції посилання, треба при оголошенні перед її ім'ям написати знак амперсанд (&) і кожного разу при виклику функції перед її ім'ям теж писати амперсанд (&). Зазвичай функція повертає посилання на яку-небудь глобальну змінну (чи її частину - посилання на елемент

глобального масиву), посилання на статичну змінну (чи її частина) або посилання на один з аргументів, якщо він був також переданий по посиланню.

```
<?
$a = 3; $b = 2;
function & ref($par){
global $a, $b;
 if ($par % 2 == 0) return $b;
 else return $a;
}
$var =& ref(4);
echo $var, " і ", $b"<br>"; //виведе 2 і 2
$b = 10;
echo $var, " і ", $b"<br>"; // виведе 10 і 10
?>
```

При використанні синтаксису посилань в змінну $var нашого прикладу не копіюється значення змінної $b поверненою функцією $ref, а створюється посилання на цю змінну. Тобто тепер змінні $var і $b ідентичні і змінюватимуться одночасно.

### 4.6 Змінні функції

PHP підтримує концепцію змінних функцій. Це означає, що якщо ім'я змінної закінчується круглими дужками, то PHP шукає функцію з таким же ім'ям і намагається її виконати.

```
<?
/* створимо дві прості функції:
Add_sign - додає підпис до рядка і
Show_text - виводить рядок тексту */
function Add_sign($string
 $sign="З повагою, Петро"){
echo $string ". ".$sign;
}
function Show_text(){
 echo "Відправити повідомлення поштою<br>";
}
$func = "Show_text";
 // створюємо змінну зі значенням
 // рівним імені функції Show_text
$func();
 // це викличе функцію Show_text
$func = "Add_sign";
```

```
 // створюємо змінну зі значенням
 // рівним імені функції Add_sign
$func("Привіт всім <br>");
 // це викличе функцію
 // Add_sign з параметром "Привіт всім"
?>
```

В даному прикладі функція Show_text просто виводить рядок тексту. Здавалося б, навіщо для цього створювати окрему функцію, якщо існує спеціальна функція echo(). Річ у тому, що такі функції, як echo(), print(), unset(), include() і тому подібне не можна використати в якості змінних функцій. Тобто якщо ми напишемо:

```
<?
$func = "echo ";
$func("TEXT");
?>
```

то інтерпретатор виведе помилку:

```
Fatal error: Call to undefined function:
echo() in
c:\users\nina\tasks\func\var_f.php on line 2
```

Тому для того, щоб використати будь-яку з перелічених вище функцій як змінну функцію, треба створити власну функцію, що ми і зробили в попередньому прикладі.

### 4.7 Внутрішні (вбудовані) функції

Говорячи про функції, визначувані користувачем, все ж не можна не сказати пару слів про вбудовані функції. З деякими зі вбудованих функцій, такими як echo(), print(), date(), include(), ми вже познайомилися. Насправді усі перераховані функції, окрім date(), є мовними конструкціями. Вони входять в ядро PHP і не вимагають ніяких додаткових налаштувань і модулів. Функція date() теж входить до складу ядра PHP і не вимагає налаштувань. Але є і функції, для роботи з якими треба встановити різні бібліотеки і підключити відповідний модуль. Наприклад, для використання функцій роботи з базою даних MySql слід скомпілювати PHP з підтримкою цього розширення. Останнім часом найбільш поширені розширення і відповідно до їх функції

спочатку включають до складу PHP так, щоб з ними можна працювати без яких би то не було додаткових налаштувань інтерпретатора.

Приклад. Нехай потрібно створити інтерфейс, який дозволяв би створювати html -форми. Користувач вибирає, які елементи і в якій кількості треба створити, придумує їм назви, а наша програма сама генерує необхідну форму.

Розіб'ємо завдання на декілька підзадач: вибір типів елементів введення і їх кількості, створення назв елементів введення і обробка отриманих даних, тобто безпосередньо генерація форми. Перше завдання досить просте: треба написати відповідну форму, наприклад подібну до приведеної нижче :

```
<form action="ask_names.php">
Створити елемент "рядок введення тексту" : <input
 type=checkbox name=types[]
 value=string><br>
Кількість елементів : <input type=text
 name=numbers[string]
 size=3><br>
<br>
Створити елемент "текстова область": <input
 type=checkbox
 name=types[] value=text><br>
Кількість елементів : <input type=text
 name=numbers[text]
 size=3><br>
<input type=submit value="Створити">
</form>
```

Коли ми пишемо в імені елементу форми, наприклад types[], це означає, що його ім'я - наступний елемент масиву types. Тобто у нас перший елемент форми ("рядок введення тексту") матиме ім'я types[0], а другий (текстова область) - types[1]. У браузері task_form.html виглядатиме приблизно так:

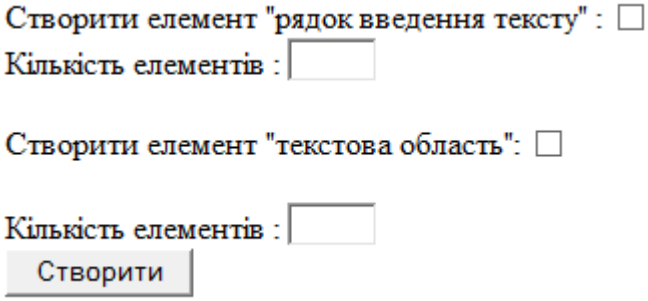



Рисунок 4.1. Форма для вибору створюваних елементів і їх кількості

Після відправки даних цієї форми ми отримаємо інформацію про те, які елементи і скільки елементів кожного типу треба створити. Наступний скрипт ask_names.php просить назви для цих елементів:

```
<?
$file = "task.php";
/* файл, який оброблятиме
згенеровану цим скриптом форму */
function Ask_names(){
 // функція генерує форму для
 // введення назв елементів введення
global $file;
 //оголошуємо, що хочемо використати цю
 // змінну, задану поза функцією
if (isset($_GET["types"])){
 $st = '<form action="'.$file.'">';
 foreach ($_GET["types"] as $k => $type){
/* перебираємо усі типи елементів,
які треба створити */

 $num = $_GET["numbers"][$type];
 // скільки елементів кожного типу потрібні
 for ($i=1;$i<=$num;$i++){
 // створюємо $num рядків для введення
 $st.="Введіть ім'я $i-го елементу типу $type: ";
 $st.="<input type=text name=names[$type][]><br>";
 }
 // зберігаємо тип і число необхідних
 // елементів введення цього типу
$st.="<input type=hidden name=types[] value=$type>";
$st.= "<input type=hidden name=numbers[] value=
$num><br>";
}
$st.="<input type=submit name=send value=send>
</form>";
return $st;
 // у змінній $st міститься код форми
 // для запиту імен
} else echo "Select type";
}
echo Ask_names();
 // викликаємо функцію і виводимо
 // результати її роботи
?>
```

Припустимо, треба створити два елементи типу "текстовий рядок" і один

елемент типу "текстова область", як і відмічено у формі вище. Тоді скрипт ask_names.php обробить її таким чином, що ми отримаємо таку форму:

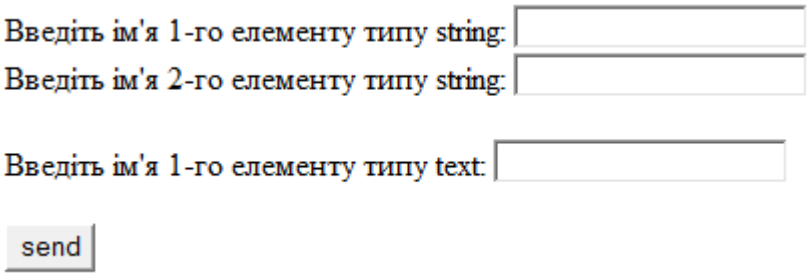



Рисунок 4.2 Форма для введення назв створюваних елементів

Введемо в цю форму, наприклад, рядка "Назва", "Автор" і "Короткий зміст". Ці дані оброблятиме скрипт task.php:

```
<?
$show_file = "task_show.php";
/* файл, який оброблятиме дані
 створеною цим файлом форми */
function Create_element($type,$name){
 // функція створює елемент введення
 // за типом і назвою
 $str="";
 switch($type){
 case "string":
  $str.="$name:<input    type=text    name=string[]>
     <br>"; break;
 case "text":
  $str.="$name:   <textarea   name=text[]></textarea>
     <br>";    break;
 }
return $str;
}
function Create_form(){
 // функція створює форму
 // з потрібними елементами
global $show_file;
 $str = '<form action="'.$show_file.'">';
 foreach ($_GET["types"] as $k => $type){
 // перебираємо типи елементів
 $num = $_GET["numbers"][$k];
// число елементів цього типу
     for ($i=1;$i<=$num;$i++){
 $arr = $_GET["names"][$type][$i - 1];
```

```
 // ім'я створюваного елементу
 $str.=Create_element($type,$arr);
 // викликаємо функцію для
 // створення елементу
 }
 }
 $str .= "<input type=submit value=send></form>";
 echo $str;
}
$crt = "Create_form";
$crt(); // викликаємо функцію створення
 // форми Create_form
?>
```

Результатом роботи цього скрипта з вхідними даними, приведеними вище, буде наступна форма:

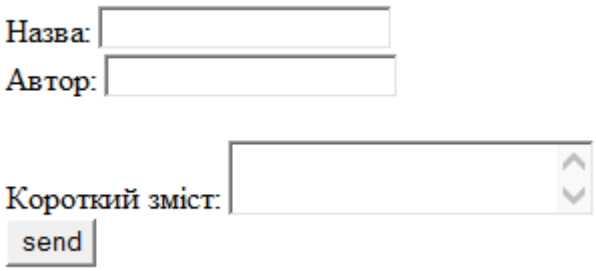



Рисунок 4.3 Приклад форми, що генеруєся програмою

**Контрольні питання до теми 4**

1. Для чого потрібно створювати власні функції?
2. Які види функцій користувача можна створити в PHP?
3. Як можуть передаватись аргументи у функції?
4. Для чого використовується передавання аргументів за посиланням?
5. Як організувати повернення декількох результатів однією функцією,
6. Які переваги можна отримати від створення змінних-функцій?
7. В чому різниця між статичними і глобальними змінними?
8. Як задати функцію, яка повертає посилання на результат?

# Тема 5. Сеанси та сесії в PHP

## 5.1 Авторизація доступу

Що таке авторизація доступу? Спробуємо пояснити на прикладі із звичайного життя. Ви хочете узяти у бібліотеці книгу. Але ця послуга доступна тільки тим, у кого є читацький квиток. Можна сказати, що за допомогою цього квитка робиться "авторизація доступу" до бібліотечних ресурсів. Бібліотекар після пред'явлення йому читацького квитка знає, хто бере книгу, і у разі потреби (наприклад, книгу довго не повертають) може вжити заходи (подзвонити боржникові додому). Бібліотекар має значно більше прав, чим звичайний відвідувач: він може давати або не давати книги певному відвідувачеві, може виставляти напоказ новинки і прибирати в архів рідко читані книги і тому подібне.

У інформаційних технологіях усе приблизно так же. В мережі існує величезна кількість ресурсів, тобто безліч "бібліотек". У кожної з них свій "бібліотекар", тобто людина або група людей, що відповідають за зміст ресурсу і надання користувачам інформації. Їх називають адміністраторами. Функції адміністратора, як правило, включають додавання нової інформації, видалення і редагування існуючої, налаштування способів відображення інформації користувачеві. А у функції користувача (простого відвідувача ресурсу) входить тільки пошук і перегляд інформації.

Як же відрізнити користувача від адміністратора? У реальній бібліотеці це якось очевидно, але якщо ролі бібліотекаря і відвідувача бібліотеки перенести у віртуальну реальність, то ця очевидність зникає. Бібліотекар, як і відвідувач, має доступ до бібліотечних ресурсів через Internet. А згідно з протоколом HTTP усі клієнти абсолютно рівноправні. Як же зрозуміти, хто зайшов на сайт? Звичайний користувач (відвідувач) або адміністратор (бібліотекар)? Якщо це простий користувач, то як зберегти це знання, щоб не допустити відвідувача в закриті архіви сайту? Тобто виникає питання, як ідентифікувати клієнта, який послав запит, і зберігати відомості про нього, поки він знаходиться на сайті?

Найпростіший варіант, який приходить в голову, - це реєстрація людини в системі і видача йому аналога читацького квитка, а саме логіна і пароля для входу в адміністративну частину системи. Ця інформація зберігається на комп'ютері-сервері, і при вході в систему перевіряється відповідність введених користувачем логіна і пароля тим, що зберігаються в системі. Правда, тут в порівнянні з реальною бібліотекою ситуація змінюється: читацький квиток

потрібно бібліотекареві для входу в закриту частину системи, а читач може заходити на сайт вільно. В принципі можна реєструвати і простих відвідувачів. Тоді усіх зареєстрованих користувачів треба розділити на групи: бібліотекарі (адміністратори) і читачі (прості користувачі), наділивши їх відповідними правами. Ми не вдаватимемося до цих тонкощів і скористаємося найпростішим варіантом, коли введення логіна і пароля потрібно для доступу до деяких сторінок сайту.

У нас є файл index.html - домашня сторінка Василенка Василя.

```
<html>
<head><title>My home page</title></head>
<body>
Привіт всім!
Мене звуть Василь Василенко і це моя домашня сторінка.
<a href="secret_info.html">Для Петра</a>
</body></html>
```

і файл secret_info.html, який містить секретну інформацію, читати яку дозволено тільки Васиному другу Петру.

```
<html>
<head><title>Secret info</title></head>
<body>
Тут я хочу ділитися секретами з другом Петром.</p>
</body></html>
```

Якщо залишити обидва ці файли як є, то будь-який відвідувач, клікнувши на посилання "Для Петра", потрапить на секретну сторінку. Щоб цього уникнути, треба додати проміжний скрипт, який перевірятиме, чи дійсно Петро хоче потрапити на секретну сторінку. І зробити так, щоб головний файл посилався не відразу на secret_info.html, а спочатку на цей скрипт.

```
<html>
<head><title>My home page</title></head>
<body>
<p>Привіт всім!
Мене звуть Василь Василенко і це моя домашня сторінка.</p>
<a href="authorize.php">Для Петра</a>
</body>
</html>
```

Сам скрипт авторизації повинен надавати форму для введення логіна і пароля, перевіряти їх правильність і перенаправляти на секретну сторінку, якщо перевірка пройшла успішно, і видавати повідомлення про помилку інакше.

```
<?
if (!isset($_GET['go'])){
 // перевіряємо, чи відправлені дані формою
 echo "<form> // форма для авторизації (введення
             // логіна і пароля)
 Login: <input type=text name=login>
 Password: <input type=password name=passwd>
 <input type=submit name=go value=Go>
 </form>";
}else {
// якщо форма заповнена, то порівнюємо логін
// і пароль з правильними логіном і паролем
 if ($_GET['login']=="pit" &&
 $_GET['passwd']=="123") {
 Header("Location: secret_info.html");
 //і перенаправляємо на секретну сторінку
 }else echo "неправильний ввід,спробуйте ще раз
<br>";
}
?>
```

Начебто усе досить просто. Але припустимо, у нас не одна секретна сторінка, а декілька. Причому вони пов'язані між собою перехресними посиланнями. Тоді виникає необхідність постійно пам'ятати пароль і логін відвідувача сайту (якщо він такий має). Щоб розв'язати цю проблему, можна в кожну сторінку вбудувати скрипт, який передаватиме логін і пароль від сторінки до сторінки в якості прихованих параметрів форми. Але такий спосіб не зовсім безпечний: ці параметри можна перехопити і підробити. У PHP існує зручніший і безпечніший метод вирішення проблеми зберігання даних про відвідувача впродовж сеансу його роботи з сайтом - це механізм сесій.

### 5.2 Механізм сесій

Сесії - це механізм, який дозволяє створювати і використати змінні, що зберігають своє значення впродовж усього часу роботи користувача з сайтом.

Ці змінні для кожного користувача мають різні значення і можуть використовуватися на будь-якій сторінці сайту до виходу користувача з

системи. При цьому кожного разу, заходячи на сайт, користувач набуває нових значень змінних, що дозволяють ідентифікувати його впродовж цього сеансу або сесії роботи з сайтом. Звідси і назва механізму - сесії.

Завдання ідентифікації користувача вирішується шляхом присвоєння кожному користувачеві унікального номера, так званого ідентифікатора сесії (SID, Session IDentifier). Він генерується PHP у той момент, коли користувач заходить на сайт, і знищується, коли користувач йде з сайту, і є рядком з 32 символів (наприклад, ac4f4a45bdc893434c95dcaffb1c1811). Цей ідентифікатор передається на сервер разом з кожним запитом клієнта і повертається назад разом з відповіддю сервера.

Існує два основні способи передання ідентифікатора сесії:

- за допомогою cookies. Cookies були створені спеціально як метод однозначної ідентифікації клієнтів і є розширенням протоколу HTTP. В цьому випадку ідентифікатор сесії зберігається в тимчасовому файлі на комп'ютері клієнта, що послав запит. Метод, поза сумнівом, хороший, але багато користувачів відключають підтримку cookies на своєму комп'ютері через проблеми з безпекою;
- за допомогою параметрів командного рядка.В цьому випадку ідентифікатор сесії автоматично вбудовується в усі запити (URL), що передаються серверу, і зберігається на стороні сервера.

Наприклад: адреса http://mypage.ua/test.php перетворюється на адресу http:// mypage.ua /test.php?PHPSESSID=ac4f4a45bdc893434c95dcaffb1c1811

Цей спосіб передання ідентифікатора використовується автоматично, якщо у браузеру, що відправив запит, вимкнені cookies. Він досить надійний - передавати параметри в командному рядку можна завжди. З іншого боку, ідентифікатор сесії можна підглянути, скористатися збереженим варіантом в рядку браузеру або підробити. Хоча, звичайно, усі ці проблеми або надумані або їх можна вирішити. Наприклад, хто зможе запам'ятати рядок з 32 різних символів? А якщо правильно організувати роботу з сесіями (вчасно їх знищувати), то номер сесії, що навіть зберігся у браузері, нічого не дасть. До питань безпеки ми ще повернемося у кінці лекції.

### 5.2.1 Налаштування сесій

Перш ніж почати працювати з сесіями, слід розібратися в тому, як коректно настроювати їх обробку інтерпретатором PHP. Сама робота з сесіями в PHP підтримується по замовчуванню. Це означає, що встановлювати ніяких

додаткових елементів не треба. А ось знати, що записано в налаштуваннях цього модуля, корисно, щоб уникнути помилок при роботі з ним.

Налаштування PHP, у тому числі і для роботи з сесіями, прописуються у файлі php.ini. Звернемося до цього файлу.

Як ми вже знаємо, ідентифікатор сесії (число, по якому можна унікально ідентифікувати клієнта, що послав запит) зберігається або на комп'ютері-сервері, або на комп'ютері-клієнта, або і там, і там.

Параметр session.save_path в php.ini, визначає, де на сервері зберігатимуться ці сесії. Через нього найчастіше виникають проблеми для Windows-сервервв, тому що по замовчуванню значення session.save_path встановлене в /tmp. І якщо в кореневій директорії сервера такої теки немає, то при запуску сесій видаватиметься помилка.

Сервер може обробляти велику кількість сесій одночасно, і усі їх тимчасові файли зберігатимуться в директорії, заданій параметром session.save_path. Якщо система погано працює з теками великого розміру, то зручно використати піддиректорії. Для цього, окрім назви теки, в значення параметра додають ще і число, що визначає глибину вкладеності піддиректорій в цій теці: N;/dir. Це значення треба обов'язково узяти в лапки, оскільки крапка з комою є одним з символів коментарів у файлі налаштувань PHP. Усі директорії і піддиректорії для зберігання даних сесії треба створити самостійно.

Наприклад: 2;/Temp визначає, що змінні сесій зберігатимуться в теках виду c:\Temp\0\a\, c:\Temp\0\b\ і тому подібне.

Зберігання даних на стороні клієнта здійснюється за допомогою cookies. Роботу PHP з cookies можна настроїти, зокрема, за допомогою параметрів session.use_cookies, session.cookie_lifetime і т. п.

Параметр session.use_cookies визначає, чи використати cookies при роботі з сесіями. По замовчуванню ця опція включена (тобто набуває значення "1").

Параметр session.cookie_lifetime задає тривалість життя cookies в секундах. По замовчуванню це "0", тобто дані в cookies вважаються правильними до закриття вікна браузеру.

Окрім цих параметрів, корисними можуть виявитися session.name, що визначає ім´я сесії, session.auto_start, що дозволяє автоматично запускати сесії, session.serialize_handler, задаючий спосіб кодування даних сесії, і параметр session.cache_expire, визначає, через скільки хвилин застаріває документ в кеші.

Ім´я сесії session.name по замовчуванню встановлюється як PHPSESSID і використовується в cookies як ім´я змінної, в якій зберігається ідентифікатор сесії. Автоматичний запуск сесій по замовчуванню відключений, але його можна задати, зробивши значення session.auto_start рівним "1". Для кодування даних сесії по замовчуванню використовується php. Застарівання даних, збережених в кеші, відбувається через 180 хвилин.

### 5.2.2 Створення сесії

Перше, що треба зробити для роботи з сесіями (якщо вони вже налагоджені адміністратором сервера), це запустити механізм сесій. Якщо в налаштуваннях сервера змінна session.auto_start встановлена в значення "0" (якщо session.auto_start=1, то сесії запускаються автоматично), то будь-який скрипт, в якому треба використати ці сесії, повинен розпочинатися з команди

```
session_start();
```

Отримавши таку команду, сервер створює нову сесію або відновлює поточну, грунтуючись на ідентифікаторі сесії, переданому за запитом. Як це робиться? Інтерпретатор PHP шукає змінну, в якій зберігається ідентифікатор сесії (по замовчуванню це PHPSESSID) спочатку в cookies, потім в змінних, переданих за допомогою POST- і GET-запитів. Якщо ідентифікатор знайдений, то користувач вважається ідентифікованим, проводиться заміна усіх URL і виставлення cookies. Інакше користувач вважається новим, для нього генерується новий унікальний ідентифікатор, потім проводиться заміна URL і виставлення cookies.

Команду session_start() треба викликати в усіх скриптах, в яких належить використати змінні сесії, причому до виведення яких-небудь даних у браузер. Це пов'язано з тим, що cookies виставляються тільки до виведення інформації на екран.

Отримати ідентифікатор поточної сесії можна за допомогою функції session_id().

Для наочності сесії можна задати ім´я за допомогою функції session_name([ім´я_сесії]). Робити це треба ще до ініціалізації сесії. Отримати ім´я поточної сесії можна за допомогою цієї ж функції, викликаної без параметрів : session_name();

Приклад. Створення сесії

Перейменуємо наш файл index.html, щоб оброблялися php-скрипти, наприклад в Index.php, створимо сесію і подивимося, який вона отримає

ідентифікатор і ім´я.

```
<?
session_start();
 // створюємо нову сесію або відновлюємо поточну
echo session_id(); // виводимо ідентифікатор сесії
?>
<html>
<head><title>My home page</title></head>
... // домашня сторінка
</html>
<?
echo session_name(); // виводимо ім´я
                // поточної сесії.
           // В даному випадку це PHPSESSID
?>
```

Якщо виконати те ж саме з файлом authorize.php, то значення змінних (id сесії та її ім´я), що виводяться, будуть такими ж, якщо перейти на нього з index.php і не закривати перед цим вікно браузеру (тоді ідентифікатор сесії зміниться).

### 5.2.3 Реєстрація змінних сесії

Проте від самих ідентифікатора та імені сесії нам користі для вирішення наших завдань мало. Ми ж хочемо передавати і зберігати впродовж сесії наші власні змінні (наприклад, логін і пароль). Для того, щоб цього добитися, треба просто зареєструвати свої змінні:

session_register(ім´я_змінної1,ім´я_змінної2,...);

Помітимо, що реєструються не значення, а імена змінних. Зареєструвати змінну досить один раз на будь-якій сторінці, де використовуються сесії. Імена змінних передаються функції session_register() без знаку $. Усі зареєстровані таким чином змінні стають глобальними (тобто доступними з будь-якої сторінки) впродовж цієї сесії роботи з сайтом.

Зареєструвати змінну також можна, просто записавши її значення в асоціативний масив $_SESSION, тобто написавши

$_SESSION['ім´я_змінної'] = 'значення_змінної';

У цьому масиві зберігаються усі зареєстровані (тобто глобальні) змінні

сесії.

Доступ до таких змінних здійснюється за допомогою масиву $_SESSION['ім´я_змінної'] (чи $HTTP_SESSION_VARS['ім´я_змінної'] для версії PHP 4.0.6 і більше ранніх). Якщо ж в налаштуваннях php включена опція register_globals, то до сесійних змінних можна звертатися ще і як до звичайних змінних, наприклад так: $ім´я_змінної.

Якщо register_globals=off (відключені), то користуватися session_register() для реєстрації змінних переданих методами POST або GET, не можна, тобто це просто не працює. І взагалі, не рекомендується одночасно використати обидва методи реєстрації змінних, $_SESSION і session_register().

Приклад. Реєстрація змінних.Зареєструємо логін і пароль, що вводяться користувачем на сторінці авторизації.

```
<?
session_start(); // створюємо нову сесію
                 // або відновлюємо поточну
if (!isset($_GET['go'])){
 echo "<form>
 Login: <input type=text name=login>
 Password: <input type=password name=passwd>
 <input type=submit name=go value=Go>
 </form>";
}else {
 $_SESSION['login']=$_GET['login'];
        // реєструємо змінну login
 $_SESSION['passwd']=$_GET['passwd'];
        // реєструємо змінну passwd
        // тепер логін і пароль - глобальні
        // змінні для цієї сесії
 if ($_GET['login']=="pit" &&
 $_GET['passwd']=="123") {
 Header("Location: secret_info.php");
 // перенаправляємо на сторінку secret_info.php
 }else echo "неправильний ввід, спробуйте ще раз
<br>";
 }
 print_r($_SESSION); // виводимо усі змінні сесії
?>
```

Тепер, потрапивши на сторінку secret_info.php, та і на будь-яку іншу сторінку сайту, ми зможемо працювати з введеними користувачем логіном і паролем, які зберігатимуться в масиві $_SESSION. Таким чином, якщо змінити код секретної сторіночки (зверніть увагу, ми перейменували її в

secret_info.php) так:

```
<?php
session_start(); // створюємо нову сесію
                  // або відновлюємо поточну
print_r($_SESSION); // виводимо усі змінні сесії
?>
<html>
 <head><title>Secret info</title></head>
 <body>
 <p>Тут я хочу ділитися секретами
     з другом Петром.
 </body>
</html>
```

Тоді ми отримаємо у браузері на секретній сторінці наступне:

```
Array ( [login] => pit [passwd] => 123 )
Тут я хочу ділитися секретами
з другом Петром.
```

У результаті отримаємо.список змінних, зареєстрованих на authorize.php і, власне, саму секретну сторінку.

Що це нам дає? Припустимо, хакер хоче прочитати секрети Василя і Петра. І він якось дізнався, як називається секретна сторінка (чи сторінки). Тоді він може спробувати просто ввести її адресу в рядку браузеру, минувши сторінку авторизації (введення пароля). Щоб уникнути такого проникнення в наші таємниці, треба дописати всього пару рядків в код секретних сторінок:

```
<?php
session_start(); // створюємо нову сесію
                 // або відновлюємо поточну
print_r($_SESSION); // виводимо усі змінні сесії
if (!($_SESSION['login']==pit &&
$_SESSION['passwd']==123))
 // перевіряємо правильність пароля-логіна
 Header("Location: authorize.php");
 // якщо помилка, то перенаправляємо на
    // сторінку авторизації
?>
<html>
<head><title>Secret info</title></head>
... // тут розташовується секретна інформація
</html>
```

### 5.2.4 Видалення змінних сесії

Окрім уміння реєструвати змінні сесії (тобто робити їх глобальними упродовж усього сеансу роботи), корисно також уміти видаляти такі змінні і сесію в цілому.

Функція session_unregister(ім´я_змінної) видаляє глобальну змінну з поточної сесії (тобто видаляє її зі списку зареєстрованих змінних). Якщо реєстрація проводилася за допомогою $_SESSION ($HTTP_SESSION_VARS для версії PHP 4.0.6 і більш ранніх), то використовують мовну конструкцію unset(). Вона не повертає ніякого значення, а просто знищує вказані змінні.

Де це може згодитися? Наприклад, для знищення даних про відвідувача (зокрема, логіна і пароля) після його відходу з секретної сторінки. Якщо правильний логін і пароль збережуться і вікно браузеру після відвідування сайту не закрили, то будь-який інший користувач цього комп'ютера зможе прочитати закриту інформацію.

Приклад. Знищення змінних сесії

У файл secret_info.php додамо рядок для виходу на головну сторінку:

```
<?php
// ... php код
?>
<html>
<head><title>Secret info</title></head>
... // тут розташовується секретна інформація
<a href="index.php">На головну</a>
</html>
```

У Index.php знищимо логін і пароль, введені раніше :

```
<?
session_start();
session_unregister('passwd'); // знищуємо пароль
unset($_SESSION['login']); // знищуємо логін
print_r($_SESSION); // виводимо глобальні змінні сесії
?>
<html>
<head><title>My home page</title></head>
... // домашня сторінка
</html>
```

Тепер, щоб потрапити на секретну сторінку, треба буде знову вводити логін і пароль.

Для того, щоб скинути значення усіх змінних сесії, можна використати функцію session_unset();

Знищити поточну сесію цілком можна командою session_destroy(); Вона не скидає значення глобальних змінних сесії і не видаляє cookies, а знищує усі дані, що асоціюються з поточною сесією.

```
<?
session_start(); // ініціалізували сесію
$test = "Змінна сесії";
$_SESSION['test']= $test; // реєструємо змінну
 // $test. якщо register_globals=on, то можна
     // використати session_register('test');
print_r($_SESSION);// виводимо усі глобальні змінні
echo session_id();// виводимо ідентифікатор сесії
echo "<hr>";
session_unset();// знищуємо глобальні змінні сесії
print_r($_SESSION);
echo session_id();
echo "<hr>";
session_destroy(); // знищуємо сесію
print_r($_SESSION);
echo session_id();
?>
```

В результаті роботи цього скрипта будуть виведені три рядки: в першій - масив з елементом test і його значенням, а також ідентифікатор сесії, в другій - порожній масив і ідентифікатор сесії, в третій - порожній масив. Таким чином, видно, що після знищення сесії знищується і її ідентифікатор, і ми більше не можемо ні реєструвати змінні, ні взагалі виконувати які-небудь дії з сесією.

### 5.3 PHP і Cookies

Cookies - це механізм зберігання даних браузером віддаленого комп'ютера для ідентифікації відвідувачів, що повертаються, і зберігання параметрів веб-сторінок (наприклад, змінних).

Наведемо приклад використання Cookies на конкретному прикладі.

Припустимо, нам треба написати лічильник відвідування сайту. Нам треба знати, яке число відвідувань сайту здійснювалося кожним конкретним відвідувачем.

Це завдання можна вирішити двома способами. Перший з них полягає у веденні обліку IP-адрес користувачів. Для цього потрібна база даних всього з

однієї таблиці, зразкова структура якої така :

| IP-адрес | Число відвідувань |
|---|---|
| 210.124.134.203 | 7 |
| 212.201.78.207 | 14 |
| 83.103.203.73 | 3 |

Коли користувач заходить на сайт, нам треба визначити його IP -адрес, знайти у базі даних інформацію про його відвідування, збільшити лічильник і вивести його у браузер відвідувача. Написати обробник (скрипт) подібної процедури нескладно. Проте при використанні такого методу у нас з'являються проблеми наступного характеру :

Для кожного IP -адресу треба вести облік в одній таблиці, яка може бути дуже великою. А з цього виходить, що ми нераціонально використовуємо процесорний час і дисковий простір;

У більшості домашніх користувачів IP -адреса є динамічною. Тобто, сьогодні адреса 212.218.78.124, а завтра - 212.218.78.137. Таким чином, велика вірогідність ідентифікувати одного користувача кілька разів.

Можна використати другий спосіб, який набагато легше в реалізації і ефективніший. Ми встановлюємо в Cookie змінну, яка зберігатиметься на диску видаленого користувача. Ця змінна і зберігатиме інформацію про відвідування. Вона зчитуватиметься скриптом при зверненні відвідувача до сервера. Вигода такого методу ідентифікації очевидна. По-перше, нам не треба зберігати безліч непотрібної інформації про IP -адресах. По-друге, нас не цікавлять динамічні IP -адреса, оскільки дані про свої відвідування зберігаються конкретно у кожного відвідувача сайту.

Тепер зрозуміло, для чого ми можемо використати Cookie - для зберігання невеликої за об'ємом інформації у клієнта (відвідувача) сайту, наприклад: налаштування сайту (колір фону сторінок, мова, оформлення таблиць и.т.д.), а також іншої інформації.

Файли cookies є звичайними текстовими файлами, які зберігаються на диску у відвідувачів сайтів. Файли cookies і містять ту інформацію, яка була в них написана сервером.

### 5.3.1 Програмування cookies

Приступимо до програмування cookies.

Для установки Cookies використовується функція SetCookie(). Для цієї функції можна вказати шість параметрів, один з яких є обов'язковим:

name - задає ім'я (рядків), закріплене за Cookie;

value - визначає значення змінної (рядок);

expire - час "життя" змінної (ціле число). Якщо цей параметр не вказати, то cookie "житимуть" до кінця сесії, тобто до закриття браузеру. Якщо час вказаний, то, коли воно настане, cookie самознищуватиметься.

path - шлях до cookie (рядок);

domain - домен (рядок). В якості значення встановлюється ім'я хоста, з якого Cookie був встановлений;

secure - передача cookie через захищене HTTPS –з'єднання.

Зазвичай використовуються тільки три перші параметри.

Приклад установки cookies :

```
<?php
// Встановлюємо Cookie до кінця сесії:
SetCookie("Test","Value");
// Встановлюємо Cookie на одну годину
//після установки:
SetCookie("My_Cookie","Value",time()+3600);
?>
```

При використанні cookies необхідно мати на увазі, що cookies повинні встановлюватися до першого виводу інформації у браузер (наприклад, оперетором echo або виведенням якої-небудь функції). Тому бажано встановлювати cookies на самому початку скрипта. Cookies встановлюються за допомогою певного заголовка сервера, а якщо скрипт виводить що-небудь, то це означає, що починається тіло документу. В результаті Cookies не будуть встановлені і може бути виведене попередження. Для перевірки успішності установки cookies можна використати такий метод:

```
<?php
// Встановлюємо Cookie до кінця сесії:
// У разі успішної установки Cookie,
// функція SetCookie повертає TRUE:
if(SetCookie("Test","Value"))
    echo "<h3>Cookies успішно встановлені</h3>";
?>
```

Функція SetCookie() повертає TRUE у разі успішної установки Cookie. У разі, якщо Cookie встановити не вдається SetCookie() поверне FALSE і можливо, попередження (залежить від налаштувань PHP). Приклад невдалої

установки cookie :

```
<?php
// Cookies встановити не вдасться, оскільки
// перед відправкою заголовка Cookie ми виводимо
// у браузер рядок 'Hello ':
echo "Hello";
// Функція SetCookie поверне FALSE:
if(SetCookie("Test","Value"))
    echo "<h3>Cookie успішно встановлено!</h3>";
else echo "<h3>Cookie встановити не вдалося!</h3>";
?>
```

Cookie встановити не вдалося, оскільки перед посилкою заголовка cookie ми вивели у браузер рядок "Hello".

### 5.3.2 Читання значень cookies

Отримати доступ до cookies і їх значенням досить просто. Вони зберігаються в суперглобальних масивах і $_COOKIE і $HTTP_COOKIE_VARS.

Доступ до значень здійснюється по імені встановлених Cookies, наприклад:

```
echo $_COOKIE['my_cookie'];
// Виводить значення встановленою
// Cookie 'My_Cookie'
```

Приклад установки cookie і подальшого його читання :

```
<?php
// Встановлюємо Cookie 'test' зі значенням
// 'Hello' на одну годину:
setcookie("test", "Hello", time()+3600);
// При наступному запиті скрипта виводить 'Hello ':
echo @$_COOKIE['test'];
?>
```

У розглянутому прикладі при першому зверненні до скрипта встановлюється cookie "test" зо значенням "hello". При повторному зверненні до скрипта буде виведено значення cookie "test", тобто рядок "Hello".

При читанні значень cookies звертайте увагу на перевірку існування cookies, наприклад, використовуючи оператор isset(). Або шляхом

пригнічення виведення помилок опереатором @

А ось приклад, як побудувати лічильник числа завантажень сторінки за допомогою cookies:

```
<?php
// Перевіряємо, чи був вже встановлений
// Cookie 'Mortal'. Якщо так, то читаємо його
// значення і збільшуємо значення
// лічильника звернень до сторінки:
if (isset($_COOKIE['Mortal']))
 $cnt=$_COOKIE['Mortal']+1;
else $cnt=0;
// Встановлюємо Cookie 'Mortal' із значенням
// лічильника р часом "життя" до 18/07/29,
// Тобто на дуже довгий час:
setcookie("Mortal",$cnt, 0x6FFFFFFF);
// Виводить число відвідувань цієї сторінки :
echo   "<p>Ви   відвідували   цю   сторінку   <b>".
    @$_COOKIE['Mortal']".</b> раз</p>";
?>
```

### 5.3.2 Видалення cookies

Іноді виникає необхідність видалення cookies. Зробити це нескладно, необхідно лише знову встановити cookie з ідентичним ім'ям і порожнім параметром. Наприклад:

```
<?php
// Видаляємо Cookie 'Test ':
SetCookie("Test","");
?>
```

### 5.3.4 Встановлення масиву cookies і його читання

Ми можемо встановити масив cookies, використовуючи квадратні дужки в іменах cookies[], а потім прочитати масив cookies і значення цього масиву:

```
<?php
// Встановлюємо масив Cookies :
setcookie("cookie[1],"Перший");
setcookie("cookie[2],"Другий");
setcookie("cookie[3],"Третій");
// Після перезавантаження сторінки ми відобразимо
// Склад масиву Cookies 'cookie ':
if (isset($_COOKIE['cookie'])){
```

```
    foreach ($_COOKIE['cookie'] as $name => $value)
    {
 echo "$name: $value <br>";
 }
}
?>
```

Переваги використання cookies незаперечні. Проте існують і деякі проблеми їх використання. Перша з них полягає в тому, що відвідувач може блокувати прийом cookies браузером або просто видалити усе cookies або їх частина. Таким чином, ми можемо мати деякі проблеми в ідентифікації таких відвідувачів.

### 5.4 Безпека при роботі із сеансами

Взагалі кажучи, слід розуміти, що використання механізму сесій не гарантує повної безпеки системи. Для цього треба вживати додаткові заходи. Звернемо увагу на проблеми з безпекою, які можуть виникнути при роботі з сесіями і, зокрема, з тими програмами, що ми написали.

По-перше, небезпечно передавати туди-сюди пароль, його можуть перехопити. Крім того, ми зареєстрували його як глобальну змінну сесії, значить, він зберігся в cookies на комп'ютері-клієнті. Це теж погано. І взагалі, паролі і логіни по-хорошому повинні зберігатися у базі даних. Нехай інформація про користувачів зберігається у базі даних "test" (у таблиці "users"), а ми маємо до неї доступ під логіном my_user і паролем my_passwd.

По-друге, що робити, якщо хтось написав скрипт підбору пароля для секретної сторінки? В цьому випадку на сторінку авторизації багато разів повинен стукатися якийсь сторонній скрипт. Тому треба просто перевіряти, чи з нашого сайту прийшов запит на авторизацію, і якщо ні, то не пускати його далі. Адреса сторінки, з якою поступив запит, можна отримати за допомогою глобальної змінної $_SERVER['HTTP_REFERER']). Хоча, звичайно, якщо за злом сайту взялися серйозно, то значення цієї змінної теж підмінять (наприклад, за допомогою того ж PHP). Проте перевірку її значення можна вважати одним з найважливіших кроків на шляху до забезпечення безпеки свого сайту.

```
<?
session_start();
 // створюємо нову сесію або відновлюємо поточну
$conn   =   mysql_connect("localhost",   "my_user",
```

```
    "my_passwd"); // встановлюємо з'єднання з БД
mysql_select_db("test"); // вибираємо робочу БД
                  // де знаходяться наші скрипти
$SERVER_ROOT="http://localhost/tasks/sessions/";
/* за допомогою регулярного виразу ^$SERVER_ROOT
і функції eregi (eregi - збіг з регулярним вираженням
без урахування регістра символів) перевіряємо, чи
починається адреси скрипта, що посилається, тобто
рядок    $_SERVER['HTTP_REFERER'])    із    рядка
$SERVER_ROOT (як у нас) */

if(eregi("^$SERVER_ROOT",$_SERVER['HTTP_REFERER']))
{
// якщо так, то робимо майже те ж, що і раніше,
// пароль реєструвати не будемо
if (!isset($_POST['go'])){
 echo "<form method=POST >
 Login: <input type=text name=login>
 Password: <input type=password name=passwd>
 <input type=submit name=go value=Go>
 </form>";
}else {
/* запит до бази даних : вибираємо з таблиці users
login, який співпадає з переданим за запитом, причому
пароль у нього теж повинен співпасти з введеним
користувачем. Якщо цього немає, то вважаємо, що логін
і пароль введені невірно */
$sql   =   "SELECT   login   FROM   users   WHERE
login=".$_POST['login']."AND
passwd=".$_POST['passwd'].";
    // відправляємо запит до БД
$q = mysql_query($sql,$conn);
    // число рядків у відповіді на запит
$n = mysql_num_rows($q);
if (!$n==0){
 $_SESSION['user_login']=$_POST['login'];
 // реєструємо змінну login
 Header("Location: secret_info.php");
 // перенаправляємо на сторінку secret_info.php
 }else
    echo "неправильний ввід, спробуйте ще раз<br>";
}
print_r($_SESSION); // виводимо усі змінні сесії
}
?>
```

Начебто перші дві проблеми розв'язані. Але є ще одна. Що робити, якщо

хакер просто допише в рядок запиту значення якої-небудь глобальної змінної (наприклад, логіна)? Взагалі це можливо, тільки якщо register_globals=On. Просто інакше ми використовуємо для роботи з глобальними змінними масив $_SESSION і з ним такі фокуси не проходять. Все ж спробуємо розв'язати і цю проблему. Для цього треба очистити рядок запиту перед тим, як порівнювати значення параметрів. Тобто спочатку скинемо значення $user_login. Потім цю змінну треба знову зареєструвати, але не як нову, а як вже існуючу. Для цього знак долара при реєстрації НЕ опускається. Ось що вийшло:

```
<?php
unset($user_login); // знищуємо змінну
// створюємо нову сесію або відновлюємо поточну
session_start();
// реєструємо змінну як вже існуючу
session_register($user_login);
if (!($user_login==pit)) // перевіряємо логін
    // якщо помилка, то перенаправляємо
    //на сторінку авторизації
 Header("Location: authorize.php");
?>
<html>
<head><title>Secret info</title></head>
... // тут розташовується секретна інформація
</html>
```

Отже, ми познайомилися з сесіями і основними способами роботи з ними, проблемами, що виникають при їх використанні, і можливими рішеннями цих проблем.

**Контрольні питання до теми 5**

1. В чому полягає потреба авторизації?
2. Які є способи авторизації і які можливі проблеми при їх використанні?
3. Як працює мехнізм сесій?
4. Які параметри визначають сесію?
5. Які функції використовуються при роботі із сесіями в PHP?
6. Для чого існує масив масиву $_SESSION?
7. Що таке Cookies?
8. Як задаються Cookies в PHP?
9. Як прочитати значення Cookies?
10. Які заходи безпеки потрібно вживати при роботі із сесіями?
11.

# Тема 6. Взаємодія PHP та баз даних

## 6.1 PHP та MySQL

MySQL - це одна з найпоширеніших систем керування базами даних (СКБД) з відкритим кодом, яка була створена як альтернатива комерційним системам. Вона використовується, в першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування. У дистрибутив PHP входить розширення, що містить вбудовані функції для роботи з базою даних MySQL. У цій темі ми познайомимося з деякими основними функціями для роботи з MySQL, які знадобляться для вирішення завдань побудови web -интерфейсов з метою відображення і наповнення бази даних. Виникає питання, навіщо будувати такі інтерфейси? Для того, щоб вносити інформацію у базу даних і переглядати її вміст могли люди, не знайомі з мовою запитів SQL. При роботі з web - інтерфейсом для додавання інформації у базу даних людині треба просто ввести ці дані в html -форму і відправити їх на сервер, а наш скрипт зробить усе інше. А для перегляду вмісту таблиць досить просто клацнути по посиланню і зайти на потрібну сторінку.

Для наочності будуватимемо ці інтерфейси для деякої таблиці Artifacts, в якій міститься інформація про експонати віртуального музею інформатики. Кожен експонат в колекції Artifacts описується за допомогою наступних характеристик:

- назва (title);
- автор (author);
- опис (description);
- альтернативна назва (alternative);
- зображення (photo).

Назва і альтернативна назва є рядками менш ніж 255 символів завдовжки (тобто мають тип VARCHAR(255)), опис - текстове поле (має тип TEXT), а в полях "автор" і "зображення" містяться ідентифікатори автора з колекції Persons і зображення експоната з колекції Images відповідно.

Отже, у нас є певна таблиця у базі даних. Щоб побудувати інтерфейс для додавання інформації в цю таблицю, треба її структуру (тобто набір її полів) відобразити в html -форму.

Розіб'ємо це завдання на наступні підзадачі:

- встановлення з' єднання з БД ;

- вибір робочої БД ;
- отримання списку полів таблиці ;
- відображення полів в html - форму.

Після цього дані, введені у форму, треба записати у базу даних. Розглянемо усі ці завдання по порядку.

### 6.2 Етапи взаємодії PHP та БД

Отже, перше, що треба зробити, - це встановити з' єднання з базою даних. Скористаємося функцією mysql_connect.

Синтаксис mysql_connect

```
ресурс mysql_connect ([рядок server
 [, рядок username [, рядок password
[, логічне new_link [, ціле client_flags]]]])
```

Ця функція встановлює з' єднання з сервером MySQL і повертає вказівник на це з'єднання або FALSE у разі невдачі. Для відсутніх параметрів встановлюються наступні значення по замовчуванню:

- server = 'localhost: 3306'
- username = ім 'я користувача власника процесу сервера
- password = порожній пароль

Якщо функція викликається двічі з одними і тими ж параметрами, то нове з'єднання не встановлюється, а повертається посилання на старе з'єднання. Щоб цього уникнути, використовують параметр new_link, який примушує у будь-якому випадку відкрити ще одне з'єднання.

Параметр client_flags - це комбінація наступних констант: MYSQL_CLIENT_COMPRESS (використати протокол стискування), MYSQL_CLIENT_IGNORE_SPACE(дозволяє вставляти пропуски після імен функцій), MYSQL_CLIENT_INTERACTIVE (чекати interactive_timeout секунд - замість wait_timeout - до закриття з' єднання).

Параметр new_link з'явився в PHP 4.2.0, а параметр client_flags - в PHP 4.3.0.

З' єднання з сервером закривається при завершенні виконання скрипта, якщо воно до цього не було закрите за допомогою функції mysql_close().

Отже, встановлюємо з' єднання з базою даних на локальному сервері для користувача nina з паролем "123":

```
<?
$conn = mysql_connect("localhost", "nina", "123")
```

```
or die("Неможливо встановити
    з'єднання: ". mysql_error());
echo "З'єднання встановлене";
mysql_close($conn);
?>
```

Дія mysql_connect рівносильно команді:

```
shell>mysql -u nina -p 123
```

Після установки з'єднання треба вибрати базу даних, з якою працюватимемо. Наші дані зберігаються у базі даних book. У MySQL вибір бази даних здійснюється за допомогою команди use :

```
mysql>use book;
```

У PHP для цього існує функція mysql_select_db.

Синтаксис mysql_select_db :

```
логічне mysql_select_db ( рядок database_name
 [, ресурс link_identifier])
```

Ця функція повертає TRUE у разі успішного вибору бази даних і FALSE - інакше.

Зробимо базу даних book робочою:

```
<?
$conn = mysql_connect(
 "localhost", "nina", "123")
or die("Неможливо встановити
 з' єднання: ". mysql_error());
echo "З' єднання встановлено";
mysql_select_db("book");
?>
```

Тепер можна зайнятися власне рішенням задачі. Як отримати список полів таблиці? Дуже просто. У PHP і на цей випадок є своя команда - mysql_list_fields.

Синтаксис mysql_list_fields

```
ресурс mysql_list_fields ( рядок database_name,
 рядок table_name    [, ресурс link_identifier])
```

Ця функція повертає список полів в таблиці table_name у базі даних database_name. Виходить, що вибирати базу даних нам було необов'язково, але це згодиться пізніше. Як можна помітити, результат роботи цієї функції - змінна типу ресурс. Тобто це не зовсім те, що ми хотіли отримати. Це посилання, яке можна використати для отримання інформації про поля таблиці, включаючи їх назви, типи і прапори.

Функція mysql_field_name повертає ім'я поля, отриманого в результаті виконання запиту. Функція mysql_field_len повертає довжину поля. Функція mysql_field_type повертає тип поля, а функція mysql_field_flags повертає список прапорів поля, записаних через пропуск. Типи поля можуть бути int, real, string, blob і так далі. Прапори можуть бути not_null, primary_key, unique_key, blob, auto_increment і так далі

Синтаксис у усіх цих команд однаковий:

```
рядок mysql_field_name (ресурс result, ціле field_offset)
рядок mysql_field_type (ресурс result, ціле field_offset)
рядок mysql_field_flags(ресурс result, ціле field_offset)
рядок mysql_field_len (ресурс result, ціле field_offset)
```

Тут result - це ідентифікатор результату запиту (наприклад, запиту, відправленого функціями mysql_list_fields або mysql_query (про неї буде розказано пізніше)), а field_offset - порядковий номер поля в результаті.

Взагалі кажучи, те, що повертають функції типу mysql_list_fields або mysql_query, є таблицею, а точніше, вказівник на неї. Щоб отримати з цієї таблиці конкретні значення, треба задіяти спеціальні функції, які відрядковий читає цю таблицю. До таких функцій і відносяться mysql_field_name і тому подібне. Щоб перебрати усі рядки в таблиці результату виконання запиту, треба знати число рядків в цій таблиці. Команда mysql_num_rows (ресурс result ) повертає число рядків у безлічі результатів result.

А тепер спробуємо отримати список полів таблиці Artifacts (колекція експонатів).

```
<?
$conn = mysql_connect(
 "localhost", "nina", "123")
or die("Неможливо встановити
 з'єднання: ". mysql_error());
echo "З'єднання встановлене";
mysql_select_db("book");
$list_f = mysql_list_fields (
```

```
 "book", "Artifacts",$conn);
 $n = mysql_num_fields($list_f);
for($i=0;$i<$n; $i++){
 $type = mysql_field_type($list_f, $i);
 $name_f = mysql_field_name($list_f,$i);
 $len = mysql_field_len($list_f, $i);
 $flags_str = mysql_field_flags (
 $list_f, $i);
echo "<br>Ім'я поля : ". $name_f;
echo "<br>Тип поля : ". $type;
echo "<br>Довжина поля : ". $len;
echo "<br>Рядок прапорів поля : ".
 $flags_str . "<hr>";
}
?>
```

В результаті повинно вийти приблизно ось що (якщо в таблиці всього два поля, звичайно) :

Ім'я поля : id

Тип поля : int

Довжина поля : 11

Рядок прапорів поля :

not_null primary_key auto_increment

Ім'я поля : title

Тип поля : string

Довжина поля : 255

Рядок прапорів поля :

Тепер підкоректуємо попередній приклад. Не просто виводитимемо інформацію про поле, а відображати його у відповідний елемент html -форми. Так, елементи типу BLOB переведемо в textarea (помітимо, що поле description, яке ми створювали з типом TEXT, відображається якщо має тип BLOB ), числа і рядки відобразимо в текстові рядки введення <input type=text>, а елемент, що має мітку автоінкремента, взагалі не відображатимемо, оскільки його значення встановлюється автоматично.

Усе це вирішується досить просто, за винятком виділення зі списку прапорів прапора auto_increment. Для цього треба скористатися функцією explode.

Синтаксис explode :

```
масив explode(рядок separator,рядок string[,int limit])
```

Ця функція розбиває рядок string на частини за допомогою роздільника separator і повертає масив отриманих рядків.

У нашому випадку в якості роздільника треба узяти пропуск " ", а в якості початкового рядка для розбиття - рядок прапорів поля.

Отже, створимо форму для введення даних в таблицю Artifacts:

```
<?
$conn=mysql_connect("localhost", "nina", "123");
// встановлюємо з' єднання
$database = "book";
$table_name = "Artifacts";
// вибираємо базу даних для роботи
mysql_select_db($database);
// отримуємо список полів в таблиці
$list_f = mysql_list_fields($database,$table_name);
// число рядків в результаті попереднього
//запиту (тобто скільки всього полів в таблиці
$n = mysql_num_fields($list_f);
// створюємо форму для введення даних
echo "<form method=post action=insert.php>";
echo "  <TABLE BORDER=0 CELLSPACING=0 width=50%
       > <tr> <TD BGCOLOR='#005533' align=center>
<font color='#FFFFFF'> <b> Add new row in $table_name
</b></font></td></tr><tr><td></td></tr></TABLE>";
echo  "<table  border=0  CELLSPACING=1  cellpadding=0
       width=50% >";
// для кожного поля отримуємо його ім'я, тип,
// довжину і прапорці
for($i=0;$i<$n; $i++){
$type = mysql_field_type($list_f, $i);
$name_f = mysql_field_name ($list_f,$i);
$len = mysql_field_len($list_f, $i);
$flags_str = mysql_field_flags ($list_f, $i);
// з рядка прапорців робимо масив, де кожен елемент
// масиву - прапор поля
$flags = explode(" ", flags_str);
 foreach ($flags as $f){
 if ($f == 'auto_increment') $key = $name_f;
 }
/* для кожного поля, що не є автоінкрементом, в
залежності від його типу виводимо
відповідний елемент форми */
if ($key <> $name_f){
echo "<tr><td align=right bgcolor='#C2E3B6'><font
size=2><b> ". $name_f ".</b></font></td>";
```

```
switch ($type){
 case "string":
 $w = $len/5;
 echo "<td><input type=text name=\"$name_f\"
 size = $w ></td>";
 break;
 case "int":
 $w = $len/4;
 echo "<td><input type=text name=\"$name_f\"
 size = $w ></td>";
 break;
 case "blob":
 echo "<td><textarea rows=6 cols=60
             name=\"$name_f\"> </textarea></td>";
 break;
 }
 }
 echo "</tr>";
}
echo "</table>";
echo "<input type=submit name='add' value='Add'>";
echo "</form>";
?>
```

Отже, форма створена. Тепер треба зробити найголовніше - відправити дані з цієї форми в нашу базу даних. Як ви вже знаєте, для того, щоб записати дані в таблицю, використовується команда INSERT мови SQL. Наприклад:

```
mysql> INSERT INTO Artifacts
 SET title='Петренко';
```

Виникає питання, як можна скористатися такою командою (чи будь-якою іншою командою SQL) в PHP скрипті. Для цього існує функція mysql_query().

Синтаксис mysql_query

```
ресурс mysql_query (рядок query
    [,ресурс link_identifier])
```

mysql_query() посилає SQL -запрос активній базі даних MySQL сервера, який визначається за допомогою вказівника link_identifier (це посилання на якесь з'єднання з сервером MySQL ). Якщо параметр link_identifier опущений,

використовується останнє відкрите з'єднання. Якщо відкриті з'єднання відсутні, функція намагається з'єднатися з СУБД, аналогічно функції mysql_connect() без параметрів. Результат запиту буферизируется.

Зауваження: рядок запиту НЕ повинен закінчуватися крапкою з комою.

Тільки для запитів SELECT, SHOW, EXPLAIN, DESCRIBE, mysql_query() повертає вказівник на результат запиту, або FALSE, якщо запит не був виконаний. У інших випадках mysql_query() повертає TRUE, якщо запит виконаний успішно, і FALSE - у разі помилки. Значення, не рівне FALSE, говорить про те, що запит був виконаний успішно. Воно не говорить про кількість рядів, що торкнулися або повернених. Цілком можлива ситуація, коли успішний запит не торкнеться жодного ряду. mysql_query() також вважається помилковим і поверне FALSE, якщо у користувача недостатньо прав для роботи з вказаною в запиті таблицею.

Отже, тепер ми знаємо, як відправити запит на вставку рядків у базу даних. Помітимо, що в попередньому прикладі елементи форми ми назвали іменами полів таблиці. Тому вони будуть доступні в скрипті insert.php, оброблювальному ці форми, як змінні виду $_POST['ім'я_поля']:

```
<?
$conn=mysql_connect("localhost", "nina", "123");
$database = "book";
$table_name = "Artifacts";
mysql_select_db($database);
$list_f = mysql_list_fields($database,$table_name);
$n = mysql_num_fields($list_f);
// складемо один запитий відразу для усіх полів
// таблиці починаємо створювати запит,
$sql = "INSERT INTO $table_name SET ";
// перебираємо усі поля таблиці
for($i=0;$i<$n; $i++){
 // обчислюємо ім 'я поля
 $name_f = mysql_field_name($list_f,$i);
 // обчислюємо значення поля
 $value = $_POST[$name_f];
 $j = $i + 1;
 // дописуємо в рядок $sql пару ім'я=значення
 $sql = $sql.$name_f".='$value'";
 // якщо поле не останнє в списку, то ставимо кому
 if($j <> $n) $sql = $sql"., "; }
 // перш ніж записувати щось у базу
 // можна подивитися, який запитий вийшов
 //echo $sql;
```

```
 // відправляємо запит
 $result = mysql_query($sql,$conn);
 // виводимо повідомлення чи успішно
 // виконаний запит
 if (!$result) echo " Can't add ($table_name) ";
 else echo "Success!<br>";
?>
```

Отже, завдання додавання даних за допомогою web -інтерфейсу ми вирішили. Проте тут є одна тонкість. При рішенні ми не враховували той факт, що значення деяких полів ( author, photo ) повинні братися з інших таблиць ( Persons, Images ). Оскільки MySQL із зовнішніми ключами не працює (вже працює - прим. експерта), цей момент залишається на совісті розробників системи, тобто на нашій совісті. Треба дописати програму так, щоб була можливість вводити в такі поля правильні значення. Але ми робити цього не будемо, оскільки завдання лекції полягає в тому, щоб познайомити читача з елементами технології, а не в тому, щоб створити працюючу систему. Крім того, наявних у читача знань цілком достатньо, щоб розв'язати цю проблему самостійно. Ми ж звернемося до іншого завдання - відображення даних, що зберігаються у базі цих СУБД MySQL.

Щоб відобразити якісь дані у браузері за допомогою PHP, треба спочатку отримати ці дані у вигляді змінних PHP. При роботі з MySQL без посередника (такого, як PHP) вибірка даних робиться за допомогою команди SELECT мови SQL :

```
mysql> SELECT * FROM Artifacts;
```

У попередній главі ми говорили, що будь-який запит, у тому числі і на вибірку, можна відправити на сервер за допомогою функції mysql_query(); Там у нас стояло трохи інше завдання - отримати дані з форми і відправити їх за допомогою запиту на вставку у базу даних. Результатом роботи mysql_query() там могло бути тільки один з виразів, TRUE або FALSE. Тепер же вимагається відправити запит на вибір усіх полів, а результат відобразити у браузері. І тут результат - це ціла таблиця значень, а точніше, вказівник на цю таблицю. Так що потрібні якісь аналоги функції mysql_field_name(), тільки щоб вони витягали з результату запиту не ім'я, а значення поля. Таких функцій в PHP декілька. Найбільш популярні - mysql_result() і mysql_fetch_array().

Синтаксис mysql_result:

```
змішане mysql_result (ресурс result,
 ціле row [, змішане field])
```

mysql_result() повертає значення одного осередку результату запиту. Аргумент field може бути порядковим номером поля в результаті, ім'ям поля або ім'ям поля з ім'ям таблиці через точку tablename.fieldname. Якщо для імені поля в запиті застосовувався аліас ('select foo as bar from...'), використайте його замість реального імені поля.

Працюючи з великими результатами запитів, слід задіяти одну з функцій, оброблювальних відразу цілий ряд результату (наприклад, mysql_fetch_row(), mysql_fetch_array() і так далі). Оскільки ці функції повертають значення декількох осередків відразу, вони НАБАГАТО швидше mysql_result(). Крім того, треба врахувати, що вказівка чисельного зміщення (номери поля) працює набагато швидше, ніж вказівка колонки або колонки і таблиці через точку.

Виклики функції mysql_result() не повинні змішуватися з іншими функціями, працюючими з результатом запиту.

Синтаксис mysql_fetch_array:

```
масив mysql_fetch_array ( ресурс result
 [, ціле result_type])
```

Ця функція обробляє ряд результату запиту, повертаючи масив (асоціативний, чисельний або обоє) з обробленим рядом результату запиту, або FALSE, якщо рядів більше немає.

mysql_fetch_array() - це розширена версія функції mysql_fetch_row(). Окрім зберігання значень в масиві з чисельними індексами, функція повертає значення в масиві з індексами по назві колонок.

Якщо декілька колонок в результаті матимуть однакові назви, буде повернена остання колонка. Щоб отримати доступ до перших, слід використати чисельні індекси масиву або аліаси в запиті. У разі аліасів саме їх ви не зможете використати в іменах колонок, як, наприклад, не зможете використати "photo" в описаному нижче прикладі.

```
select Artifacts.photo as art_image,
 Persons.photo as pers_image
from Artifacts, Persons
```

Важливо помітити, що mysql_fetch_array() працює НЕ повільніше, ніж

mysql_fetch_row(), і надає зручніший доступ до даних.

Другий опціональний аргумент result_type у функції mysql_fetch_array() є константою і може набувати наступних значень: MYSQL_ASSOC, MYSQL_NUM і MYSQL_BOTH. Ця можливість додана в PHP 3.0.7. Значенням по замовчуванню є: MYSQL_BOTH.

Використовуючи MYSQL_BOTH, отримаємо масив, що складається як з асоціативних індексів, так і з чисельних. MYSQL_ASSOC поверне тільки асоціативні відповідності, а MYSQL_NUM - тільки чисельні.

Зауваження: імена полів, повертані цією функцією, регистронезалежні.

Тепер відобразимо дані з Artifacts у вигляді таблиці у браузері:

```
<?
/* спочатку робимо ті ж, що і раніше: встановлюємо
з'єднання, вибираємо базу і отримуємо список і
число полів в таблиці Artifacts */
$conn=mysql_connect("localhost", "nina", "123");
$database = "book";
$table_name = "Artifacts";
mysql_select_db($database);
$list_f = mysql_list_fields($database,$table_name);
$n1 = mysql_num_fields($list_f);
// збережемо імена полів в масиві $names
for($j=0;$j<$n1; $j++){
 $names[] = mysql_field_name ($list_f,$j);
}
// створюємо SQL запит
$sql = "SELECT * FROM $table_name";
// відправляємо запит на сервер
$q = mysql_query($sql,$conn) or die();
// отримуємо число рядків результату
$n = mysql_num_rows($q);
//малюємо HTML - таблицю
echo " <TABLE BORDER=0 CELLSPACING=0 width=90%
 align=center><tr>
<td BGCOLOR='#005533' align=center>
 <font    color='#FFFFFF'><b>$table_name</b></font>
</td>
</tr></TABLE>";
echo "<table cellspacing=0 cellpadding=1 border=1
 width=90% align=center>";
// відображаємо назви полів
echo "<tr>";
foreach ($names as $val){
 echo "<th ALIGN=CENTER BGCOLOR='#C2E3B6'>
```

```
 <font size=2>$val</font></th>";
 }
 // відображаємо значення полів
 echo "</tr>";
 for($i=0;$i<$n; $i++){ // перебираємо усі рядки в
 // результаті запиту на вибірку
 echo "<tr>";
 foreach ($names as $val){ // перебираємо всі
 // імена полів
 $value = mysql_result($q,$i,$val); // отримуємо
 // значення поля
 echo "<td><font size=2>  $value</font>
        </td>"; // виводимо значення поля
 }
 echo "</tr>";
 }
echo "</table>";
```

Зробимо те ж саме за допомогою mysql_fetch_array() :

```
<?
/* ... початок аналогічний, як
і в попередньому прикладі */
// відображаємо значення полів надаємо значення
// поля у вигляді асоціативного масиву
while($row = mysql_fetch_array($q, MYSQL_ASSOC)){
 echo "<tr>";
 foreach ($names as $val){
 echo "<td><font size=2> 
 $row[$val]</font></td>";
 // виводимо значення поля
 }
 echo "</tr>";
 }
echo "</table>";
?>
```

**Контрольні питання до теми 6**

1. Для чого потрібні бази даних і як забезпечується їх взаємодія з PHP?
2. Назвіть етапи взаємодії PHP та БД.
3. Назвіть функцію PHP для встановлення з’єднання з БД.

4. Як отримати список полів таблиці?
5. Як розбити рядок на частини за допомогою роздільника?
6. Як виконати запит до бази даних засобами PHP?
7. Як отримати та інтерпретувати результат запиту до БД?
8.

# Тема 7. Збереження і обробка даних в PHP

## 7.1 Отримання доступу для роботи з файлом

Взагалі кажучи, в PHP не існує функції, призначеної саме для створення файлів. Більшість функцій працюють із вже існуючими файлами у файловій системі сервера. Є декілька функцій, які дозволяють створювати тимчасові файли, або, що те ж саме, файли з унікальним для поточної директорії ім'ям. А ось для того, щоб створити звичайнісінький файл, треба скористатися функцією, яка відкриває локальний або віддалений файл. Називається ця функція fopen(). Що означає "відкриває файл"? Це означає, що fopen зв'язує цей файл з потоком управління програми. Причому зв'язування буває різним залежно від того, що ми хочемо робити з цим файлом : читати його, записувати в нього дані або робити і те і інше. Синтаксис цієї функції такий:

```
resource fopen ( ім'я_файлу, тип_доступу
    [, use_include_path])
```

В результаті роботи ця функція повертає вказівник (типу ресурс) на відкритий нею файл. В якості параметрів цієї функції передаються: ім'я файлу, який треба відкрити, тип доступу до файлу (визначається тим, що ми збираємося робити з ним) і, можливо, параметр, визначальний, чи шукати вказаний файл в include_path. Є ще один опціональний параметр, але про нього ми говорити не будемо, щоб не ускладнювати виклад. Обговоримо детальніше кожного з цих трьох параметрів.

Параметр ім'я_файлу має бути рядком, що містить правильне локальне ім'я файлу або URL -адрес файлу в мережі. Якщо ім'я файлу розпочинається з вказівки протоколу доступу (наприклад, http://... чи ftp://...), то інтерпретатор вважає це ім'я адресою URL і шукає обробник вказаного в URL протоколу. Якщо обробник знайдений, то PHP перевіряє, чи дозволено працювати з об'єктами URL як із звичайними файлами (директива allow_url_fopen ). Якщо allow_url_fopen=off, то функція fopen викликає помилку і генерується попередження. Якщо ім'я файлу не розпочинається з протоколу, то вважається, що вказано ім'я локального файлу. Щоб відкрити локальний файл, треба, щоб PHP мав відповідні права доступу до цього файлу.

Параметр use_include_path, встановлений в значення 1 або TRUE, примушує інтерпретатор шукати вказаний в fopen() файл в include_path. Нагадаємо, що include_path - це директива з файлу налаштувань PHP, задаюча список директорій, в яких можуть знаходитися файли для включення. Окрім

функції fopen() вона використовується функціями include() і require().

Параметр тип_доступу може приймати одне з наступних значень (див табл. 7.1).

Отже, щоб створити файл, треба, як би безглуздо це не звучало, відкрити неіснуючий файл на запис.

```
<?php
$h = fopen("my_file.html", "w");
/* відкриває на запис файл my_file.html
якщо він існує, або створює порожній
файл з таким ім'ям, якщо його ще немає */
$h = fopen("dir/another_file.txt", "w+");
/* відкриває на запис і читання або створює
файл another_file.txt в директорії dir */
$h = fopen("http://www.myserver.ua/file.php ","r");
/* відкриває на читання файл, що знаходиться по
вказаному адресу*/
?>
```

Створюючи файл, треба враховувати, під якою операційною системою ви працюєте, і під якою ОС імовірно цей файл читатиметься. Річ у тому, що різні операційні системи по-різному відмічають кінець рядка. У Unix - подобніх ОС кінець рядка позначається \n, в системах типу Windows - \r\n. Windows пропонує спеціальний прапор t для перекладу символів кінця рядка, у систем типу Unix у свої символи кінця рядка. В протилежність цьому існує прапор b, використовуваний найчастіше для бінарних файлів, завдяки якому такої трансляції не відбувається. Використати ці прапори можна, просто дописавши їх після останнього символу вибраного типу доступу до файлу. Наприклад, відкриваючи файл на читання, замість r слід використати rt, щоб перекодувати усі символи кінця рядка в \r\n. Якщо не використати прапор b при відкритті бінарних файлів, то можуть з'являтися помилки, пов'язані зі зміною утримуваного файлу. З міркувань переносимості програми на різні платформи рекомендується завжди використати прапор b при відкритті файлів за допомогою fopen().

Таблиця 7.1. Значень набувають параметром тип доступу

| Тип доступу | Опис |
|---|---|
| r | Відкриває файл тільки для читання; встановлює вказівник позиції у файлі на початок файлу. |

| | |
|---|---|
| r+ | Відкриває файл для читання і запису; встановлює вказівник файлу на його початок. |
| w | Відкриває файл тільки для запису; встановлює вказівник файлу на його початок і усікає файл до нульової довжини. Якщо файл не існує, то намагається створити його. |
| w+ | Відкриває файл для читання і запису; встановлює вказівник файлу на його початок і усікає файл до нульової довжини. Якщо файл не існує, то намагається створити його. |
| a | Відкриває файл тільки для запису; встановлює вказівник файлу в його кінець. Якщо файл не існує, то намагається створити його. |
| a+ | Відкриває файл для читання і запису; встановлює вказівник файлу в його кінець. Якщо файл не існує, то намагається створити його. |
| x | Створює і відкриває файл тільки для запису; поміщає вказівник файлу на його початок. Якщо файл вже існує, то fopen() повертає false і генерується попередження. Якщо файл не існує, то робиться спроба створити його. Цей тип доступу підтримується починаючи з версії PHP 4.3.2 і працює тільки з локальними файлами. |
| x+ | Створює і відкриває файл для читання і запису; поміщає вказівник файлу на його початок. Якщо файл вже існує, то fopen() повертає false і генерується попередження. Якщо файл не існує, то робиться спроба створити його. Цей тип доступу підтримується, починаючи з версії PHP 4.3.2, і працює тільки з локальними файлами. |

Що відбувається, якщо відкрити або створити файл за допомогою fopen не вдається? В цьому випадку PHP генерує попередження, а функція fopen повертає як результат своєї роботи значення false. Такого роду попередження можна "подавити" (заборонити) за допомогою символу @.

Наприклад, така команда не виведе попередження, навіть якщо відкрити файл не вдалося:

```
$h = @fopen("dir/another_file.txt", "w+");
```

Таким чином, функція fopen() дозволяє створити тільки порожній файл і зробити його доступним для запису. Як же записати дані в цей файл? Як прочитати дані із вже існуючого файлу?

Перш ніж відповісти на ці питання, розглянемо, як закрити встановлене за допомогою fopen() з'єднання.

Після виконання необхідних дій з файлом, будь то читання або запис даних або що-небудь інше, з'єднання, встановлене з цим файлом функцією fopen(), треба закрити. Для цього використовують функцію fclose(). Синтаксис у неї наступний:

```
fclose (вказівник на файл)
```

Ця функція повертає TRUE, якщо з'єднання успішно закрите, і FALSE - інакше. Параметр цієї функції повинен вказувати на файл, успішно відкритий, наприклад, за допомогою функції fopen().

```
<?php
$h = fopen("my_file.html", "w");
fclose($h);
?>
```

Звичайно, якщо не закривати з'єднання з файлом, ніяких помилок виконання скрипта не станеться. Але в цілому для сервера це може мати серйозні наслідки. Наприклад, хакер може скористатися відкритим з'єднанням і записати у файл вірус, не кажучи вже про зайву витрату ресурсів сервера. Так що радимо завжди закривати з'єднання з файлом після виконання необхідних дій.

### 7.2 Запис і читання даних

Для того, щоб записати дані у файл, доступ до якого відкритий функцією fopen(), можна використати функцію fwrite(). Синтаксис у неї наступний:

```
int fwrite ( вказівник на файл, рядок [, довжина])
```

Ця функція записує вміст рядка у файл, на який вказує вказівник на файл. Якщо вказаний додатковий аргумент, то запис закінчується після того, як записана кількість символів, рівна значенню цього аргументу, або коли буде досягнутий кінець рядка.

В результаті своєї роботи функція fwrite() повертає число записаних байтів або false, у разі помилки.

Приклад. Нехай в нашій робочій директорії немає файлу my_file.html. Створимо його і запишемо в нього рядок тексту :

```
<?php
```

```
$h = fopen("my_file.html", "w");
$text = "Цей текст запишемо у файл".;
if (fwrite($h,$text))
 echo "Запис пройшов успішно";
else
 echo "Сталася помилка при записі даних";
fclose($h);
?>
```

В результаті роботи цього скрипта у браузері ми побачимо повідомлення про те, що запис пройшов успішно, а у файлі my_file.html з'явиться рядок "Цей текст запишемо у файл". Якби цей файл існував до того, як ми виконали цей скрипт, усі дані, що знаходяться в нім, були б видалені.

Якщо ж ми напишемо такий скрипт:

```
<?php
$h = fopen("my_file.html", "a");
$add_text = "Додамо текст у файл".;
if(fwrite($h,$add_text, 7))
 echo "Додавання тексту пройшло успішно<br>";
else echo "Сталася помилка при додаванні
     даних<br>";
fclose($h);
?>
```

то до рядка, вже існуючого у файлі my_file.html, додасться ще сім символів з рядка, що міститься в змінній $add_text, тобто слово "Додамо"

Функція fwrite() має псевдонім fputs(), використовуваний так само, що і сама функція. Далі ми розглянемо, які методи читання даних з файлу пропонує мова PHP.

Якщо ми хочемо прочитати дані з існуючого файлу, однієї функції fopen(), як і у випадку із записом даних, недостатньо. Вона лише повертає вказівник на відкритий файл, але не зчитує жодного рядка з цього файлу. Тому для того, щоб прочитати дані з файлу, треба скористатися однією із спеціальних функцій : file, readfile, file_get_contents, fread, fgets і тому подібне

Функція fread здійснює читання даних з файлу. Її можна використати і для читання даних з бінарних файлів, не побоюючись їх ушкодження. Синтаксис fread() такий:

```
string fread (вказівник на файл, довжина);
```

При виклику цієї функції відбувається читання даних довжини (у байтах), визначеної параметром довжина, з файлу, на який вказує вказівник на файл. Параметр вказівник на файл має бути реально існуючою змінною типу ресурс, що містить в собі зв'язок з файлом, відкритий, наприклад, за допомогою функції fopen(). Читання даних відбувається до тих пір, поки не зустрінеться кінець файлу або доки не буде прочитано вказане параметром довжина число байтів.

В результаті роботи функція fread() повертає рядок з прочитаною з файлу інформацією.

Як ви помітили, в цій функції параметр довжина - обов'язковий. Отже, якщо ми хочемо рахувати увесь файл в рядок, треба знати його довжину. PHP може самостійно вичислити довжину вказаного файлу. Для цього треба скористатися функцією filesize(ім'я файлу). У разі помилки ця функція поверне false. На жаль, її можна використати тільки для отримання розміру локальних файлів.

Приклад. Прочитаємо вміст файлу my_file.html:

```
<?php
$h = fopen("my_file.html", "r+");
// відкриваємо файл на запис і читання
$content = fread($h,
 filesize("my_file.html"));
// зчитуємо вміст файлу в рядок
fclose($h); // закриваємо з'єднання з файлом
echo $content;
// виводимо вміст файлу
// на екран браузеру
?>
```

Для того, щоб рахувати вміст бінарного файлу, наприклад зображення, в таких системах, як Windows, рекомендується відкривати файл за допомогою прапора rb або йому подібних, таких, що містять символ b у кінці.

Функція filesize() кешує результати своєї роботи. Якщо змінити вміст файлу my_file.html і знову запустити наведений вище скрипт, то результат його роботи не зміниться. Більше того, якщо запустити скрипт, що зчитує дані з цього файлу за допомогою іншої функції (наприклад, fgets ), то результат може виявитися таким, начебто файл не змінився. Щоб цього уникнути, треба очистити статичний кеш, додавши в код програми команду clearstatcache();

За допомогою функції fgets() можна прочитати з файлу рядок тексту. Синтаксис цієї функції практично такий же, як і у fread(), за винятком того, що

довжину прочитуваного рядка вказувати необов'язково:

```
string fgets ( вказівник на файл, [ довжина])
```

В результаті роботи функція fgets() повертає рядок завдовжки ( довжина-1 ) байт з файлу, на який вказує вказівник на файл. Читання закінчується, якщо прочитано ( довжина-1 ) символів або зустрівся символ перекладу рядка або кінець файлу. Нагадаємо, що в PHP один символ - це один байт. Якщо довжина прочитуваного рядка не вказана (ця можливість з'явилася починаючи з PHP 4.2.0), то зчитується 1 Кбайт (1024 байт) тексту або, що те ж саме, 1024 символи. Починаючи з версії PHP 4.3, якщо параметр довжина не заданий, зчитується рядок цілком. У разі помилки функція fgets() повертає false. Для версій PHP починаючи з 4.3 ця функція безпечна для двійкових файлів.

```
<?php
$h = fopen("my_file.html", "r+");
$content = fgets($h, 2);
// читає перший символ з
// першого рядка файлу my_file.html
fclose($h);
echo $content;
?>
```

Обидві функції, fread() і fgets(), припиняють зчитування даних з файлу, якщо зустрічають кінець файлу. У PHP є спеціальна функція, перевіряюча, чи дивиться вказівник позиції файлу на кінець файлу. Це булева функція feof(), в якості параметра якої передається вказівник на з'єднання з файлом.

Наприклад, ось так можна рахувати усі рядки файлу my_file.html:

```
<?php
$h = fopen("my_file.html", "r");
while (!feof ($h)){
 $content = fgets($h);
 echo $content"<br>";
}
fclose($h);
?>
```

Існує різновид функції fgets() - функція fgetss(). Вона теж дозволяє зчитувати рядок з вказаного файлу, але при цьому видаляє з нього усі html - теги, що зустрілися, за виключенням, можливо, деяких. Синтаксис fgetss()

такий:

```
string fgetss(вказівник на файл,
              довжина [, допустимі теги])
```

Зверніть увагу, що тут аргумент довжина обов'язковий.

Приклад. Нехай у нас є файл my_file.html наступного змісту :

```
<h1>Без зусиль не виймеш і рибку із ставка.</h1>
<b>Тихіше їдеш - далі будеш</b> У семи няньок<i> дитя без ока</i>.
```

Виведемо на екран усі рядки файлу my_file.html, видаливши з них усі теги, окрім <b> і <i>:

```
<?php
$h = fopen("my_file.html", "r");
while (!feof ($h)){
 $content = fgetss($h, 1024'<b><i>');
 echo $content"<br>";
}
fclose($h);
?>
```

В результаті роботи цього скрипта отримаємо:

Без зусиль не виймеш і рибку із ставка. Тихіше їдеш - далі будеш У семи няньок дитя без ока.

Природно, якщо можна зчитувати інформацію з файлу відрядковий, то можна зчитувати її і посимвольний. Для цього призначена функція fgetc(). Легко здогадатися, що синтаксис у неї наступний:

```
string fgetc ( вказівник на файл )
```

Ця функція повертає символ з файлу, на який посилається вказівник на файл, і значення, що обчислюється як FALSE, якщо зустрінутий кінець рядка.

Ось так, наприклад, можна рахувати файл по одному символу:

```
<?php
$h = fopen("my_file.html", "r");
while (!feof ($h)){
 $content = fgetc($h);
 echo $content"<br>";
```

```
}
fclose($h);
?>
```

Насправді для того, щоб прочитати вміст файлу, відкривати з'єднання з ним за допомогою функції fopen() зовсім не обов'язково. У PHP є функції, які дозволяють робити це, використовуючи лише ім'я файлу. Це функції readfile( ), file( ) і file_get_contents( ). Розглянемо кожну з них детальніше.

Функція readfile() зчитує файл, ім'я якого передане їй в якості параметра ім'я_файлу, і виводить його вміст на екран. Якщо додатковий аргумент use_include_path має значення TRUE, то пошук файлу із заданим ім'ям здійснюється і по директоріях, що входять в include_path.

Синтаксис:

```
int readfile ( ім'я_файлу [, use_include_path])
```

У програму ця функція повертає число прочитаних байтів (символів) файлу, а у разі помилки - FALSE. Повідомлення про помилку в цій функції можна подавити оператором @.

Приклад. Наступний скрипт виведе на екран вміст файлу my_file1.html і розмір цього файлу, якщо він існує. Інакше виведеться наше повідомлення про помилку - рядок "Error in readfile".

```
<?php
$n = @readfile ("my_file1.html");
/* виводить на екран вміст файлу і
записує його розмір в змінну $n */
if (!$n) echo "Error in readfile";
/* якщо функція readfile() виконалася
з помилкою, то $n=false і виводимо
повідомлення про помилку */
else echo $n;
 // якщо помилки не було, то виводимо число
 // прочитаних символів
?>
```

За допомогою функції readfile() можна читати вміст видалених файлів, вказуючи їх URL -адрес в якості імені файлу, якщо ця опція не відключена в налаштуваннях сервера.

Відразу ж виводити вміст файлу на екран не завжди зручно. Іноді треба записати інформацію з файлу в змінну, щоб надалі виконати з нею які-небудь

дії. Для цього можна використати функцію file() або file_get_contents().

Функція file() призначена для прочитування інформації з файлу в змінну типу масив. Синтаксис у неї такий же, як і у функції readfile(), за винятком того, що в результаті роботи вона повертає масив:

```
array file ( string $filename [, int $flags = 0
        [, resource $context ]] )
```

Що за масив повертає ця функція? Кожен елемент цього масиву є рядком у файлі, інформацію з якого ми зчитуємо (його ім'я задане аргументом ім'я_файлу ). Символ нового рядка теж включається в кожного з елементів масиву. У разі помилки функція file(), як і усе вже розглянуті, повертає false. Додатковий аргумент use_include_path знову ж таки визначає, шукати або ні цей файл в директоріях include_path. Відкривати видалені файли за допомогою цієї функції теж можна, якщо не заборонено сервером. Починаючи з PHP 4.3 робота з бінарними файлами за допомогою цієї функції стала безпечною.

В якості необов'язкового параметра flags може можна вказати одну або більше наступних констант:

FILE_USE_INCLUDE_PATH - Шукає файл в include_path.

FILE_IGNORE_NEW_LINES - Не додавати новий рядок до кінця кожного елементу масиву

FILE_SKIP_EMPTY_LINES - Пропускати порожні рядки

$trimmed = file('somefile.txt ', FILE_IGNORE_NEW_LINES | FILE_SKIP_EMPTY_LINES);

Наприклад, у нас є файл my_file.html наступного змісту :

```
<h1>Без зусиль не виймеш
 і рибку із ставка.</h1>
<b>Тихіше їдеш - далі будеш</b>
```

Прочитаємо його вміст за допомогою функції file() :

```
<?php
$arr = file ("my_file.html");
foreach($arr as $i => $a) echo $i,":
 htmlspecialchars($a) "<br>";
?>
```

В результаті на екран буде виведено наступне повідомлення:

```
0: <h1>Без зусиль не виймеш
 і рибку із ставка.</h1>
1: <b>Тихіше їдеш - далі будеш</b>
```

У версіях PHP починаючи з 4.3 з'явилася можливість зчитувати вміст файлу в рядок. Робиться це за допомогою функції file_get_contents(). Як і дві попередні функції, в якості параметрів вона набуває значення імені файлу і, можливі, вказівка шукати його в директоріях include_path. Для порядку все одно приведемо її синтаксис:

```
string file_get_contents ( string $filename
[, bool $use_include_path = false
[, resource $context [, int $offset = - 1
[, int $maxlen ]]]] )
```

де filename - Ім'я читаного файлу;

use_include_path - Починаючи з версії PHP 5 можна використати константу FILE_USE_INCLUDE_PATH для пошуку файлу в include path.

context - Коректний ресурс контексту, створений за допомогою функції stream_context_create(). Якщо у використанні особливого контексту немає необхідності, можна пропустити цей параметр передавши в нього значення NULL;

offset Зміщення, з якого розпочнеться читання оригінального потоку. Пошук зміщення (offset) не підтримується при роботі з віддаленими файлами. Спроба пошуку зміщення на нелокальних файлах може працювати при невеликих зміщеннях, але цей результат є непередбачуваним, оскільки він працює на потоці, що буферизований;

maxlen - Максимальний розмір читаних даних. По замовчуванню читання здійснюється доки не буде досягнутий кінець файлу. Врахуйте, що цей параметр застосовується і до потоку з фільтрами.

Ця функція абсолютно ідентична функції file(), тільки повертає вона вміст файлу у вигляді рядка. Крім того, вона безпечна для обробки бінарних даних і може зчитувати інформацію з віддалених файлів, якщо це не заборонено налаштуваннями сервера.

```
$section = file_get_contents('./people.txt ', NULL,
          NULL, 20, 14);
$file = file_get_contents('./people.txt ',
      FILE_USE_INCLUDE_PATH);
```

Отже, створювати файл ми навчилися, записувати дані в нього - навчилися, зчитувати дані з файлу - теж навчилися. Але ось питання: а що коли файлу, з яким ми намагаємося виконати усі ці операції, не існує? Чи він недоступний для читання або запису? Очевидно, що у такому разі жодна з вивчених нами функцій працювати не буде і PHP видасть повідомлення про помилку. Щоб відстежувати такого роду помилки, можна використати функції file_exists(), is_writable(), is_readable().

Функція file_exists() перевіряє, чи існує файл або директорія, ім'я якої передане їй в якості аргументу. Якщо директорія або файл у файловій системі сервера існує, то функція повертає TRUE, інакше - FALSE. Результат роботи цієї функції кеші руется. Відповідно очистити кеш можна, як вже відзначалося, за допомогою функції clearstatcache(). Для нелокальних файлів використати функцію file_exists() не можна.

Синтаксис:

```
bool file_exists (ім'я файлу або директорії)
```

Приклад застосування:

```
<?php
$filename = 'c:/users/files/my_file.html';
if (file_exists($filename)){
 print "Файл <b>$filename</b> існує";
} else {
 print "Файл <b>$filename</b>
 НЕ існує";
}
?>
```

Якщо окрім перевірки існування файлу треба дізнатися ще, чи дозволено записувати інформацію в цей файл, слід використати функцію is_writable() або її псевдонім - функцію is_writeable().

Синтаксис:

```
bool is_writable (ім'я файлу або директорії)
```

Ця функція повертає TRUE, якщо файл (чи директорія) існує і доступний для запису. Доступ до файлу здійснюється під тим обліковим записом користувача, під яким працює сервер (найчастіше це користувач nobody або www). Результати роботи функції is_writable кешуються.

Якщо окрім перевірки існування файлу треба дізнатися ще, чи дозволено читати інформацію з нього, треба використати функцію is_readable().

Синтаксис:

```
bool is_readable (ім'я файлу)
```

Ця функція працює подібно до функції is_writable().

```
<?php
$filename = 'c:/users/files/my_file.html';
if (is_readable($filename)){
 print "Файл <b>$filename</b> існує
 і доступний для читання";
} else {
 print "Файл <b>$filename</b>
 НЕ існує або НЕ доступний для читання";
}
?>
```

Останнє, що ми хочемо вивчити з дій над файлами, - це видалення файлів. Для того, щоб видалити файл за допомогою мови PHP, треба скористатися функцією unlink(). Синтаксис цієї функції можна описати таким чином:

```
bool unlink ( ім'я_файлу)
```

Ця функція видаляє файл, що має ім'я ім'я_файлу, повертає TRUE у разі успіху цієї операції і FALSE - у разі помилки. Щоб видалити файл, треба теж мати відповідні права доступу до нього (наприклад, доступу тільки на читання для видалення файлу недостатньо).

```
<?php
$filename = 'c:/users/files/my_file.html';
unlink($filename);
 // видаляємо файл з ім'ям
 // c:/users/files/my_file.html
?>
```

### 7.3 Завантаження файлу на сервер

Тепер вирішимо складніше і часто виникаюче на практиці завдання завантаження файлу на сервер. Перше, що треба зробити, щоб завантажити

файл на сервер, це створити html -форму. Для того, щоб за допомогою цієї форми можна було завантажувати файли, вона повинна містити атрибут enctype в тегу form зі значенням multipart/form - data, а також елемент input типу file.

```
<form enctype="multipart/form-data"
 action="parse.php" method="post">
 <input type="hidden" name="MAX_FILE_SIZE"
 value="30000" />
 Завантажити файл: <input type="file"
 name="myfile" /><br>
 <input type="submit"
 value="Відправити файл" />
</form>
```

Тип вмісту multipart/form-data — це складений тип вмісту, що найчастіше використовується для відправки HTML -форм з бінарними (не-ASCII) даними методом POST протоколу HTTP.

Помітимо, що ми додали у формі приховане поле, яке містить в собі максимальний допустимий розмір завантажуваного файлу у байтах. При спробі завантажити файл, розмір якого більше вказаного в цьому полі значення, буде зафіксована помилка. У браузері створена нами форма виглядатиме як рядок для введення тексту з додатковою кнопкою для вибору файлу з локального диска (рис. 7.1).

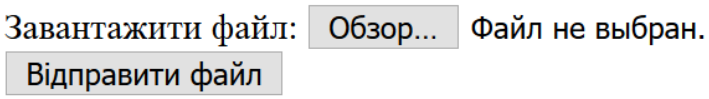



Рисунок 7.1 Приклад форми для завантаження файлу на сервер

Тепер треба написати скрипт, який оброблятиме отриманий файл.

Уся інформація про завантажений на сервер файл міститься в глобальному масиві $_FILES. Цей масив з'явився починаючи з PHP 4.1.0. Якщо включена директива register_globals, то значення переданих змінних доступні просто по їх іменах.

Якщо ми завантажили з комп'ютера-клієнта файл з ім'ям critics.htm розміром 15136 байт, то скрипт з єдиною командою print_r($_FILES); виведе на екран наступне:

```
Array ([myfile] =>
```

```
 Array ([name] => critics.htm
 [type] => text/html
 [tmp_name] => C:\WINDOWS\TEMP\php49F.tmp
 [error] => 0
 [size] => 15136
 )
)
```

Взагалі кажучи, масив $_FILES завжди має наступні елементи:

- $_FILES['myfile']['name'] - ім'я, яке мав файл на машині клієнта.
- $_FILES['myfile']['type'] - mime -тип відправленого файлу, якщо браузер надав цю інформацію. У нашому прикладі це text/html.
- $_FILES['myfile']['size'] - розмір завантаженого файлу у байтах.
- $_FILES['myfile']['tmp_name'] - тимчасове ім'я файлу, під яким він був збережений на сервері.
- $_FILES['myfile']['error'] - код помилки, що з'явилася при завантаженні.

Тут 'myfile' - це ім'я елементу форми, за допомогою якого було проведено завантаження файлу на сервер. Тобто воно може бути іншим, якщо елемент форми назвати інакше. Але ось інші ключі ( name, type і т. д.) залишаються незмінними для будь-якої форми.

Якщо register_globals=On, то доступні також додаткові змінні, такі як $myfile_name, яка еквівалентна $_FILES['myfile']['name'], і тому подібне

Помилок при завантаженні в PHP виділяють п'ять типів і відповідно $_FILES['myfile']['error'] може мати п'ять значень:

0 - помилки не сталося, файл завантажений успішно;

1 - завантажуваний файл перевищує розмір, встановлений директивою upload_max_filesize у файлі налаштувань php.ini;

2 - завантажуваний файл перевищує розмір, встановлений елементом MAX_FILE_SIZE форми html;

3 - файл був завантажений частково;

4 - файл завантажений не був;

По замовчуванню завантажені файли зберігаються в тимчасовій директорії сервера, якщо інша директорія не вказана за допомогою опції upload_tmp_dir у файлі налаштувань php.ini.

Перемістити завантажений файл в потрібну директорію можна за допомогою функції move_uploaded_file().

Функція move_uploaded_file() має наступний синтаксис:

```
bool move_uploaded_file (тимчасове_ім'я_файлу,
```

```
 місце_призначення )
```

Ця функція перевіряє, чи дійсно файл, позначений рядком тимчасове_ім'я_файлу, був завантажений через механізм завантаження HTTP методом POST. Якщо це так, то файл переміщається у файл, заданий параметром місце_призначення (цей параметр містить як шлях до нової директорії для зберігання, так і нове ім'я файлу).

Якщо тимчасове_ім'я_файлу задає неправильний завантажений файл, то ніяких дій виконано не буде, і move_uploaded_file() поверне FALSE. Те ж саме станеться, якщо файл з якихось причин не може бути переміщений. В цьому випадку інтерпретатор виведе відповідне попередження. Якщо файл, заданий параметром місце_призначення, існує, то функція move_uploaded_file() перезапише його.

```
<?
/* У версіях PHP, старших 4.1.0, замість масиву
$_FILES треба використати масив $HTTP_POST_FILES */
$uploaddir = 'c:/uploads/'; // зберігатимемо
// завантажувані файли в цю директорію
$dest = $uploaddir.$_FILES['myfile']['name'];
 // ім'я файлу залишимо незмінним
print "<pre>";
if(move_uploaded_file($_FILES['myfile']
    ['tmp_name'],$dest)){
    /* переміщаємо файл з тимчасової теки
    у вибрану директорію для зберігання */
 print "Файл успішно завантажений <br>";
} else {
 echo "Сталася помилка при завантаженні файлу.
 Деяка налагоджувальна інформація:<br>";
 print_r($_FILES);
}
print "</pre>";
?>
```

**Контрольні питання до теми 7**

1. Назвіть PHP-функцію для отримання доступу до файлу.
2. Які налаштування PHP використосуються при роботі з файлами.
3. Назвіть і прокоментуйте реживи відкривання файлів.
4. Як і навіщо закривати файл.
5. Назвіть функції читання файлів у PHP, яка між ними різниця?

6. Назвіть функції запису у файлів мовою PHP, відмінності між ними?
7. Як перевірити режим, в якому відкритий файл?
8. Назвіть функцію для завантаження даних на сервер.
9. Для чого використовується глобальний масив $_FILES?

# Тема 8. Робота з масивами даних

Мова PHP надає безліч функцій для роботи з масивами даних. Як правило, ці функції вирішують завдання, що найчастіше зустрічаються, пов'язані з обробкою масивів. У цій лекції ми розглянемо деякі з таких функцій і з їх допомогою вирішимо декілька прикладних завдань. Зокрема, будуть розглянуті функції для пошуку елементів в масиві, для сортування елементів масиву, застосування створених користувачем функцій до усіх елементів масиву і розбиття масиву на підмасиви.

Раніше було розглянуто, як можна створити масив даних. Нагадаємо, що масив можна створити двома способами:

1. За допомогою конструкції array:

```
$array_name = array("key1"=>"value1"
 "key2"=>"value2");
```

2. Безпосередньо задаючи значення елементам масиву:

```
$array_name["key1"] = value1;
```

Наприклад, нам треба зберігати список документів, які будуть видалені з бази даних. Природно зберігати його у вигляді масиву, ключем в якому буде ідентифікатор документу (його унікальний номер), а значенням - назва документу. Цей масив можна створити таким чином:

```
<?
$del_items = array("10"=>"Наука і життя"
 "12"=>"Інформатика");
$del_items["13"] = "Програмування на Php";
 // додаємо елемент в масив
?>
```

### 8.1 Операції з масивами

Масив - це тип даних, з даними цього типу мають бути визначені операції. Які ж операції можна робити з масивами? Масиви можна додавати і порівнювати.

Додають масиви за допомогою стандартного оператора " + ". Взагалі кажучи, цю операцію по відношенню до масивів точніше назвати об'єднанням. Якщо у нас є два масиви, $a і $b, то результатом їх додавання (об'єднання) буде

масив $c, що складається з елементів $a, до яких справа дописані елементи масиву $b. Причому, якщо зустрічаються співпадаючі ключі, то в результуючий масив включається елемент з першого масиву, тобто з $a. Таким чином, якщо Додаються масиви в мові PHP, від зміни місць доданків сума міняється.

```
<?
$a=array("i"=>"Інформатика","м"=>"Математика");
$b = array("i"=>"Історія", "м"=>"Біологія",
 "ф"=>"Фізика");
$c = $a + $b;
$d = $b + $a;
print_r($c);
/* отримаємо: Array([i]=>Інформатика
        [м]=>Математика [ф]=>Фізика) */
print_r($d);
/* отримаємо: Array([i]=>Історія
        [м]=>Біологія [ф]=>Фізика) */
?>
```

Порівнювати масиви можна, перевіряючи їх рівність або нерівність або еквівалентність або нееквівалентність. Рівність масивів - це коли співпадають усі пари ключ / значення елементів масивів. Еквівалентність - коли окрім рівності значень і ключів елементів вимагається ще, щоб елементи в обох масивах були записані в одному і тому ж порядку. Рівність значень в PHP позначається символом " == ", а еквівалентність - символом " === ".

```
<?
$a = array("i"=>"Інформатика",
 "м"=>"Математика");
$b = array("м"=>"Математика",
 "i"=>"Інформатика");
if ($a == $b) echo "Масиви рівні і";
 else echo "Масиви НЕ рівні і ";
if ($a === $b) echo " еквівалентні";
 else echo " НЕ еквівалентні";
// отримаємо echo "Масиви рівні і
 НЕ еквівалентні"
?>
```

Далі розглянемо ще одну важливу операцію з масивом - підрахунок кількості його елементів. Для її реалізації в PHP є спеціальна функція count().

Не раз вже ми використали функцію count(), щоб обчислити кількість

елементів масиву. Насправді ця функція обчислює число елементів в змінній взагалі. Якщо застосувати її до будь-якої іншої змінної, вона поверне 1. Виняток становить змінна типу NULL - count(NULL) є 0. Крім того, застосовуючи цю функцію до багатовимірного масиву, щоб отримати число його елементів, треба використати додатковий параметр COUNT_RECURSIVE.

```
<?
$del_items = array("langs" => array(
 "10"=>"Python", "12"=>"Lisp"),
 "other"=>"Інформатика");
echo count($del_items). "<br>";
 // виведе 2
echo count($del_items, COUNT_RECURSIVE);
 // виведе 4
?>
```

Ми не повторюватимемо усе, що було сказано про масиви в попередніх лекціях. У цій лекції ми розглянемо деякі вбудовані функції для роботи з масивами. І почнемо ми з функцій для пошуку значень в масиві.

Функція in_array дозволяє встановити, чи міститься в заданому масиві шукане значення:

```
in_array("шукане значення", "масив",
 ["обмеження_на_тип"]);
```

Якщо третій аргумент заданий як true, то в масиві треба знайти елемент, співпадаючий з шуканим не лише за значенням, але і за типом. Якщо шукане значення - рядок, то порівняння чутливе до регістра.

Наприклад, є масив невивчених нами мов програмування. Ми хочемо дізнатися, чи міститься в цьому масиві мова PHP. Напишемо наступну програму:

```
<?php
$langs = array("Lisp", "Python", "Java"
 "PHP", "Perl");
if (in_array("PHP",$langs, true))
 echo "Потрібно б вивчити PHP<br>";
// виведе повідомлення "Потрібно б вивчити PHP"
if (in_array("php",$langs))
 echo "Потрібно б вивчити php<br>";
// нічого не виведе, оскільки в масиві
```

```
// є рядок "PHP", а не "php"
?>
```

В якості шуканого значення цієї функції може виступати і масив. Правда, ця властивість була додана тільки починаючи з PHP 4.2.0.

Наприклад:

```
<?php
$langs = array("Lisp", "Python", array("PHP",
     "Java"), "Perl");
if (in_array(array("PHP", "Java"),$langs))
     echo "Потрібно б вивчити PHP і Java<br>";
?>
```

Ще одна функція для пошуку значення в масиві – це array_search(). На відміну від in_array() в результаті роботи array_search() повертає значення ключа, якщо елемент знайдений, і false - інакше. А ось синтаксис у цих функцій однаковий:

```
array_search("шукане значення", "масив",
 ["обмеження_на_тип"]);
```

Порівняння рядків чутливе до регістра, а якщо вказаний опціональний аргумент, то порівнюються ще і типи значень. До PHP 4.2.0, якщо шукане значення не було знайдене, ця функція повертала помилку або порожнє значення NULL.

Приклад. Тепер, навпаки, нехай у нас є масив мов програмування, які ми знаємо. Причому ключем кожного елементу є номер, що вказує, яким по рахунку була вивчена ця мова.

```
<?php
$langs = array("", "Lisp", "Python", "Java"
 "PHP", "Perl");
if (!array_search("PHP",$langs))
 echo "Потрібно б вивчити PHP<br>";
else {
 $k = array_search("PHP",$langs);
 echo "PHP я вивчила $k - м";
}
?>
```

В результаті ми отримаємо рядок:

```
PHP я вивчила 4-м
```

Очевидно, що ця функція більше функціональна, ніж in_array, оскільки ми не лише отримуємо інформацію про те, що шуканий елемент в масиві є, але і дізнаємося, де саме в масиві він знаходиться. А що буде, якщо шуканих елементів в масиві декілька? У такому разі функція array_search() поверне ключ першого зі знайдених елементів.

Щоб отримати ключі усіх елементів, треба скористатися функцією array_keys(). Функція array_keys() вибирає усі ключі масиву. Але у неї є додатковий аргумент, за допомогою якого можна отримати список ключів елементів з конкретним значенням. Синтаксис цієї функції такий:

```
array_keys ("масив", ["значення для пошуку"])
```

Функція array_keys() повертає як рядкові, так і числові ключі масиву, організовуючи усі значення у вигляді нового масиву з числовими індексами.

Приклад. Ми записали масив мов, які вивчили. Список був довгим, і деякі мови були записані кілька разів. У нас виникла підозра, що одна з таких мов - Lisp. Давайте це перевіримо:

```
<?php
$langs =
array("Lisp", "Python", "Java", "PHP"
 "Perl", "Lisp");
$lisp_keys = array_keys($langs, "Lisp");
echo "Lisp входить в масив ".
 count($lisp_keys) ". разу:<br>";
foreach ($lisp_keys as $val){
 echo "під номером $val <br>";
}
?>
```

В результаті отримаємо:

```
Lisp входить в масив 2 рази:
під номером 0
під номером 5
```

Функція array_keys(), як і дві попередні, залежить від регістра, тобто елементів LISP в масиві вона не виявить. array_keys() з'явилася тільки в PHP4. У PHP3 для реалізації її функціональності треба придумувати свою функцію.

Якщо є функція для отримання усіх ключів масиву, то можна припустити, що існує і функція для набуття усіх значень масиву. Дійсно, вона існує. Це функція array_values(масив). Усі значення переданого їй масиву записуються в новий масив, проіндексований цілими числами, тобто усі ключі масиву втрачаються, залишаються тільки значення. Але повернемося до нашого прикладу.

Отже, ми з'ясували, що мова Lisp випадково згадана в нашому масиві двічі. Оскільки вивчити одну мову двічі не можна ("учив, але забув" не вважається), то треба якось позбавитися від мов, що повторюються. Зробити це досить просто за допомогою функції array_unique().

Функція array_unique(масив) повертає новий масив, в якому елементи, що повторюються, фігурують в одному екземплярі. Таким чином, замість декількох однакових значень і їх ключів ми маємо одне значення. Який у нього буде ключ? Як з декількох ключів однакових елементів вибирається той, який буде збережений в новому масиві? Відбувається наступне. Усі елементи масиву перетворяться в рядки і сортуються. Потім обробник запам'ятовує перший ключ для кожного значення, а інші ключі ігнорує.

Спробуємо позбавитися від мов, що повторюються, в списку вивчених.

```
<?php
$langs =
array("Lisp", "Java", "Python", "Java"
 "PHP", "Perl", "Lisp");
print_r(array_unique($langs));
?>
```

Отримаємо наступне:

```
Array ([0] => Lisp [1] => Java [2] => Python [3]
 => PHP [4] => Perl )
```

### 8.2 Сортування масивів

Необхідність сортування даних, у тому числі і даних, що зберігаються у вигляді масивів, дуже часто виникає при рішенні найрізноманітніших завдань. Якщо в мові Сі для того, щоб вирішити це завдання, треба написати десятки рядків коду, то в PHP це робиться однією простою командою.

Функція sort має наступний синтаксис

```
sort (масив [, прапорці])
```

і сортує масив, тобто упорядковує його значення по зростанню. Ця функція видаляє усі ключі, що існували в масиві, замінюючи їх числовими індексами, що відповідають новому порядку елементів. У разі успішного завершення роботи вона повертає true, інакше - false.

Приклад. Нехай у нас є два масиви: ціни товарів - їх назви і, навпаки, назви товарів - їх ціни. Упорядкуємо ці масиви по зростанню:

```
<?
$items = array(10 => "хліб", 20 => "молоко"
 30 => "бутерброд");
sort($items);
 // рядки сортуються в алфавітному
 // порядку, ключі втрачаються
print_r($items);
$rev_items = array("хліб" => 10,
 "бутерброд" => 30, "молоко" => 20);
sort($rev_items);
 // числа сортуються по зростанню
 // ключі втрачаються
print_r($rev_items);
?>
```

Отримаємо:

```
Array ([0]=> бутерброд,[1] => молоко,[2] => хліб )
Array ([0] => 10,[1] => 20,[2] => 30 )
```

Як додатковий аргумент може використовуватися одна з наступних констант :

- SORT_REGULAR - автоматичний вибір методу;
- SORT_NUMERIC - порівнювати елементи масиву як числа;
- SORT_STRING - порівнювати елементи масиву як рядки.

Якщо вимагається зберігати індекси елементів масиву після сортування, то треба використати функцію:

```
 asort (масив [, прапори]).
```

Якщо необхідно відсортувати масив в зворотному порядку, тобто від найбільшого значення до найменшого, то можна задіяти функцію:

```
rsort (масив [, прапори]).
```

А якщо при цьому треба ще і зберегти значення ключів, то слід використати функцію:

```
arsort(масив [, прапори]).
```

Як ви, напевно, помітили синтаксис у цих функцій абсолютно такий же, як у функції sort . Відповідно і значення прапорів можуть бути такими ж, як у sort: SORT_REGULAR, SORT_NUMERIC, SORT_STRING . До речі кажучи, прапор SORT_NUMERIC з'явився тільки в PHP4.

```
<?php
$books = array("Пушкін"=>"Руслан і Людмила"
 "Толстой"=>"Війна і мир"
 "Лермонтов"=>"Герой нашого часу");
asort($books);
 // сортуємо масив,зберігаючи значення ключів
print_r($books);
echo "<br>";
rsort($books);
 // сортуємо масив в зворотному порядку
 // ключі будуть замінені
print_r($books);
?>
```

В результаті роботи цього скрипта отримаємо:

```
Array ([Толстой] => Війна і мир
 [Лермонтов] => Герой нашого часу
 [Пушкін] => Руслан і Людмила )
Array ([0] => Руслан і Людмила
 [1] => Герой нашого часу
 [2] => Війна і мир )
```

Приклад. Нехай ми створюємо каталог описів документів. У кожного документу є автор, назва, дата публікації і короткий зміст. Ми вже не раз відображали описи, складені з цих характеристик. Кожного разу порядок відображення цих елементів залежав від створеної нами програми. Тепер же ми хочемо мати можливість змінювати порядок відображення елементів за бажанням користувача. Складемо для цього наступну форму:

```
<form action=task.php>
<table border=1>
```

```
<tr><td>Назва </td><td><input type=text
 name=title size=5> </td></tr>
<tr><td>Короткий зміст </td><td><input
 type=text name=description size=5>
</td></tr>
<tr><td>Автор </td><td><input type=text
 name=author size=5> </td></tr>
<tr><td>Дата публікації </td><td><input
 type=text name=published size=5></td></tr>
</table>
<input type=submit value="Відправити">
</form>
```

Упорядковуватимемо дані, передані цією формою, по спаданню їх значень, зберігаючи при цьому значення ключів. Для цього зручно скористатися функцією arsort(). Оскільки нам важливий тільки новий порядок елементів, збережемо в новому масиві ключі початкового масиву в потрібному порядку. Ми зберігаємо ключі початкового масиву, оскільки вони є іменами елементів, з яких конструюється опис документу, а пам'ятати їх важливо. Отже, отримуємо такий скрипт:

```
<?php
print_r($_GET); echo "<br>";
arsort ($_GET);
 // сортуємо масив в зворотному порядку
 // зберігаючи ключі
print_r($_GET); echo "<br>";
$ordered_names = array_keys($_GET);
 // Додаємо новий масив
foreach($ordered_names as $key => $val)
echo "$key:$val <br>";
 // виводимо елементи нового масиву
?>
```

### 8.3 Сортування масиву по ключах

Очевидно, що може виникнути необхідність в сортуванні масиву по значеннях ключів. Наприклад, якщо у нас є масив даних про книги, як в наведеному вище прикладі, то цілком імовірно, що ми захочемо відсортувати книги з імен авторів. Для цього в PHP також не треба писати багато рядків коду - можна просто скористатися функцією ksort() для сортування по зростанню (прямий порядок сортування ) або krsort() - для сортування по спаданню (зворотний порядок сортування ). Синтаксис цих функцій знову ж

таки аналогічний синтаксису функції sort().

```
<?php
$books = array("Пушкін"=>"Руслан і Людмила"
 "Толстой"=>"Війна і мир"
 "Лермонтов"=>"Герой нашого часу");
ksort($books);
 // сортуємо масив,
 // зберігаючи значення ключів
print_r($books);
?>
```

Отримаємо:

```
Array ([Лермонтов] => Герой нашого часу
 [Пушкін] => Руслан і Людмила
 [Толстой] => Війна і мир )
```

Окрім двох простих способів сортування значень масиву (по спаданню або по зростанню) PHP пропонує користувачеві можливість самому задавати критерії для сортування даних. Критерій задається за допомогою функції, ім'я якої вказується в якості аргументу для спеціальних функцій сортування usort() або uksort(). По назвах цих функцій можна здогадатися, що usort() сортує значення елементів масиву, а uksort() - значення ключів масиву за допомогою визначеної користувачем функції. Обидві функції повертають true, якщо сортування пройшло успішно, і false - інакше. Їх синтаксис виглядає таким чином:

```
usort (масив, сортуюча функція)
uksort (масив, сортуюча функція)
```

Звичайно ж, не можна сортувати масив за допомогою будь-якої призначеної для користувача функції. Ця функція повинна задовольняти певним критеріям, що дозволяють порівнювати елементи масиву. Як має бути влаштована сортуюча функція? По-перше, вона повинна мати два аргументи. У них інтерпретатор передаватиме пари значень елементів для функції usort() або ключів масиву для функції uksort(). По-друге, сортуюча функція повинна повертати:

- ціле число, менше нуля, якщо перший аргумент менший за другий;
- число, рівне нулю, якщо два аргументи рівні;
- число більше нуля, якщо перший аргумент більший за другий.

Як і для інших функцій сортування, для функції usort() існує аналог, що не змінює значення ключів, - функція uasort().

Приклад. Нехай у нас є масив, що містить такі відомості про літературні твори, як назва, автор і рік створення. Ми хочемо упорядкувати книги з дати створення.

```
<?php
// масив виглядає таким чином:
$books = array("Герой нашого часу" =>
 array ("Лермонтов", 1840),
 "Руслан і Людмила" => array("Пушкін", 1820),
 "Війна і мир" => array ("Толстой", 1863),
 "Ідіот" => array("Достоєвський", 1868));
/* можна, звичайно переписати цей масив інакше,
зробивши рік видання, наприклад, індексом, але
зручніше написати свою функцію сортування */
uasort($books, "cmp");
// сортуємо масив за допомогою функції cmp
foreach ($books as $key => $book){
 echo "$book[0]: \"$key\"<br>";
}
function cmp($a,$b){
// функція, що визначає спосіб сортування
 if ($a[1] < $b[1]) return - 1;
 elseif ($a[1]==$b[1]) return 0;
 else return 1;
}
?>
```

В результаті отримаємо:

```
Пушкін: "Руслан і Людмила"
Лермонтов: "Герой нашого часу"
Толстой: "Війна і мир"
Достоєвський: "Ідіот"
```

Ми застосували нашу власну функцію сортування до усіх елементів масиву. Далі розглянемо, як застосувати до елементів масиву будь-яку іншу призначену для користувача функцію.

Функція:

```
array_walk(масив, функція [, дані])
```

застосовує створену користувачем функцію до усіх елементів масиву масив і повертає true у разі успішного виконання операції і false - інакше.

Призначена для користувача функція, як правило, має два аргументи, в які по черзі передаються значення і ключ кожного елементу масиву. Але якщо при виклику функції array_walk() вказаний третій аргумент, то він буде розглянутий як значення третього аргументу призначеної для користувача функції, сенс якого визначає сам користувач. Якщо функція користувача вимагає більше аргументів, ніж в неї передано, то при кожному виклику array_walk() видаватиметься попередження.

Якщо необхідно працювати з реальними значеннями масиву, а не з їх копіями, слід передавати аргумент у функцію по посиланню. Проте треба мати на увазі, що не можна додавати або видаляти елементи масиву і виконувати дії, що змінюють сам масив, оскільки в цьому випадку результат роботи array_walk() вважається невизначеним.

```
<?php
$books1 = array(
 "А.С. Пушкін"=>"Руслан і Людмила"
 "Л.Н. Толстой"=>"Війна і мир"
 "М.Ю. Лермонтов"=>"Герой нашого часу");
// створюємо функцію, яку хочемо
// застосувати до елементів масиву

function try_walk($val,$key,$data){
 echo "$data $val написав $key<br>";
}
// застосовуємо до усіх елементів масиву
// $books1 функцію try_walk
array_walk($books1, "try_walk", "Роман");
?>
```

В результаті роботи скрипта отримаємо:

```
Роман "Руслан і Людмила" написав А.С. Пушкін
Роман "Війна і мир" написав Л.Н. Толстой
Роман "Герой нашого часу" написав М.Ю. Лермонтов
```

Помітимо, що ми не змінили значень у елементів масиву. Щоб їх змінити, потрібно було передавати значення в змінну $val функції try_walk по посиланню.

```
<?php
$books1 = array(
 "А.С. Пушкін"=>"Руслан і Людмила"
 "Л.Н. Толстой"=>"Війна і мир"
 "М.Ю. Лермонтов"=>"Герой нашого часу");
// створюємо функцію, яку хочемо
// застосувати до елементів масиву

function try_walk(&$val,$key){
 $key = "<p>Автор: ".$key ".<br>";
 $val = "Назва: ". $val ".</p>";
 echo $key.$val;
}
// застосовуємо до усіх елементів масиву
// $book1 функцію try_walk

array_walk($books1, "try_walk");
print_r($books1);
?>
```

В результаті роботи скрипта отримаємо:

```
Автор: А.С. Пушкін
Назва: "Руслан і Людмила"
Автор: Л.Н. Толстой
Назва: "Війна і мир"
Автор: М.Ю. Лермонтов
Назва: "Герой нашого часу"
Array ([А.С. Пушкін] =>
 Назва: "Руслан і Людмила"
 [Л.Н. Толстой] =>
 Назва: "Війна і мир"
 [М.Ю. Лермонтов] =>
 Назва: "Герой нашого часу")
```

### 8.6 Виділення підмасиву

Оскільки масив - це набір елементів, цілком імовірно, потрібно буде виділити з нього який-небудь піднабір. У PHP для цих цілей є функція array_slice . Її синтаксис такий:

```
array_slice (масив, номер_елементу [, довжина])
```

Ця функція виділяє підмасив довжини довжина в масиві масив, починаючи з елементу, номер якого заданий параметром номер_елементу.

Позитивний номер_елементу вказує на порядковий номер елементу відносно початку масиву, негативний - на номер елементу з кінця масиву.

```
<?php
$arr = array(1,2,3,4,5);
$sub_arr = array_slice($arr, 2);
print_r($sub_arr);
/* виведе Array ( [0] => 3 [1] =>4 [2] => 5 ) тобто
підмасив, що складається з елементів 3, 4, 5 */
$sub_arr = array_slice($arr,-2);
print_r($sub_arr);
 // виведе Array ( [0] => 4 [1] => 5 )
 // тобто підмасив, з елементів 4, 5
?>
```

Якщо задати параметр довжина при використанні array_slice, то буде виділений підмасив, що має рівно стільки елементів, скільки задано цим параметром. Довжину можна вказувати і негативну. Якщо в цю функцію переданий негативний параметр length, до послідовності увійдуть усі елементи початкового масиву, починаючи з позиції offset і закінчуючи позицією, віддаленою на length елементів від кінця array.

```
<?php
$arr = array(1,2,3,4,5);
$sub_arr = array_slice($arr, 2, 2);
 // містить масив з елементів 3, 4
$sub = array_slice($arr,-3, 2);
 // теж містить масив з елементів 3, 4
$sub1 = array_slice($arr, 0, - 1);
 // містить масив з
 // елементів 1, 2, 3, 4
$sub2 = array_slice($arr,-4, - 2);
 // містить масив з елементів 2, 3
?>
```

Є ще одна функція, схожа на array_slice(), - це array_chunk(). Вона розбиває масив на декілька підмасивів заданої довжини. Синтаксис її такий:

```
array_chunk ( масив, розмір [, зберігати_ключі])
```

В результаті роботи array_chunk() повертає багатовимірний масив, елементи якого є отриманими підмасивами. Якщо задати параметр зберігати ключі як true, то при розбитті будуть збережені ключі початкового масиву.

Інакше ключі елементів замінюються числовими індексами, які розпочинаються з нуля.

Приклад. У нас є список запрошених, оформлений у вигляді масиву їх прізвищ. У нас є столики на три персони. Тому треба розподілити усіх запрошених по троє.

```
<?php
$persons = array("Іванов", "Петров"
 "Сидорова", "Зайцева", "Волкова");
$triples = array_chunk($persons, 3);
 // ділимо масив на підмасиви по три елементи
foreach ($triples as $k => $table){
 // виводимо отримані трійки
 echo "За столиком номер $k сидять: <ul>";
 foreach ($table as $pers)
 echo "<li>$pers";
 echo "</ul>";
}
?>
```

В результаті отримаємо:

```
за столиком номер 0 сидять:
• Іванов
• Петров
• Сидорова
за столиком номер 1 сидять:
• Зайцева
• Волкова
```

Далі ми познайомимося з функцією, що обчислює суму усіх елементів масиву. Саме завдання обчислення суми значень масиву гранично просте. Але навіщо писати зайвий раз один і той же код, якщо можна скористатися спеціально створеною і завжди доступною функцією. Функція ця називається, як можна здогадатися, array_sum(). І в якості параметра їй передається тільки ім'я масиву, суму значень елементів якого треба обчислити.

Як приклад використання цієї функції приведемо рішення складнішої задачі, ніж просто обчислення суми елементів. Цей приклад також ілюструє застосування функції array_slice(), яку ми обговорювали трохи раніше.

Приклад. Нехай даний масив натуральних чисел. Треба знайти в ньому таке число, що сума елементів праворуч від нього дорівнює сумі елементів зліва від нього.

```
<?php
//масив задається функцією array
$arr = array(2,1,3,4,5,6,4);
// перебираємо кожен елемент масиву $arr. Усередині
// циклу поточний ключ масиву міститься в змінній //
$k, поточне значення - в змінній $val
foreach ($arr as $k => $val){
 $p = $k + 1;
 $out_next = array_slice($arr,$p);
 // отримуємо масив елементів,
    // що йдуть після поточного
 $out_prev = array_slice($arr,0,$k);
 // отримуємо масив елементів,
    // що йдуть перед поточним.
    // підраховуєио суму елементів
 $next_sum = array_sum($out_next);
 $prev_sum = array_sum($out_prev);
    // якщо сума елементів до поточного рівна
    // сумі елементів після, то виводимо
    // значення поточного елементу
 if ($next_sum==$prev_sum)
 echo "value:$val";
// можна подивитися вміст масивів на кожному кроці
 print_r($out_next); echo "<br>";
 print_r($out_prev);
 echo "$next_sum, $prev_sum<br>";
 echo "<hr>";
}
?>
```

Отже, ми вивчили ряд функцій, що спрощують роботу з масивами даних. Ми розглянули функції для пошуку значення серед елементів масиву; функції для сортування елементів масиву, як по їх значеннях, так і по ключах ; функції, що дозволяють застосовувати до усіх елементів масиву функцію, створену користувачем. Крім того, ми вивчили функцію, що виділяє підмасиви з елементів масиву, і функцію, що обчислює суму усіх елементів масиву. Використання усіх цих функцій було продемонстроване на прикладах. Усі функції для роботи з масивами доступні без яких-небудь конфігураційних налаштувань PHP, і користуватися ними можна абсолютно вільно.

**Контрольні питання до теми 8.**

1. Які є способи визначення масивів?
2. Що таке конкатенація масивів, як її виконати?

3. Як знайти потрібне (потрібні) значення в масиві?
4. Як отримати масив із унікальними значеннями?
5. Які у PHP є функції сортування масивів, в чому між ними відмінність?
6. Як задати специфічну функцію сортування?
7. Які є методи сортування масиву по ключах?
8. Як виділити фрагмент масиву?

# Тема 9. Робота з рядками

У одній з перших тем було наведено три способи задання рядків : за допомогою одинарних лапок, подвійних лапок і за допомогою heredoc-синтаксису. Відмічали ми і основні відмінності між цими способами. В основному вони торкаються обробки змінних усередині рядка.

```
<?php
echo 'В такому рядку НЕ обробляються змінні і
    більшість послідовностей';
echo "Тут змінні і послідовності обробляються";
echo <<<EOD
    Тут теж обробляються як змінні, так і керуючі
    послідовності. І крім того, можна вводити
    символи лапок без їх екранування зворотним
    слешем.
EOD;
?>
```

Вже не раз, починаючи з найпершої лекції, ми використали функцію echo . Насправді, echo - не функція, а мовна конструкція, тому використати при її виклику круглі дужки не обов'язково. Echo дозволяє виводити на екран рядки, передані їй в якості параметрів. Параметрів у echo може бути скільки завгодно. Їх розділяють комами або об'єднують за допомогою оператора конкатенації і ніколи не беруть в круглих дужок.

```
<?
echo "Прийшов ", "побачив ", "переміг ";
// виведе рядок "Прийшов побачив переміг"
// багато хто вважає за краще передавати декілька
// параметрів в echo за допомогою конкатенації
echo "Прийшов ". "побачив ". "переміг ";
// теж виведе рядок
// "Прийшов побачив переміг"
echo ("Прийшов ", "побачив ", "переміг ");
// видасть помилку: unexpected ','
?>
```

Окрім мовної конструкції echo існує ряд функцій для виведення рядків. Це в першу чергу функція print і її різновиди printf, sprintf і тому подібне.

Функція print дозволяє виводити на екран тільки один рядок і, як і echo, не може бути викликана за допомогою змінних функцій, оскільки є мовною

конструкцією.

Функція print_r не відноситься до рядкових функцій, як можна було б подумати. Вона відображає інформацію про змінну у формі, зрозумілій користувачеві.

Функції sprintf і printf обробляють переданий їм рядок відповідно до заданого формату. Але про них ми говорити не будемо. А поговоримо про те, як можна здійснювати пошук в тексті, представленому у вигляді рядка.

### 9.1 Пошук елементу в рядку

Для того, щоб визначити, чи входить цей підрядок до складу рядка, використовується функція strpos(). Синтаксис strpos() такий:

```
strpos (початковий рядок, рядок для пошуку
 [,з якого символу шукати])
```

Вона повертає позицію появи шуканого рядка в початковому рядку або повертає логічне false, якщо входження не знайдене. Додатковий аргумент дозволяє задавати символ, починаючи з якого здійснюватиметься пошук. Окрім логічного false ця функція може повертати і інші значення, які наводяться до false (наприклад, 0 або ""). Тому для того, щоб перевірити, чи знайдений шуканий рядок, рекомендують використати оператор еквівалентності " === ".

```
<?
$str = "Ідея наносити дані на перфокарти і потім
     зчитувати і обробляти їх автоматично належала
Джону
     Биллингсу а її технічне рішення здійснив Герман
     Холлерит.  Перфокарта  Холлерита  виявилася
настільки
     вдалою, що без щонайменших змін проіснувала до
наших
     днів".;
$pos = strpos($str, "Холлерит");
if ($pos !== false) echo "Шуканий рядок зустрівся
 в позиції номер $pos ";
else echo "Шуканий рядок не знайдено";
     /* помітимо, що ми перевіряємо значення $pos на
     еквівалентність з false. Інакше рядок, що
     знаходиться в першій позиції, не буде знайдено,
     оскільки 0 інтерпретується як false. */
?>
```

Якщо значення параметра рядок_для_пошуку не є рядком, то воно перетвориться до цілого типу і розглядається як ASCII -код символу. Щоб отримати ASCII -код будь-якого символу в PHP, можна скористатися функцією ord("символ").

Наприклад, якщо ми напишемо $pos = strpos($str, 228); то інтерпретатор вважатиме, що ми шукаємо символ "д". Якщо додати цей рядок в наведений вище приклад і вивести результат, то отримаємо повідомлення, що шуканий рядок знайдено у позиції 1.

Функція, протилежна по змісту до ord - це chr (код_символу). Вона по ASCII -коду виводить символ, що відповідає цьому коду.

За допомогою функції strpos можна знайти номер тільки першої появи рядка в початковому рядку. Природно, є функції, які дозволяють вичислити номер останньої появи рядка в початковому рядку. Це функція strrpos(). Її синтаксис такий:

```
strrpos (початковий рядок, символ для пошуку)
```

На відміну від strpos() ця функція дозволяє знайти позицію останньої появи в рядку вказаного символу.

Бувають ситуації, коли знати позицію, де знаходиться шуканий рядок, необов'язково, а треба просто отримати усі символи, які розташовані після входження цього рядка. Можна, звичайно, скористатися і приведеними вище функціями strpos() і strrpos(), але можна зробити і простіше - виділити підрядок за допомогою призначених саме для цього функцій.

### 9.2 Виділення підрядка

Говорячи про виділення підрядка з шуканого рядка в мові PHP, в першу чергу варто відмітити функцію strstr() :

```
strstr (початковий рядок, рядок для пошуку)
```

Вона знаходить першу появу шуканого рядка і повертає підрядок, починаючи з цього шуканого рядка до кінця початкового рядка.

Якщо рядок для пошуку не знайдений, то функція поверне false. Якщо рядок для пошуку не належить рядковому типу даних, то вона переводиться в ціле число і розглядається як код символу. Крім того, ця функція чутлива до регістра, тобто якщо ми паралельно шукатимемо входження слів "Ідея" і "ідея", то результати будуть різними. Замість strstr() можна використати

абсолютно ідентичну їй функцію strchr().

Приклад. Виділимо з рядка, що містить назву і автора дослідження, підрядок, що розпочинається із слова "Назва" :

```
<?
$str = "Автор: Іванченко Іван (<a href = mailto:
     van@mail.ua> написати лист</a>),
Назва: 'Дослідження мов програмування' ";
echo "<b>Початковий рядок: </b>", str;
if (!strstr($str, "Назва"))
 echo "Рядок не знайдений<br>";
else echo "<p><b>Отриманий підрядок: </b>
 strstr($str, "Назва");
?>
```

В результаті отримаємо:

```
Початковий рядок: Автор: Іванов Іван
 (написати лист)
 Назва: 'Дослідження мов
 програмування'
Отриманий підрядок: Назва:
 'Дослідження мов програмування'
```

Для реалізації регістронезалежного пошуку підрядка існує відповідний аналог цієї функції - функція stristr (початковий рядок, шуканий рядок). Діє і використовується вона точно так, як і strstr(), за винятком того, що регістр, в якому записані символи шуканого рядка, не грає ролі при пошуку.

Очевидно, що функція strstr() не занадто часто використовується - на практиці рідко буває треба отримати підрядок, що розпочинається з певного слова або рядка. Але в деяких випадках і вона може згодитися. Крім того, в PHP є і зручніші функції для пошуку входжень. Найбільш потужні з них, звичайно, пов'язані з регулярними виразами. Їх ми розглянемо в одній з подальших лекцій.

Іноді ми не знаємо, з яких символів розпочинається шуканий рядок, але знаємо, наприклад, що починається вона з п'ятого символу і закінчується за два символи до кінця початкового рядка. Як виділити підрядок по такому опису? Дуже просто, за допомогою функції substr(). Її синтаксис можна записати таким чином:

```
substr (початковий рядок,
```

```
позиція початкового символу [, довжина]).
```

Ця функція повертає частину рядка завдовжки, заданою параметром довжина, починаючи з символу, вказаного параметром позиція початкового символу. Позиція, з якою починається підрядок, що виділяється, може бути як позитивним цілим числом, так і негативним. У останньому випадку відлік елементів робиться з кінця рядка. Якщо параметр довжина опущений, то substr() повертає підрядок від вказаного символу і до кінця початкового рядка. Довжина підрядка, що виділяється, теж може бути задана негативним числом. Це означає, що вказане число символів відкидається з кінця рядка.

Приклад 8.5. Припустимо, у нас є фраза, виділена жирним шрифтом за допомогою тега <b> мови HTML. Ми хочемо отримати цю ж фразу, але в звичайному стилі. Напишемо таку програму:

```
<?php
$word = "<b>Hello, world!</b>";
echo $word "<br>";
$pure_str = substr($word, 3, - 4);
/* виділяємо підрядок,починаючи з 3-го символу,
 не включаючи 4 символи з кінця рядка */
echo $pure_str;
?>
```

В результаті роботи цього скрипта отримаємо:

**Hello, world!**
Hello, world!

Насправді вирішити таке задання можна набагато простіше, за допомогою функції strip_tags :

```
strip_tags (рядок [, допустимі теги])
```

Ця функція повертає рядок, з якого видалені усі html- та php-теги. За допомогою додаткового аргументу можна задати теги, які не будуть видалені з рядка. Список з декількох тегів вводиться без яких-небудь знаків роздільників. Функція видає попередження, якщо зустрічає неправильні або неповні теги.

```
<?php
$string = "<b>Bold text</b> <i>Italic text</i>";
```

```
$str = strip_tags($string);
 // видаляємо усі теги з рядка
$str1 = strip_tags($string, '<b>');
 // видаляємо усі теги окрім тега <b>
$str2 = strip_tags($string, '<i>');
 // видаляємо усі теги окрім тегів <i>
echo $str"<br>"str1"<br>"str2;
?>
```

В результаті отримаємо:

Bold text Italic text

**Bold text Italic text**

*Bold text Italic text*

Наведемо інший приклад використання функції substr(). Припустимо, у нас є якесь сполучення з вітанням і підписом автора. Ми хочемо видалити спочатку вітання, а потім і підпис, залишивши тільки змістовну частину повідомлення.

```
<?php
$text = "Привіт! Сьогодні ми вивчаємо роботу
 з рядками. Автор".;
$no_hello = substr($text, 8);
 // прибираємо вітання
$content = substr($text, 8, 38);
 // те ж саме, що substr($text, 8, - 6).
 // Прибираємо підпис.
echo $text "<br>"no_hello,
 "<br>"content;
?>
```

В результаті отримаємо:

```
Привіт! Сьогодні ми вивчаємо роботу з рядками. Автор.
Сьогодні ми вивчаємо роботу з рядками. Автор.
Сьогодні ми вивчаємо роботу з рядками.
```

Якщо нам треба отримати один конкретний символ з рядка, знаючи його порядковий номер, то не слід задіяти функції типу substr . Можна скористатися простішим синтаксисом - записуючи номер символу у фігурних дужках після імені строкової змінної. У контексті попереднього прикладу букву " р ",

розташовану другою по рахунку, можна отримати так:

```
echo $text{1}; // виведе символ "р"
```

Помітимо, що номером цього символу є число один, а не два, оскільки нумерація символів рядка робиться починаючи з нуля.

Раз вже ми почали говорити про символи в рядку і їх нумерації, то мимоволі виникає питання, скільки усього символів в рядку і як це вичислити. Число символів в рядку - це довжина рядка . Вичислити довжину рядка можна за допомогою функції strlen (рядок). Наприклад, довжина рядка "Розробка інформаційної моделі" обчислюється за допомогою команди: strlen ("Розробка інформаційної моделі"); і дорівнює 32 символам.

Отже, як виділяти і знаходити підрядки, ми розглянули. Тепер навчимося замінювати рядок, що входить до складу початкового рядка, на інший рядок по нашому вибору.

### 9.3 Заміна входження підрядка

Для заміни входження підрядка можна використати функцію str_replace(). Це проста і зручна функція, що дозволяє вирішувати безліч завдань, що не вимагають особливих тонкощів при виборі замінюваного підрядка. Для того, щоб робити заміни із складнішими умовами, використовують механізм регулярних виразів і відповідні функції ereg_replace() і preg_replace(). Синтаксис функції str_replace() такий:

```
str_replace(шукане значення,
 значення для заміни, об'єкт)
```

Функція str_replace() шукає в даному об'єкті значення і замінює його значенням, призначеним для заміни. Чому ми говоримо тут не про рядки для пошуку і заміни і початковий рядок, а про значення і об'єкт, в якому відбувається заміна? Річ у тому, що починаючи з PHP 4.0.5 будь-який аргумент цієї функції може бути масивом.

Якщо об'єкт, в якому здійснюється пошук і заміна, є масивом, то ці дії виконуються для кожного елементу масиву і в результаті повертається новий масив.

```
<?php
$greeting = array("Привіт", "Привіт всім"!,
 "Привіт, дорога"!); // об'єкт
```

```
$new_greet = str_replace("Привіт",
 "Добрий ранок", $greeting); // робимо заміну
print_r($new_greet);
/* отримаємо: Array ([0]=>Добрий ранок
 [1]=>Добрий ранок усім!
 [2]=>Добрий ранок, дорога!) */ ?>
```

Якщо шукане значення і значення для заміни - масиви, то береться по одному значенню з кожного масиву і здійснюється їх пошук і заміна в об'єкті. Якщо значень для заміни менше, ніж значень для пошуку, то як нові значення використовується порожній рядок.

```
<?php
$greeting = array("Привіт", "Привіт всім"!,
 "Привіт, дорога"!,"Вітаємо",
 "Вітаємо, друзі", "Ні"); // об'єкт
$search = array ("Привіт"
 " Вітаємо ", "Ні"); // значення, які замінюватимемо
$replace = array ("Добрий ранок", "День добрий");
 // значення, якими замінюватимемо
$new_greet    =    str_replace($search,    $replace,
$greeting);
 // робимо заміну
print_r($new_greet);
 //виводимо отриманий масив
?>
```

В результаті отримаємо такий масив:

```
Array (
[0] => Добрий ранок
[1] => Добрий ранок усім!
[2] => Добрий ранок, дорога!
[3] => День добрий
[4] => День добрий, друзі
[5] =>
)
```

Якщо значення для пошуку - масив, а значення для заміни - рядок, то цей рядок буде використаний для заміни усіх знайдених значень.

```
<?php
$greeting = array("Привіт", "Привіт всім"!,
 "Привіт, дорога"!, "Вітаємо",
```

```
 "ВІтаємо, друзі"); // об'єкт

$search = array ("Привіт", "Вітаємо");
 // значення, які замінюватимемо
$replace = "День добрий";
 // значення, яким замінюватимемо
$new_greet = str_replace($search,
 $replace, $greeting); // робимо заміну
print_r($new_greet);
 //виводимо отриманий масив
?>
```

Отримаємо:

```
Array (
[0] => День добрий
[1] => День добрий усім!
[2] => День добрий, дорога!
[3] => День добрий
[4] => День добрий, товариші
)
```

Функція str_replace() чутлива до регістра, але існує її регістронезалежний аналог - функція str_ireplace(). Проте ця функція підтримується не в усіх версіях PHP.

Функція substr_replace поєднує в собі властивості двох вже розглянутих нами функцій - функції str_replace() і substr(). Її синтаксис такий:

```
substr_replace (початковий рядок, рядок для заміни,
 позиція початкового символу [, довжина])
```

Ця функція замінює частину рядка рядком, призначеним для заміни. Замінюється та частина рядка (тобто підрядок), який розпочинається з позиції, вказаної параметром позиція початкового символу. За допомогою додаткового аргументу довжина можна обмежити число замінюваних символів. Тобто, фактично, ми не вказуємо конкретно рядок, який треба замінити, ми тільки описуємо, де він знаходиться і, можливо, яку довжину має. У цьому відмінність функції substr_replace() від str_replace().

Як і у випадку з функцією substr() аргументи позиція початкового символу і довжина можуть бути негативними. Якщо позиція початкового символу негативна, то заміна робиться, починаючи з цієї позиції відносно кінця рядка. Від’ємна довжина задає, скільки символів від кінця рядка не

мають бути замінені. Якщо довжина не вказується, то заміна відбувається до кінця рядка.

```
<?php
$text = "Мене звуть Вася".;
echo "Початковий рядок: $text<hr>";
/* Наступні два рядки замінять увесь
початковий рядок рядком 'А мене - Петя' */
echo substr_replace($text,
        'А мене - Петя', 0) . "<br>\n";
echo substr_replace($text, 'А мене - Петя',
        0, strlen($text)). "<br>\n";
// Наступний рядок додасть слово 'Привіт! '
// у початок початкового рядка
echo  substr_replace($text,  'Привіт!,  0,  0)  .
"<br>\n";

// Наступні два рядки замінять ім'я Вася
// на ім'я Іван в початковому рядку
echo  substr_replace($text,  'Іван',  11,  -1)  .
"<br>\n";
echo  substr_replace($text,  'Іван',  -5,  -1)  .
"<br>\n";
?>
```

В результаті роботи цього скрипта отримаємо:

```
Початковий рядок: Мене звуть Вася.
----------------------------------------
А мене - Петя
А мене - Петя
Привіт! Мене звуть Вася.
Мене звуть Іван.
Мене звуть Іван.
```

### 9.4 Розділення і з'єднання рядка

Дуже корисні функції - функція розділення рядка на частини і зворотна їй функція - об'єднання рядків в один рядок. Чому дуже корисні? Наприклад, якщо ви динамічно генеруєте форму за бажанням користувача, можна запропонувати йому вводити елементи для створення списку вибору, розділяючи їх яким-небудь символом. І для того, щоб обробити отриманий список значень, якраз і згодиться уміння розбивати рядок на шматочки. Для реалізації такого розбиття в PHP можна використати декілька функцій:

```
explode(роздільник, початковий рядок
 [,максимальне число елементів])
split (шаблон, початковий рядок
 [, максимальне число елементів])
preg_split (шаблон, початковий рядок
 [, максимальне число елементів
 [,прапорці]])
```

Останні дві функції працюють з регулярними виразами, тому в цій лекції ми їх розглядати не будемо. Розглянемо простішу функцію - explode().

Функція explode() ділить початковий рядок на підрядки, кожна з яких відокремлена від сусідньої за допомогою вказаного роздільника, і повертає масив отриманих рядків. Якщо заданий додатковий параметр максимальне число елементів, то число елементів в масиві буде не більше цього параметра, в останній елемент записується увесь залишок рядка. Якщо в якості роздільника вказаний порожній рядок "", то функція explode() поверне false. Якщо символу роздільника в початковому рядку немає, то повертається масив з початковим рядком без змін.

Приклад. Нехай потрібно створити елемент форми - випадний список і значення для цього списку повинні ввести користувач, не знайомий з мовою html. Створимо таку форму:

```
<form action=exp.php>
 Введіть варіанти для вибору автора статті
 через двокрапку (":") :<br>
 <input type=text name=author size=40>
 <br>
 <input type=submit value="Створити елемент">
</form>
```

Скрипт, який її оброблятиме (exp.php), може бути таким:

```
<?php
$str = $_GET["author"];
$names = explode(":",$str);
 // розбиваємо рядок введений
 // користувачем за допомогою":"
$s = "<select name=author>";
 // створюємо випадний список
foreach ($names as $k => $name){
 $s .= "<option value=$k>$name";
 // додаємо елементи до списку
}
```

```
$s .= "</select>";
echo $s;
?>
```

У результаті, якщо ми введемо рядок у форму (рис.9.1) то отримаємо випадаючий список (рис. 9.2).

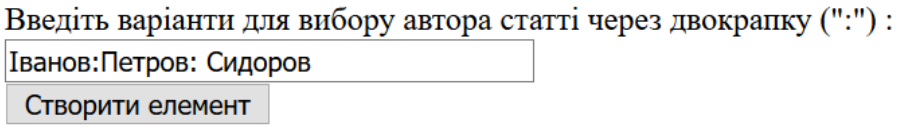



Рисунок 9.1 Введення значень для створення випадного списку


Рисунок 9.2 Випадаючий список, отриманий в результаті обробки форми

Окрім розділення рядка на частини іноді, навпаки, виникає необхідність об'єднання декількох рядків в одне ціле. Функція, пропонована для цього мовою PHP, називається implode() :

```
implode (string $glue, array $pieces)
```

Ця функція об'єднує елементи масиву за допомогою переданого їй об'єднуючого елементу (наприклад, комі). На відміну від функції explode(), порядок аргументів у функції implode() не має значення.

Приклад. Припустимо, ми зберігаємо ім'я, прізвище і по батькові людини окремо, а виводити їх на сторінці треба разом. Щоб з'єднати їх в один рядок, можна використати функцію implode() :

```
<?php
$data = array("Іванов", "Іван", "Іванович");
$str = implode(" "data);
echo $str;
?>
```

В результаті роботи цього скрипта отримаємо рядок:

```
Іванов Іван Іванович
```

У функції implode() існує псевдонім - функція join(), тобто ці дві функції відрізняються лише іменами.

### 9.5 Обробка рядків, що містять html-код

Досить часто ми працюємо з рядками, що містять html-теги. Якщо відобразити такий рядок у браузер за допомогою звичайних функцій відображення даних echo() або print(), то ми не побачимо самих html -тегів, а отримаємо рядок, що відформатував відповідно до цих тегів. Браузер обробляє усі html -теги відповідно до стандарту мови HTML. Іноді нам треба бачити безпосередньо рядок, без обробки її браузером. Щоб цього добитися, треба перед тим, як виводити, застосувати до неї функцію htmlspecialchars().

Функція:

```
htmlspecialchars (рядок [, стиль лапок [, кодування]])
```

переводить спеціальні символи, такі як «<», «>», «&», «"» , «'» у такі сутності мови HTML, як «<», «>», «&», «"», «'».

Додатковий аргумент стиль лапок визначає, як повинні інтерпретуватися подвійні і одинарні лапки. Він може мати одне з трьох значень : ENT_COMPAT, ENT_QUOTES, ENT_NOQUOTES. Константа ENT_COMPAT означає, що подвійні лапки мають бути переведені в спецсимволи, а одинарні повинні залишитися без змін. ENT_QUOTES говорить, що повинні конвертуватися і подвійні і одинарні лапки, а ENT_NOQUOTES залишає і ті і інші лапки без змін.

У параметрі кодування можуть бути задані такі кодування, як UTF-8, ISO-8859-1 і інші (кириличні кодування також підтримуються - див https://www.php.net/manual/en/function.htmlspecialchars.php).

```
<?php
$new = htmlspecialchars("<a
 href='mailto:au@ukr.net'>
 Написати лист</a>"ENT_QUOTES);
echo $new;

/* наш рядок перекодує в таку:
&lt;a href=&#039;mailto:au@ukr.net&#039;&gt;
Написати лист&lt;/a&gt; */
```

У браузері ми побачимо:

```
<a href='mailto: au@ukr.net '>
Написати лист</a>
```

Функція htmlspecialchars() перекодує тільки найчастіше використовувані спецсимволи. Якщо необхідно конвертувати усі символи по суті HTML, слід задіяти функцію htmlentities(). Російські букви при використанні цієї функції теж кодуються спеціальними послідовностями. Наприклад, буква " А " замінюється комбінацією " À ". Її синтаксис і принцип дії аналогічний синтаксису і принципу дії htmlspecialchars().

Отже, ми завершили знайомство з функціями роботи з рядками мови PHP. Звичайно ж, ми торкнулися далеко не усіх існуючих функцій, а лише малої частини. Ми вивчили функції, що дозволяють знайти набір символів в рядку, функції, замінюючі усі входження одного рядка на іншу, функції розділення рядка на частини і з'єднання декількох рядків в одну, а також розглянули функції, рядки, що дозволяють виводити на екран, містять html - код без їх форматування браузером.

**Контрольні питання до теми 9**

1. Назвіть способи задання рядкових масивів.
2. Яка різниця між способами задання масивів.
3. Як знайти символ в масиві?
4. Як отримати фрагмент рядка?
5. Назвіть функції для видалення тегів із рядка.
6. Як замінити знайдену в рядку послідовність символів?
7. Назвіть функції для розділення і з'єднання рядка.
8. Для чого використовується функція htmlspecialchars()?
9.

# Тема 10. Регулярні вирази

Регулярний вираз (regular expression, скорочено РВ) - це технологія, яка дозволяє задати шаблон і здійснити пошук даних, що відповідають цьому шаблону, в заданому тексті, представленому у вигляді рядка.

Крім того, за допомогою регулярних виразів можна змінити і видалити дані, розбити рядок за шаблоном на підрядки і багато що іншого.

Одне з поширених застосувань РВ - це перевірка рядка на відповідність яким-небудь правилам. Наприклад, наступний РВ призначене для перевірки того, що рядок містить коректний e-mail-адрес:

```
/^\w+([\.\w]+)*\w@\w((\.\w)*\w+)*\.\w{2,3}$/
```

Виглядає, звичайно, жахливо, але це працює. І якщо уміти користуватися цим механізмом віртуозно, то можна ефективно вирішити ряд задач з пошуку потрібної інформації.

Повернемося до нашого визначення РВ. У нім кілька разів повторюється термін "шаблон". Що це таке? В принципі, інтуїтивно зрозуміло, але спробуємо все ж пояснити.

Нехай маємо коректний e-mail -адрес. Це набір букв, цифр і символів підкреслення, після яких йде спеціальний символ "собака" @, потім ще один такий же набір, що містить ім'я сервера, крапку ( .) і дві або три букви, що вказують на зону домена, до якої належить поштова скринька (ua, com, org і так далі). Наведений вище РВ формалізує цей опис на мові, зрозумілій комп'ютеру. І описує не якусь конкретну електронну адресу, а усі можливі коректні електронні адреси. Таким чином, робиться формальне задання безлічі правильних e-mail 'ів за допомогою шаблону регулярного виразу.

Механізм регулярних виразів задає правила побудови шаблонів і здійснює пошук даних за цим шаблоном у вказаному рядку.

У подальшому викладі терміни РВ і "шаблон" часто використовуватимуться як синоніми, але важливо розуміти, що це не зовсім одне і те ж. Шаблон задає якийсь тип даних, а РВ - це механізм, який здійснює пошук і включає шаблон і опції пошуку, а також задає мову написання шаблонів.

Регулярні вирази прийшли з UNIX і Perl. У PHP існують такі зручні і потужні засоби роботи з рядками, як explode (розбиття рядка на підрядки), strstr (знаходження підрядка), str_replace (заміна усіх входжень підрядка). Виникає питання - навіщо придумувати щось ще?

Основна перевага РВ полягає в тому, що вони дозволяють організувати гнучкіший пошук, тобто знайти те, про що немає точного знання, але є шаблонне представлення. Наприклад, треба знайти усі семизначні номери телефонів, що зустрічаються в тексті. Ми не шукаємо якийсь заздалегідь відомий нам номер телефону, ми знаємо тільки, що шуканий номер складається з семи цифр. Для цього можна скористатися наступним РВ:

```
/\d{3}-\d{2}-\d{2}/m
```

У PHP існує два різні механізми для обробки регулярних виразів : POSIX-сумісні і Perl-сумісні(скорочено PCRE ). Їх синтаксис багато в чому схожий, проте Perl -сумісні регулярні вирази потужніші і до того ж працюють набагато швидше. Починаючи з версії PHP 4.2.0, PCRE входять в набір базових модулів і підключені за умовчанням. POSIX-сумісні РВ включені По замовчуваннютільки у версію PHP для Windows.

Основні функції для роботи з Perl-сумісними регулярними виразами:

```
preg_match(pattern, string, result[, flags])
i
preg_match_all(pattern, string, result, [flags]),
```

де:

pattern - шаблон регулярного виразу ;

string - рядок, в якому здійснюється пошук;

result - містить масив результатів (нульовий елемент масиву містить відповідність усьому шаблону, перший - першому "захопленому" підшаблону і так далі);

flags - необов'язковий параметр, що визначає, як впорядковані результати пошуку.

Ці функції здійснюють пошук за шаблоном і повертають інформацію про те, скільки разів стався збіг. Для preg_match() це 0 (немає збігів) або 1, оскільки пошук припиняється, як тільки знайдений перший збіг. Функція preg_match_all() здійснює пошук до кінця рядка і тому знаходить усі збіги. Усі точні збіги містяться в першому елементі масиву result у кожної з цих функцій (для preg_match_all() цей елемент - теж масив).

Про "захоплення" елементів буде розказано в розділі, присвяченому підвиразам.

Аналогом preg_match є булева функція POSIX -расширения ereg(string

pattern, string string [, array regs])

Функція ereg() повертає TRUE, якщо збіг знайдений, і FALSE - інакше.

Приклади, що наводяться далі, можна тестувати на перерахованих функціях. Наприклад, так:

```
<?
//рядок, в якому треба щось знайти
$str = "Мій телефонний номер: ".
"33-22-44. Номер мого редактора : ".
"222-44-55 і 323-22-33";
//шаблон, по якому шукати.
//Задає пошук семизначних номерів.
$pattern = "/\d{3}-\d{2}-\d{2}/m";
//функція, що здійснює пошук
$num_match=preg_match_all ($pattern,$str,$result);
//виведення результатів пошуку
for ($i=0;$i<$num_match;$i++)
 echo "Збіг $i : ".
 $result[0][$i]".<br>";
?>
```

### 10.1 Синтаксис регулярних виразів

Строге визначення регулярного виразу виглядає досить громіздко. Розпочнемо з неформального опису.

Регулярний виразу є рядком. Цей рядок складається з власне регулярного виразу (шаблону), виділеного за допомогою спеціального символу роздільника (це можуть бути символи " / " , " | ", " { ", " !" і т.п.) та модифікатора, що впливає на спосіб обробки РВ.

Надалі цей опис буде розширений.

Наприклад, в регулярному виразі /\d{3}-\d{2}-\d{2}/m символ " / "є роздільником, \d{3}-\d{2}-\d{2} - безпосередньо регулярний вираз (шаблон), а m - модифікатор.

Потужність регулярних виразів породжена в основі своєю їх здатністю включати в шаблон альтернативи і повторення. Вони кодуються в шаблоні за допомогою метасимволів. Метасимвол відрізняється від будь-якого іншого символу тим, що має спеціальне значення.

Одним з основних метасимволів є зворотний слеш " \ ". Він міняє тип символу, що йде за ним, на протилежний, тобто якщо це був звичайний символ, то він МОЖЕ перетворитися на метасимвол, якщо це був метасимвол, то він втрачає своє спеціальне значення і стає звичайним символом (це треба

для того, щоб вставляти в текст спеціальні символи як звичайні). Наприклад, символ d в звичайному режимі не має ніяких спеціальних значень, але \d є метасимвол, що означає "будь-яка цифра". Символ ". " у звичайному режимі означає "будь-який одиничний символ", а " \." означає просто крапку.

Інше призначення зворотного слеша - кодування недрукованих (службових) символів, таких як:

\n - символ переводу рядка;

\e - символ escape;

\t - символ табуляції;

\xhh - символ в шістнадцятиричному коді, наприклад \x41 є буква A і так далі

Ще одне призначення зворотного слеша - позначення генерованих символьних типів, таких як:

\d - будь-яка десяткова цифра ( 0-9 );

\D - будь-який символ, що не є десятковою цифрою;

\s - будь-який порожній символ (пропуск або табуляція);

\S - будь-який символ, що не є порожнім;

\w - символ, використовуваний для написання Perl-слів (це букви, цифри і символ підкреслення), так званий "словниковий символ";

\W - несловниковий символ (усі символи, окрім визначуваних \w ).

Що мається на увазі під "символьним типом"? Просто кожен метасимвол набуває значення (одне) з класу можливих значень, заданих автоматично або вручну. Символьні типи, що задаються користувачем, описуються за допомогою квадратних дужок (детальніше про це пізніше). Вище приведені символьні типи, діапазон значень яких заздалегідь визначений мовою програмування.

Приклад використання приведених вище метасимволів :

```
/\d\d\d plus \d is \w\w\w/
```

Це РВ означає: тризначне число, за яким йде підрядок plus, будь-яка цифра, потім is і слово з трьох словникових символів. Зокрема, цьому РВ задовольняють рядки: " 123 plus 3 is sum ", " 213 plus 4 is 217 ".

Взагалі розрізняють дві множини метасимволів : ті, що розпізнаються у будь-якому місці шаблону, за винятком внутрішності квадратних дужок, і ті, що розпізнаються усередині квадратних дужок.

Квадратні дужки [ ] застосовуються для опису підмножин, і усередині

регулярного виразу розглядаються як один символ, який може набувати значень, перерахованих всередині цих дужок. Проте якщо першим символом усередині дужок є ^, то значенням символьного класу можуть бути тільки символи, НЕ перераховані усередині дужок.

Приклади:

Символьний клас [абвгд] задає один з символів а, б, в, г, д, а клас [^абвгд] задає будь-який символ, окрім а, б, в, г, д.

Якщо написати [2бул]ки], то цей вираз інтерпретується як один з символів 2, б, у, л, за яким йде рядок ки], тому що перша закриваюча квадратна дужка (розбір відбувається зліва направо), що зустрілася, закінчує визначення символьного класу. Тобто цей РВ співпаде з одним з рядків 2ки], бки], уки] чи лки].

За допомогою РВ [0-9А-Яа-я] можна задати будь-яку букву або цифру.

Метасимволи, розпізнавані поза квадратними дужками, можна розділити на групи таким чином: що визначають положення шуканого тексту в рядку, пов'язані з підвиразами, обмежуючі символьний клас, квантифікатори і перерахування альтернатив.

Використання /m міняє поведінку символів ^ і $ - вони означають початок і кінець КОЖНОГО рядка

Приклади (^ і $)

Нехай даний такий текст, записаний у вигляді рядка :

```
$str = "11 aaa bbb "."ccc 22 ddd "."eee ggg 33";
```

Таблиця 10.1. Метасимволи, розпізнавані УСЕРЕДИНІ квадратних дужок

| Метасимвол | Значення |
|---|---|
| \ | Перехідний символ з множиною призначень |
| ^ | Заперечення класу, але тільки якщо це перший символ (наприклад, " ^\d " задає усе, окрім цифр) |
| - | Задає діапазон символів (наприклад, " 0-9 " задає усі цифри, " A - Z " - усі латинські букви) |
| ] | Визначає символьний клас |

Таблиця 10.2. Метасимволи, які розпізнаються ПОЗА квадратними дужками

| Метасимвол | Значення |
|---|---|
| \ | Перехідний символ з множиною призначень |
| ^ | Оголошує початок об'єкту (чи рядки у багаторядковому |

| | |
|---|---|
| | режимі). Тобто цей символ визначає, що шуканий текст повинен знаходитися на початку рядка. Альтернатива: " \A " |
| $ | Оголошує кінець об'єкту (чи рядки у багаторядковому режимі). Тобто цей символ визначає, що шуканий текст повинен знаходитися у кінці рядка. Альтернативи: " \Z ", " \z " |
| . | Співпадає з будь-яким символом, окрім символу переводу рядка (за умовчанням) |
| [ | Починає визначення символьного класу |
| ] | Закінчує визначення символьного класу |
| \| | Розділяє перерахування альтернативних варіантів |
| ( | Починає підшаблон (регулярний підвираз ) |
| ) | Закінчує підшаблон |
| ? | Розширює значення " ( ", квантифікаторів 0 або 1, і квантифікатор мінімізації |
| * | 0 чи більше повторень ( квантифікатор ) |
| + | 1 чи більше повторень ( квантифікатор ) |
| { | Починає мінімальний/максимальний квантифікатор |
| } | Закінчує мінімальний/максимальний квантифікатор |

Регулярний вираз /\d\d/m може бути зіставлений наступним підрядкам: 11, 22, 33. Якщо на початку РВ розміщено ^, то збіги шукаються на початку рядка, тому вираз /^\d\d/m знайде тільки 11.

Коли у кінці РВ розміщено знак долара $, пошук здійснюється у кінці рядка, тому виразу /\d\d$/m знайде тільки 33.

Шаблону ж /^\d\d\d$/ задовольнятиме рядок, що цілком складається з тризначного числа (тобто вона і починається і закінчується цим числом).

Знайдемо усі html -теги, розташовані на початку кожного рядка файлу 1.htm.

```
<?
//зчитуємо файл в рядок
$str = file_get_contents('1.htm');
$pattern = "!^<[^/]+>!mU";
// здійснюємо пошук
$n = preg_match_all ($pattern, $str, $res);
// виводимо результати
for ($i=0;$i<$n;$i++)
 echo htmlspecialchars($res[0][$i]). "<br>";
?>
```

Шаблон обмежений знаками окликів. Перший знак " ^ " означає, що ми шукаємо збігу на початку рядків, потім йде символ "< " - його і шукаємо в рядку, після нього повинне йти усе, що завгодно, окрім слеша (конструкція "[^/] " ), " + " говорить, що символ, що стоїть перед ним, повторюється один і більше разів і закінчується усе це символом "> ". Таким чином, виділяються усі теги на початку рядків.

Необхідно переконатися, що ім'я автора було записане правильно (спочатку прізвище з великої букви, потім ініціали через точку) і знаходиться у кінці рядка.

```
<?
//зчитуємо файл в рядок
$str = file_get_contents('1.htm');
$pattern = "!\s[А-Я][А-Яа-я]+".
 "\s([А-Я]\.\s*)$!m";
// шаблон обмежений знаками оклику, m - модифікатор
// включаючий багаторядковий режим. Перший \s
// означає, що перед прізвищем повинен йти порожній
// символ (наприклад, пропуск). [А-Яа-я] задає одну
// з букв алфавіту у будь-якому регістрі, а в
// комбінації зі знаком плюс визначає, що ця буква
// повторюється один і більше раз. Наступний \s
// означає, що між прізвищем та ініціалами
// має бути пропуск. Далі йде підвираз,
// що визначає ініціали. Це буква від А до Я, після
// якої розміщено крапка ('\.') Екрануємо крапку,
// щоб позбавитися від її спеціального значення.
// Після букви з точкою може йти або не йти пропуск
// чи декілька. Уся конструкція повторюється
// мінімум двічі. Останній символ $ означає
// що прізвище з ініціалами повинні знаходитися
// у кінці рядка.
// здійснюємо пошук
$n = preg_match_all ($pattern, $str, $res);

// виводимо результати
for ($i=0;$i<$n;$i++)
 echo htmlspecialchars($res[0][$i]). "<br>";
?>
```

Приклади ( | i .)

Нехай є деякий текст. Нам треба знайти усіх згаданих в нім людей із званнями.

```
<?
$str = "Доцент Смирнов здійснив відкриття.".
 " Його учителем була професор Іванова. ".
 "Цим відкриттям Смирнов завоював собі посаду ".
 "доктора. Раніше він був тільки кандидат".;
$pattern = "/(професор|доцент)".
 "\s[А-Я][А-Яа-я]+(\s|\.)/i";
// здійснюємо пошук
$n = preg_match_all ($pattern, $str
 $res);
// виводимо результати
for ($i=0;$i<$n;$i++)
 echo htmlspecialchars($res[0][$i]).
 "<br>";
?>
```

Метасимвол пряма риска "|" дозволяє задавати альтернативні варіанти. У прикладі ми хотіли знайти усіх професорів або доцентів. Для цього було створено підвираз "(професор|доцент)". Після звання через пропуск прізвище людини, якій воно належить, - для цього існує комбінація " \s[А-Я][А-Яа-я]+ ". Після прізвища йде або знову пропуск, або крапка, якщо це кінець пропозиції. Отримуємо знову два альтернативні варіанти:(тут крапка екранується зворотним слешем, щоб вона розумілася як звичайна крапка, без спеціального значення).

### 10.2 Підвирази (підшаблони)

У РВ підшаблони виділяють, беручи в круглі дужки. Для їх позначення окрім терміну " підшаблон " також використовують термін " підвираз ". Підшаблони можуть бути вкладеними. Виділення частини регулярного виразу у вигляді регулярного підвиразу робить наступне.

Локалізує безліч альтернатив.

Наприклад, шаблон

```
жар(ке|птиця)
```

співпадає з одним із слів " жарке ", " жарптиця " і " жар ". Тоді як без дужок це була б "жарке", " птиця " і порожній рядок.

Встановлює підшаблон як "захоплюючий" підшаблон. Це означає, що,

коли якийсь підрядок в тексті співпав з шаблоном, усі підрядки, які співпали з підшаблонами цього РВ, теж повертаються в якості результату. Дужки, що означають початок підшаблону, перераховуються зліва направо (починаючи з 1) для того, щоб дізнатися, скільки підшаблонів треба захопити.

Наприклад, є такий шаблон:

```
переможець отримає ((золоту|позолочену)
 (медаль|кубок))
```

і рядок, в якому шукаються збіги з цим шаблоном : " переможець отримає золоту медаль ". Тоді окрім цієї фрази будуть ще захоплені і видані як результати пошуку наступні збіги в підвиразах: " золоту медаль ", " золоту ", " медаль ", пронумеровані 1, 2, 3 відповідно.

Проте це не завжди зручно. Для того, щоб позбавитися від "захоплюючого" ефекту підвиразу, після відкриваючої дужки пишуть "?:". Тоді цей підвираз в результат пошуку не включається і при нумерації інших підшаблонів з "захоплюючим" ефектом не враховується.

```
переможець отримає ((?:золоту|позолочений)
 (медаль|кубок))
```

Тоді в умовах попереднього прикладу отримаємо шуканий рядок " переможець отримає золоту медаль " і рядки " золоту медаль ", " медаль ", пронумеровані 1 і 2 відповідно.

Якщо в html-файлі назва знаходиться після <body> і відокремлено від нього тільки пропусками або перекладами рядків, поміщено в тег <h1> і після нього теж може йти скількись пропусків і перекладів рядків, то його можна знайти за допомогою наступного скрипта:

```
<?
//зчитуємо файл в рядок
$str = file_get_contents('1.htm');
$pattern = "/<body.*?>[\n\s]*<h1>".
 "(.*?)<\/h1>[\n\s]*/m";
// здійснюємо пошук
$n = preg_match_all ($pattern, $str, $res);
echo $res[1][0]; // виводимо заголовок
?>
```

Помітимо, що тут виводиться перший захоплений підвираз, оскільки

нам цікава тільки сама назва, а не усе РВ. Оскільки в цьому РВ є тільки один підвираз, то його значення міститься в нульовому елементі першого масиву результатів.

### 10.3 Повторення у РВ (квантифікатори)

У попередніх прикладах ми часто писали комбінації типу \d\d. Це означає, що цифра повинна повторюватися двічі. А що ж робити, якщо повторень дуже багато або ми не знаємо, скільки саме? Виявляється, треба використати спеціальні метасимволи.

Повторення описуються за допомогою так званих квантифікаторів ( метасимволів, задаючих кількісні відношення). Існує два типи квантифікаторів : загальні (задаються за допомогою фігурних дужок) і скорочені (це скорочення найбільш поширених квантифікаторів, що історично склалися ).

Квантифікатори можуть йти за будь-яким з перерахованих елементів :

- одиночний символ (можливо, в комбінації із зворотним слешем );
- метасимвол "крапка";
- символьний клас;
- зворотне посилання (про них розповімо пізніше);
- підшаблон.

Загальні квантифікатори задають мінімальне і максимальне число дозволених повторень елементу; ці два числа, розділені комою, взяті у фігурні дужки. Числа не повинні перевищувати 65 536 і перше число має бути менше або рівне другому. Наприклад:

```
x{1,3}
```

говорить про те, що символ "x" повинен повторюватися мінімум один, а максимум три рази. Відповідно до цього шаблону задовольняють рядки: x, xx, xxx.

Якщо другий параметр відсутній, але кома є, то повторень може бути скільки завгодно. Таким чином

```
[aeuoi]{2,}
```

значить, що будь-який з символів " a ", " e ", " u ", " o ", " i " в рядку може повторюватися два і більше раз, а регулярний вираз \d{3} задає рівно три цифри.

Скорочені квантифікатори задають найбільш використовувані кількісні

відношення (повторення). Вони придумані для зручності, щоб не перевантажувати і без того складні вирази зайвим синтаксисом.

Виходячи з історичних традицій три що найчастіше зустрічаються квантифікатори, що мають наступні позначення:

* еквівалентно {0,} - тобто це нуль і більше за повторення;

+ еквівалентно {1,} - тобто це одне і більше за повторення;

? еквівалентно {0,1} - тобто це нуль або одне повторення.

Є ще один важливий момент, на який варто звернути увагу при вивченні квантифікаторів. По замовчуваннюусі квантифікатори "жадібні", вони намагаються захопити якомога більше повторень елементу. Тобто якщо вказати, що символ повинен повторюватися один і більше разів (наприклад, з допомогою * ), збіг станеться з рядком, що містить найбільше число повторень вказаного символу. Це може створити проблеми, наприклад, при спробі виділити коментарі в програмі на мові C або PHP. Коментарі в C і PHP записуються між символами /* і */, усередині яких теж можуть зустрічатися символи * і /. І спроба виявити C-коментарі за допомогою шаблону

```
/\* .* \*/
```

у рядку

```
/* перший коментар */
 не коментар
/* другий коментар */
```

не увінчається успіхом через «жадність» елементу «.*» (буде знайдено також рядок "не коментар").

Для вирішення цієї проблеми треба написати знак питання після квантифікатора. Тоді він перестане бути "жадібним" і спробує захопити як можна менше число повторень елементу, до якого він застосований ( квантифікатор застосовується до елементу, що розміщено перед ним). Так що шаблон

```
/\* .*? \*/
```

успішно виділяє C-коментарі.

У PHP існує опція PCRE_UNGREEDY, яка робить усі квантифікатори "не жадібними" По замовчуваннюі "жадібними", якщо після них йде знак питання.

```
<?
//Розглянемо html -файл, де є
//наступний рядок:
$str = "<div id=1>Привіт</div> <p>Текст, не ".
 "поміщений в тег div</p><div id=2>Поки</div>";
// Якщо ми хочемо знайти текст, що міститься між
// тегами div, природно написати такий шаблон:
$pattern = "!<div id=1>.*</div>!si";
// Але цей шаблон занадто "жадібний" і захопить
// також і текст між тегами <p>. Щоб цього
// уникнути, треба написати наступний шаблон, що
// відрізняється тільки наявністю знаку питання,
// яке забороняє квантифікатору бути "жадібним".
$pattern1 = "!<div id=1>.*?</div>!si";
// Запускаємо пошук в рядку $str збігів з шаблонами
// $pattern і $pattern1
$s = preg_match_all ($pattern, $str
 $res);
$js = preg_match_all ($pattern1
 $str, $res1);
//виводимо результати пошуку
// функція htmlspecialchars дозволяє виводити html
// його обробки браузером
echo "Жадібний шаблон :".
 htmlspecialchars($res[0][0]).
 "<br>";
echo "Нежадібний шаблон :".
 htmlspecialchars($res1[0][0]);
?>
```

Результати роботи скрипта :

```
"Жадібний" шаблон:<div id=1>Привіт</div>
<p>Текст, не поміщений в тег div</p>
<div id=2>Поки</div>
"Нежадібний" шаблон:<div id=1>Привіт</div>
```

Тепер ми в принципі можемо вирішити задання виділення змісту з html -файлу, якщо він знаходиться в тегу .

### 10.4 Модифікатори PCRE

Ще один важливий елемент регулярного виразу - це список застосовуваних до нього модифікаторів. Модифікатори - це видавана інтерпретатору регулярних виразів інструкція по обробці цього виразу.

Наприклад, вважати, що усі символи регулярного виразу відповідають як великим, так і маленьким буквам в рядку, де здійснюється пошук (табл. 10.3.)

Таблиця 10.3. Найчастіше використовувані модифікатори

| Позначення | Опис |
|---|---|
| i ( PCRE_CASELESS ) | Якщо вказаний цей модифікатор, то букви в шаблоні співпадають з буквами і верхнього, і нижнього регістра в рядку |
| m ( PCRE_MULTILINE ) | По замовчуванню рядок, що подається на вхід інтерпретатору РВ, розглядається як такий, що складається з однієї лінії. Цей модифікатор включає підтримку багаторядкового режиму |
| s ( PCRE_DOTALL ) | Якщо встановлений цей модифікатор, то метасимвол крапка "." співпадає з будь-яким символом, ВКЛЮЧАЮЧИ символ переводу рядка |
| x ( PCRE_EXTENDED ) | Примушує інтерпретатор ігнорувати пропуски між символами в шаблоні, за винятком пропусків, екранованих зворотним слешем або таких, що знаходяться усередині символьного класу, а також між неекранованим символом # поза символьним класом і символом нового рядка |
| U ( PCRE_UNGREEDY ) | Цей модифікатор інвертує "жадібність" квантифікаторів, тобто вони стають "нежадібними" по замовчуванню і "жадібними" якщо передують символу "?" |

## 10.5 Додаткові конструкції для роботи із регулярними виразами

Поза визначенням символьного класу (символьний клас задається квадратними дужками) комбінація зворотний слеш і цифра більше нуля (наприклад, \1 ) називається зворотним посиланням і є посиланням на захоплений раніше регулярний підвираз. Цих підвиразів рівно стільки, скільки відкритих круглих дужок (після яких немає знаку питання), розміщених лівіше цього елементу.

Зворотне посилання співпадає з конкретним вибраним значенням підвиразу, на який вона посилається, а не з будь-яким можливим значенням цього підвиразу. Таким чином, шаблон:

```
(відповідаль|надій)ний проявляє \ність
```

співпаде з рядками " відповідальний проявляє відповідальність ", " надійний проявляє надійність " і не співпаде з рядком " відповідальний проявляє надійність ".

Зворотні посилання можуть використовуватися усередині підвиразів. При першому використанні підвиразу посилання усередині нього не спрацьовує, але при подальших повтореннях підшаблону вона працює, як описана вище.

Твердження - це перевірка символів, які ідуть до або після поточного символу. Прості твердження закодовані послідовностями \A, \Z, ^, $ і так далі. Складніші твердження кодуються за допомогою підшаблонів. Постараємося коротко описати, як це робиться.

Існує два типи тверджень: ті, що дивляться на поточну позицію в початковому рядку ( "наглядачі вперед" ), і ті, що дивляться на символи перед поточною позицією ( "наглядачі назад" ).

Твердження, закодовані підшаблонами, порівнюються як звичайні підшаблони, за винятком того, що при їх обробці не відбувається зміни поточної позиції.

"Твердження, що дивляться вперед", шукають збіг в рядку за поточною позицією пошуку і починаються з (?= для позитивних тверджень і з (?! для негативних. Наприклад:

```
\w+(?=;)
```

співпадає із словом, що закінчується крапкою з комою (не включаючи крапку з комою в результат пошуку), і:

```
foo(?!bar)
```

співпадає з будь-якою появою foo, після якого немає bar. Як усе відбувається? Беремо рядок і шукаємо в нім foo. Як тільки знайшли, заглядаємо вперед (поточна позиція при цьому не міняється) і дивимося, чи йде далі bar. Якщо ні, то збіг з шаблоном знайдений, інакше продовжуємо пошук.

Регулярний вираз:

```
(?!foo)bar
```

не знайде усі входження bar, перед якими немає foo, тому що воно "дивиться вперед", а перед ним ніяких символів немає. Тому в цьому шаблоні ?!foo завжди вірно.

"Твердження, що дивляться назад", шукають збігу перед поточною позицією. Позитивні затвердження цього типу починаються з (?<=, негативні - з (?<! . Твердженням, що дивляться назад, дозволено шукати тільки рядка фіксованої довжини, тобто в них не можна використати квантифікатори. Наприклад:

```
(?<!foo) bar
```

знаходить усі появи bar, перед якими немає foo.

На початку лекції ми хотіли навчитися знаходити в html-файлі згадку про автора. Це можна зробити за допомогою тверджень, що "дивляться назад", в РВ (хоча можна і простіше).

```
<?
//зчитуємо файл в рядок
$str = file_get_contents('1.htm');
$pattern = "/(?<=Автор :)\s[А-Я]".
 "[а-я]*\s([А-Я]\.\s*){1,2}/m";
// здійснюємо пошук
$n = preg_match_all ($pattern, $str, $res);
// виводимо результати
for ($i=0;$i<$n;$i++)
 echo htmlspecialchars($res[0][$i]).
 "<br>";
?>
```

Частина РВ після твердження визначає, що ми шукаємо рядок (ПІБ), який розпочинається з пропуску, великої букви, потім йдуть маленькі букви в довільній кількості, пропуск і ініціали через точку. Твердження задає те, що перед цим рядком повинно стояти "Автор:".

Дату можна вичислити схожим чином. Залишаємо це в якості вправи.

Як у будь-якій мові програмування, в РВ існують умовні конструкції. Застосовуються вони до підвиразів. Тобто можна змусити процесор РВ вибирати підшаблон залежно від умови або вибирати між двома альтернативними шаблонами залежно від результату твердження або від того, чи співпав попередній захоплений підшаблон. Існують дві форми умовних підвиразів :

```
(?(умова) шаблон_виконуваний_якщо_
 умова_вірна)
(?(умова) шаблон_якщо_умова_вірна
 |шаблон_якщо_умова_невірна)
```

Існує два типи умов. Якщо текст між круглими дужками складається з послідовності цифр, то умова задовольняється, якщо захоплений підвираз з цим номером раніше співпав.

```
( \( )? [^()]+ (?(1) \) )
```

Перша частина цього РВ опціонально співпадає з круглою дужкою, що відкривається, і якщо цей символ є присутнім, то встановлює його як перший захоплений підвираз.

Друга частина співпадає з одним або більше символами, не поміщеними в круглі дужки.

Третя частина РВ - цей умовний підвираз, який перевіряє, чи співпала перша безліч дужок або ні (чи попалася нам в рядку відкриваюча кругла дужка). Якщо попалася, тобто об'єкт (рядок) розпочинається з символу " ( ", то умова вірно і обчислюється умовний шаблон, а саме потрібно наявність закриваючої круглої дужки. Інакше підшаблон ні з чим не співпадає.

Якщо умова - не послідовність цифр, то воно має бути твердженням. Це може бути позитивне або негативне твердження, що "дивиться вперед" або "дивиться назад".

```
(?(?=[^a - z]*[a - z])\d{2}-[a - z]{3}-\d{2}
 |\d{2}-\d{2}-\d{2})
```

Умова тут - позитивне твердження, що "дивиться вперед". Воно співпадає з будь-якою послідовністю не букв, після яких йде буква. Іншими словами, воно перевіряє присутність хоч би однієї букви в рядку для пошуку. Якщо буква знайдена, то робиться порівняння по першому альтернативному варіанту шаблону (\d{2}-[a - z]{3}-\d{2}), інакше - по другому (\d{2}-\d{2}-\d{2}). Цьому шаблону задовольняють рядки двох типів : dd - aaa - dd або dd - dd - dd, де d - будь-яка цифра, a - будь-яка буква.

Отже, у вищенаведеному матеріалі теми 10 було розглянуто механізм регулярних виразів, їх синтаксис і семантику, показали приклади їх використання. Взагалі механізм регулярних виразів є присутнім майже в усіх мовах програмування з невеликими відмінностями, але суть залишається тією

ж. Так що сподіваємося, що знання, отримані в процесі читання цієї лекції, допоможуть при вивченні інших мов і згодяться на практиці.

**Контрольні питання до теми 10.**

1. Що таке регулярний вираз (РВ)?
2. Для чого використовують регулярні вирази?
3. Назвіть функції для роботи із Perl-сумісними регулярними виразами.
4. Які роздільники можуть утворювати РВ.
5. Призначення символа ‘\’ у РВ.
6. Для чого використовуютья квадратні дужкиу РВ?
7. Які метасимволи розпізнаються всередині та поза квадратними дужками?
8. Як утворюються підвирази (підшаблони)?
9. Що таке квантифікатори, їх синтаксис?
10. Назвіть основні модифікатори PCRE, їх призначення.
11. Що таке зворотні посилання у РВ?
12. Що означає термін «жадібний» РВ?
13. Синтаксис та призначення РВ з умовами, їх різновиди.

# Лабораторна робота №1. Програмування з боку сервера. Введення в PHP. Методи GET та POST.

Тривалість: 2 акад. години.
Мета: набути практичних навичок з використання базових операторів PHP.
Завдання: ознайомитися із теоретичними відомостями та виконати завдання відповідно до ходу роботи.

## 1.1 Теоретичні відомості

Синтаксис PHP
Довідковий матеріал

### 1.1.1 Стилі PHP- дескрипторів

Існують чотири стилі PHP- дескрипторів (наведені нижче фрагменти коду еквівалентні):

– XML- стиль

```
<?php echo '<p>Hello world!</p>'; ?>
```

Даний стиль найбільш поширений (у даних методичних вказівках використовується саме такий стиль), адміністратори серверів не мають можливості вимкнути його, тому він гарантовано доступний у всіх випадках, що важливо у випадку розробки програмного коду, виконання якого передбачається у різних середовищах.

– Скорочений стиль

```
<? echo '<p>Hello world!</p>'; ?>
```

Це найпростіший стиль, що відповідає стилю інструкцій обробки мови SGML(StandardGeneralizedMarkupLanguage). Використання його не бажане, оскільки системні адміністратори часто його вимикають для запобігання конфліктів з XML-документами.

– SCRIPT-стиль

```
<script language='php'>echo '<p> Hello world!</p>';
</script>
```

SCRIPT- стиль часом застосовують у випадку виникнення проблем з іншими стилями у HTML-редакторах.

– ASP-стиль

```
<% echo '<p> Hello world!</p>'; %>
```

Аналогічний стиль дескриптора використовується у технологіях ASP (ActiveServerPages). Такому стилеві надають перевагу при роботі з редактором, орієнтованим на ASP.NET.

### 1.1.2 Оператори PHP

Дії, котрі повинен виконати інтерпретатор PHP, задаються операторами PHP, розташованими між відкриваючими та закриваючими дескрипторами:

```
echo '<p>Hello world!</p>';
```

Оператор `echo` виконує виведення у вікно браузера переданого йому рядка. Для розділення операторів використовується символ-крапка з комою: “;”.

### 1.1.3 Пробіли

Порожні символи, такі як порожній рядок (повернення каретки), пробіли між словами та символи табуляції, утворюють категорію пробільних. Браузери ігнорують пробільні символи у HTML-коді. Аналогічно діє і механізм PHP. Пробіли між PHP- операторами не є необхідними, проте вони підвищують читабельність коду. Фрагменти

```
echo 'Вітаємо';
echo 'на нашому сайті';
echo 'Вітаємо'; echo' на нашому сайті';
```

еквівалентні, але перша версія візуально сприймається легше.

### 1.1.4 Коментарі

Зазвичай коментарями супроводжуються більшість PHP-сценаріїв, за виключенням найпростіших.

Інтерпретатор PHP ігнорує текст, розміщений у коментарі. Для синтаксичного аналізатора коментар рівнозначний пропускам.

PHP підтримує коментарі у стилі C, C++ і сценаріїв оболонки.

```
/*Приклад
багаторядкового коментаря С-стилю*/
```

Багаторядкові коментарі починаються з символів /*і закінчуються символами */. Коментарі не можуть бути вкладеними.

Також використовують однорядкові коментарі у стилі C++:

```
echo ‘<p>Замовлення виконано</p>’; // початок виведення замовлення
```

або у стилі сценаріїв оболонки

```
echo ‘<p>Замовлення виконано</p>’; # початок виведення замовлення.
```

1.1.5 Доступ до змінних форми

Сенс використання форми полягає в отриманні скриптом інформації, введеної користувачем у текстові поля форми. Розглянемо приклад:

```
<html>
<head>
<title>Приклад форми</title>
</head>
<body>
<form action="form.php" method="post">
<input type="text" name="formvariable">
<input type="submit" value="Додати">
</form>
</body>
</html>
```

Даний HTML-скрипт створює форму для введення даних і кнопку “Додати”, яка передає введене скриптові form.php

Доступ до вмісту поля PHP-скриптом отримують наступним чином (файл form.php):

```
<?php
$variable=$_POST['formvariable'];
echo $variable;
?>
```

У даному випадкові передача даних реалізована за допомогою методу POST. У цьому випадкові відбувається неявна передача: користувач не помічає ніяких зовнішніх ознак процесу і не може здійснити припущення про імена змінних. Така передача зручна з точки зору безпеки.

Існує інший спосіб передачі даних – GET. При цьому дані передаються за допомогою сформованого додатку до назви файлу, що отримує дані. Зовні виглядає як лінк, при цьому користувач отримує доступ до імен змінних, що передаються, та їхніх значень.

```
<html>
<head>
<title>Приклад форми</title>
</head>
<body>
<form action="form.php" method="get">
<input type="text" name="formvariable">
<input type="submit" value="Додати">
</form>
</body>
</html>
```

Даний приклад відрізняється від попереднього методом передачі: замість POST використано GET.Нехай у поле введено значення “test”. Після натискання кнопки “Додати” буде особливим чином активізовано скрипт form.php: до адреси файлу буде додано символ “?”, назву змінної, переданої з форми, знак “=” та значення змінної:

```
http://localhost/form.php?formvariable=test
```

PHP-скрипт при цьому змінюють наступним чином:

```
<?php
$variable=$_GET['formvariable'];
echo $variable;
?>
```

Масиви `$_POST` та `$_GET` належать до категорії суперглобальних. Як у першому, так і у другому випадках, доступ до даних форми можна отримати через масив `$_REQUEST`.

```
<?php
$variable=$_REQUEST['formvariable'];
echo $variable;
?>
```

Детально використання масивів буде розглянуто у наступних лабораторних роботах.

Зазвичай блоки отримання даних з форми розташовують на початкові скрипта.

### 1.1.6 Конкатенація рядків

Операція конкатенації рядків використовується для об'єднання рядків (фрагментів тексту) у єдиний текст. Вона часто застосовується при виведенні даних у браузер за допомогою оператора `echo`. Реалізується за домового символу оператора конкатенації – крапки ".".

```
<?php
$text1="Hello, ";
$text2="world!";
echo $text1.$text2;
?>
<?php
$text1="world!";
echo "Hello, ".$text1;
?>
```

Довільну змінну, відмінну від змінної масиву, можна помістити у подвійні лапки і застосувати до неї оператор `echo`. Зверніть увагу на результат виконання двох наступних скриптів:

```
<?php
$text1="world!";
echo "Hello, $text1";
?>
```

Буде виведено у вікно браузера: Hello, world!

```
<?php
$text1="world!";
echo 'Hello, $text1';
?>
```

Результат: `Hello, $text1.`

Якщо ім’я змінної знаходиться між подвійними лапками, то ім’я замінюється значенням змінної, якщо ім’я змінної чи довільний текст обмежені одинарними лапками, то вони передаються без змін.

### 1.1.7 Ідентифікатори

Ідентифікатори – це імена змінних, функцій та класів. Використання ідентифікаторів регламентується наступними правилами:

– ідентифікатори можуть мати довільну довжину і складатися з букв, цифр та символів підкреслювання “_”;

– ідентифікатори не можуть починатися з цифри;

– у PHP ідентифікатори чутливі до регістру символів. Змінні `$formvariable $FormVariable`– це різні змінні. Виключення складають вбудовані PHP-функції – їхні імена можуть бути представлені довільним регістром;

– змінні можуть мати імена, що співпадають з іменами вбудованих функції. Однак, це може призвести до плутанини, тому подібних ситуацій слід уникати. Не можна також створювати функції, чиї імена співпадають з іменами вбудованих функцій.

Імена змінних у PHP починаються знаком долара ($). Пропуск цього знаку – поширена помилка.

### 1.1.8 Типи змінних

Тип змінної характеризується видом даних, котрі вона зберігає. PHP підтримує наступні базові типи даних:

– Integer (цілий) – використовується для представлення цілих чисел;

– Float (ще їх називають double – подвійної точності) – використовується для представлення дійсних чисел;

– String (рядковий) – використовується для представлення символів;

– Boolean (булевий) – використовується для зберігання значень true та false;

– Array(масив) – використовується для зберігання декількох елементів даних;

– Object (об’єкт) – використовується для зберігання екземплярів класу.

Доступні також два спеціальних типи – NULL та resource (ресурс). Змінні, котрим не присвоєно конкретного значення, котрі не визначені або набувають значення NULL, належать до типу NULL. Деякі вбудовані функції (такі, як для роботи з базами даних) повертають змінну ресурсного типу (наприклад, з’єднання з базами даних).

Також PHP підтримує типи pdfdoc та pdfinfo, якщо встановлена підтримка обробки PDF-документів.

Мова PHP – слабо типізована. На відміну від мови C, тип змінної визначається типом присвоєного їй значення.

### 1.1.9 Перетворення типів

За допомогою механізму перетворення типів можна переводити змінну або конкретне значення до іншого типу. Перетворення відбувається так само, як і на мові C.

```
<?php
$a=2.34;      // Змінна $a має тип float
$a=(int)$a;   // Тепер змінна $a має тип integer
echo $a;
?>
```

### 1.1.10 Змінні змінних

Усі мови програмування дозволяють змінювати значення змінної, деякі – тип змінної і зовсім небагато мов дозволяють змінювати ім’я змінної.

В основі цієї можливості – ідея використання значення однієї змінної як імені іншої.

Нехай існує змінна $variable. Тоді:

```
<?php
$variable=NULL;
$varname="variable";
$$varname="Hello, world!";
echo $variable;
?>
```

Запис `$$varname="Hello, world!"` еквівалентний наступному: `$variable="Hello, world!".`

1.1.11 Константи

Разом із використанням змінних, PHP надає можливість оголошення констант. Як і змінна, константа містить значення, але її значення встановлюється одноразово і не може бути зміненим у сценарії.

Константи оголошуються за допомогою функції `define`

```
<?php
define('CONSVALUE',10);
echo CONSVALUE;
?>
```

Зверніть увагу на те, що імена констант записують символами у верхньому регістрі. Дана особливість перейшла з мови С. Дотримуватися її не обов’язково, проте вона значно спрощує читання і супровід коду.

Важливо: при звертанні до констант не використовується знак долара ($).

1.1.12 Область дії змінних

Термін “область дії” належить тим розділам сценаріїв, в яких можливий доступ до деякої конкретної змінної, іншими словами, це – область, з довільного місця якої видно дану змінну. У PHP використовують шість базових правил визначення області дії:

– вбудовані суперглобальні змінні видно з довільного місця сценарію;

– константи, щойно вони оголошені, видно глобально, тобто можуть використовуватися як зовні так і всередині функцій;

– глобальні змінні, оголошені у сценарії, видно у довільному місці сценарію, але не всередині функцій;

– змінні, що використовуються всередині функцій, що оголошені як глобальні, посилаються на глобальні змінні з тими самими іменами;

– змінні, що використовуються всередині функцій, що оголошені як статичні, невидимі поза межами функцій, проте вони зберігають свої значення поміж двома викликами цих функцій;

– змінні, створені всередині функцій, є локальними стосовно своєї функції і припиняють існування після завершення функції.

Вбудовані суперглобальні змінні у PHP версій від 4.1 і вище наступні:

– `$_GLOBALS` – масив глобальних змінних. Дає можливість доступу до глобальних змінних всередині функції, наприклад: `$_GLOBALS['myvariable'];`

– `$_SERVER` – масив змінних середовища сервера;

– `$_GET` – масив змінних, переданих у сценарій за допомогою методу GET;

– `$_POST` – масив змінних, переданих у сценарій за допомогою методу POST;

– `$_COOKIE` – масив cookie- змінних;

– `$_FILES` – масив змінних завантаження файлів;

– `$_ENV` – масив змінних оточення;

– `$_REQUEST` – масив, пов'язаний із введенням даних користувачем, включаючи `$_GET, $_POST, $_COOKIE`;

– `$_SESSION` – масив змінних сеансу.

### 1.1.13 Посилання

Операція посилання позначається як "`&`" і може використовуватися у поєднанні з операцією присвоювання.

Зазвичай, коли значення змінної `$a` присвоюється змінній `$b`, створюється копія змінної `$a`, яка зберігається у пам'яті. Якщо у майбутньому значення `$a` буде змінене, то `$b` зміни не торкнуться.

```
$a=5;
$b=$a;   // $b=5;
```

```
$a=6;    // як і раніше, $b=5
```

Створення копії можна уникнути, використавши операцію посилання `&`:

```
$a=5;
$b=&$a;  // $b дорівнює 5;
$a=6;    // зверніть увагу: $b дорівнює 6
```

`$a` та `$b` вказують на одну і ту ж ділянку пам'яті. Цей зв'язок можна розірвати, "скинувши" одну зі змінних:

```
unset($a);
```

При цьому значення `$b` продовжуватиме існувати.

1.1.14 Операції порівняння

Операції порівняння наведено у табл.1.1. Результатом виконання операції порівняння є `true` або `false`.

Таблиця 1.1 – Операції порівняння PHP

| Операція | Назва | Використання |
|---|---|---|
| == | дорівнює | $a == $b |
| === | тотожно | $a === $b |
| != | не дорівнює | $a != $b |
| !== | не тотожно | $a !== $b |
| <> | не дорівнює | $a <> $b |
| < | менше | $a < $b |
| > | більше | $a > $b |
| <= | менше або дорівнює | $a <= $b |
| >= | більше або дорівнює | $a >= $b |

Операція перевірки тотожності повертає значення `true` тільки у тому випадкові, якщо обидва операнди рівні і мають однаковий тип. Наприклад:

```
$a=0;
$b='0';
echo($a==$b);     // результат: true
echo($a===$b);    // результат: false
```

Логічні операції слугують для комбінування результатів логічних умов. Перелік логічних операцій разом з описом їхнього застосування наведено у табл. 1.2.

Таблиця 1.2 – Логічні операції PHP

| Операція | Назва | Використання | Результат |
|---|---|---|---|
| ! | НЕ | !$b | Повертається true, якщо значення $a дорівнює false та навпаки |
| && | І | $a && $b | Повертається true, якщо обидві змінні $a та $b мають значення true, в іншому випадкові повертається значення false |
| \|\| | АБО | $a \|\| $b | Повертається true, якщо довільна зі змінних $a або $b має значення true, інакше повертається false |
| and | І | $a and $b | Те саме, що і &&, але з меншим пріоритетом |
| or | АБО | $a or $b | Те саме, що і \|\|, але з меншим пріоритетом |

1.1.15 Оператори if

Для прийняття рішень використовується оператор if. Операторові необхідно задати умову. Якщо умова рівна true, то виконується блок коду, розташований після оператора. Умова в операторі if записується поміж круглими дужками “(…)”. Приклад:

```
<?php
$a=0;
$b=3;
if($a<$b) echo '$a < $b';
?>
```

1.1.16. Блоки коду

Часто всередині такого умовного оператора, як if, необхідно виконати більше одного оператора. У такому випадку відповідна послідовність операторів записується у вигляді блоку. Для оголошення блоку оператори необхідно оточити фігурними дужками:

```
if($a<$b)
{
echo '$a < $b';
```

```
$a=$b;
}
```

Таким чином при виконанні умови ($a<$b) буде виконано обидва оператори

```
echo '$a < $b';
$a=$b;
```

Якщо ж умова не виконуватиметься, то обидва оператори будуть проігнорованими.

1.1.17. Оператори else

Оператор else дозволяє визначити альтернативну дію, котра повинна виконатися, якщо значення умови в операторі if виявиться false. Наприклад:

```
<?php
$a=0;
$b=3;
echo '$a='.$a.'; $b='.$b.'<br />';
if($a<$b)
echo '$a < $b';
else
echo '$a < $b';
?>
```

Результат виконання скрипта:

```
$a=0; $b=3
$a< $b
```

Вкладання операторів if один в одного дозволяє будувати складні ланцюжки.

1.1.18 Оператори elseif

У багатьох випадках прийняття рішення передбачає вибір відповідного

варіанту з деякої множини можливих варіантів. Послідовність цієї множити можна створити за допомогою оператора комбінації операторів `if – else`. За наявності послідовності умов програма може перевіряти кожну з них до тих пір, поки не знайде таку, значення якої буде `true`. Наприклад:

```
<?php
$a=23;
if($a>0 && $a<10) echo 'Результат - у першому інтервалі';
elseif($a>=10 && $a<20) echo 'Результат - у другому інтервалі';
elseif($a>=20 && $a<30) echo 'Результат - у третьому інтервалі';
?>
```

Після виконання скрипта отримують: “Результат - у третьому інтервалі”

### 1.1.19 Оператори `switch`

Оператор `switch` працює аналогічно до оператора `if`, але надає можливість умовному виразові мати як результат більше двох значень. В операторі `if` умова набуває значення `true` або `false`. Натомість в операторі `switch` умова може набувати довільну кількість значень у тих випадках, коли результат його обчислення – простий тип (`integer`, `string` чи `float`). Для забезпечення реагування на кожне таке значення слід передбачити для нього відповідний оператор `case`, а також (не обов’язково) визначити дії, що виконуватимуться по замовчуванню, якщо виникне випадок, не передбачений оператором case. Наприклад:

```
<?php
$a=1;
switch ($a)
{
case 1:
echo '$a=1';
break;
case 2:
echo '$a=2';
```

```
break;
default:
echo '$a='.$a;
break;
}
?>
```

Результат виконання: $a=1.

### 1.1.20 Цикли while

Найпростішим циклом у PHP є цикл while. Різниця між оператором if та оператором while полягає у тому, що у випадку виконання умови оператор if виконує блок коду тільки один раз, а оператор while повторює його до тих пір, поки умова виконується.

Наприклад:

```
<?php
$var=1;
while($var<=5)
{
echo $var.'<br />';
$var++;
}
?>
```

### 1.1.21 Цикли for

Цикл типу while можна записати і у більш компактній формі за допомогою оператора for. Базова структура оператора for має вигляд:

```
for(вираз 1; умова; вираз 2)
вираз 3;
```

Вираз 1 виконується один раз на початкові циклу. Як правило, він встановлює початкове значення змінної циклу.

Вираз умова перевіряється перед кожною ітерацією. Якщо цей вираз повертає значення false – цикл зупиняється.

Вираз 2 виконується у кінці кожної ітерації. Зазвичай у ньому

змінюється значення лічильника.

Вираз 3 виконується один раз під час кожної ітерації. Це, як правило, – блок коду, який містить тіло циклу. Наступний приклад виводить табличку з двох стовпчиків і п’яти рядків.

```
<?php
echo "<table>";
for($distance=50; $distance<=250; $distance+=50)
{
echo "<tr>\n <td align='right'> $distance </td>\n";
echo "<td align=right'> ".$distance."
</td>\n</tr>\n";
}
echo "</table>";
?>
```

### 1.1.22 Цикли do…while

Загальні структура оператора do…while має вигляд:

```
do
вираз;
while(умова 1);
```

Цикл do…while відрізняється від циклу while тим, що в ньому умова перевіряється у кінці. Отже, у циклі do…while оператор або блок операторів всередині циклу завжди виконується, принаймі, один раз.

```
<?php
$a=10;
do
{
echo $a.'<br />';
$a-=1;
}
while($a>=0);
?>
```

## 1.2 Хід роботи

1. Зареєструватися на веб-хостингу (наприклад https://infinityfree.net/ (рекомендовано), https://www.000webhost.com/ (працює не дуже стабільно), https://www.zzz.com.ua (тільки тиждень безкоштовного користування, далі платно, але працює більш-менш стабільно), https://profreehost.com/). Задати пароль для FTP-з'єднання. Якщо Ви вже є зареєстровані і термін дії закінчується не швидше 30 червня поточного року, то заново реєструватися не потрібно.
2. Налаштувати редактор текстів (VS code, Sublime, Notepad++, тощо) для роботи з FTP-сервером.

*Інструкція для виконання перших двох завдань тут.*

3. Завантажити на вебхостинг папку PHP. Розібратися із запропонованими викладачем файлами. Переглянути їх через браузер. Прізвище dutchak у назві папки замінити на своє. Папки всіх решту робіт називати labN_lastname, де N - номер лабораторної роботи, lastname - ваше прізвище.
4. У файл config.php в межах блоку php додано наступний код:

```
$LastModified_unix = strtotime(date("D, d M Y H:i:s",
filectime($_SERVER['SCRIPT_FILENAME'])));
$LastModified = gmdate("D, d M Y H:i:s \G\M\T",
$LastModified_unix);
echo "Файл був змінений: $LastModified"."<br>";
```

**Файл config.php повинен бути приєднаним до всіх файлів всіх лаб. робіт і бути єдиним (цієї вимоги також потрібно дотримуватися щодо інших файлів і папок, які знаходяться безпосередньо в папці PHP, окрім папок labN_lastname), тоді даний код буде виводити повідомлення про час останньої зміни файлу.**

**У файл config.php додано код:**

```
$path = __DIR__;
echo "path to config.php=".$path."<br>";  # цей шлях
можете вказувати при заданні абсолютного шляху до файлу
config.php із будь-яких php-файлів на різних рівняхі,
наприклад цей код у файлі lab1.php може виглядати
орієнтовно так "require
'/home/vol1_1/infinityfree.com/if0_35905901/htdocs/conf
ig.php';"
```

Щоб навіть при невдалій спробі приєднання файлів, решту сторінки відображалося, можете скористатися наступним кодом

```
try {
require '/home/vol1_1/infinityfree.com/if0_35905901/htdocs/config.php';
 } catch (\Throwable $e) {
    echo "This was caught: " . $e->getMessage();
}
```

Такий варіант абсолютної адресації для require не працює:
require 'http://mariia-dutchak.infinityfreeapp.com/config.php';

Тому потрібно використовувати або відносну адресацію, або через абсолютну адресу отриману через: `$path = __DIR__`

Після отримання шляху до файлу config.php , рядки

```
$path = __DIR__;
echo "path to config.php=".$path."<br>";
```

закоментуйте, або видаліть, щоб вони не виводилися у кожному файлі.

5. Реалізувати і дослідити функціонування операторів, дія яких розглядається у теоретичних відомостях. Посилання (назва повинна містити номер підпункту прикладу) на файл із реалізацією кожного із прикладів розмістити у файлі lab1.php.

6. Модифікувати приклад в файлі example1_1_5_1.php наступним чином:

- вбудувати html-код в php-блок;
- додати ще одну форму із текстовим полем і кнопкою, яка б методом GET передавала в example1_1_5_2.php введене число. Одержане число у файлі example1_1_5_2.php помножити на 2, вивести із вказівкою, що вхідні дані одержані із файлу example1_1_5_1.php;

7. Написати програму, яка виводить на сторінці таке меню:

Введіть номер завдання:

1 - обчислення максимальної температури;

2 - обчислення мінімальної температури;

3 - обчислення середньої температури.

Створює 4 текстових поля для введення значення температури і номера завдання та кнопку для передачі даних.

Програма повинна виконати завдання, номер якого буде введено (Використати оператор Switch ).

Дано: три цілих числа, що є значеннями температури та ціле число (зі значенням 1, 2, або 3), яке є номером завдання. Номер завдання вказує, які дії виконувати зі значеннями температури.

Знайти: максимальну, або мінімальну, або середню температуру в залежності від введеного номера завдання.

8. На Google Drive створити папку з назвою предмету, для облікового запису mariia.dutchak@pnu.edu.ua надати до неї доступ з правами коментування, для всіх решту встановити обмежений доступ. При наданні доступу зніміть відмітку біля сповістити.

Увага! Цю та всі наступні роботи потрібно розміщувати на вебхостингу, копії папок із виконаними роботами завантажуєте у папку дисципліни на Google Drive.

9. Для здачі робіт, у Classroom присилаєте посилання на відповідні папки з роботами, розміщені на Google Drive і знімок екрану, який демонструє розміщення файлів із виконаною роботою на вебхостингу.

10. Всі наступні роботи теж розміщуйте на вебхостингу, папки із окремими лабораторними роботами називайте labN_lastname, де N- номер роботи, lastname - Ваше прізвище. На стартовій сторінці дисципліни index.php робіть посилання на стартові сторінки відповідних робіт під назвою labN.php, N-номер роботи, в них створіть посилання на виконані завдання відповідної роботи, і бажано зворотні посилання на відповідну сторінку labN.php. Файл labN.php має знаходитися у відповідній папці labN_lastname. Також файл index.php із папки дисципліни повинен містити Ваше ПІБ та номер варіанту (порядковий номер у списку групи).

11. Посилання на стартову сторінку дисципліни на вебхостингу і посилання на папку предмету на Google Drive вкажіть тут (форму заповнюєте тільки один раз, за винятком зміни адреси вебхостингу чи папки предмету на Google Drive):
https://docs.google.com/forms/d/e/1FAIpQLSescc-Tzx9GP24oNBAKhy_vokuvTLzjSF1C2DkNyDul23HYVg/viewform?usp=pp_url

12.Здайте роботу.

### 1.3 Контрольні запитання:

1. Які типи PHP- дескрипторів існують на даний час?
2. Поясніть суть оператора у PHP.
3. Які особливості використання пробілів?
4. Які типи коментарів використовують при програмуванні на PHP?
5. Яким чином реалізують доступ до змінних форми?
6. Що таке конкатенація рядків та як вона реалізується?
7. Поясніть зміст поняття “ідентифікатор” на PHP.

8. Які типи змінних підтримує PHP?
9. Поясніть суть поняття “змінна змінних”?
10. Яким чином оголошуються константи на PHP?
11. Поясніть суть поняття “область дії змінних”?
12. Яке призначення посилань?
13. Операції порівняння на PHP.
14. Яким чином реалізують розгалуження у програмі?
15. Які оператори для створення циклів надає PHP?

## Лабораторна робота №2. PHP. Математичні функції. Оператори. Передача та отримання даних

Тривалість: 2 акад. години.
Мета: набути практичних навичок з використання базових операторів PHP.
Завдання: виконайте завдання згідно ходу роботи.

Увага! Усі виконані роботи розміщувати на вебхостингу, із вказанням покликання на них на стартовій сторінці дисципліни. Роботи мають знаходитися у відповідних папках labN, де N – номер роботи.

### 2.1 Теоретичні відомості

Теоретичні відомості Лабораторної роботи 1
Синтаксис PHP
Математичні функції
Передача значення змінних через посилання

Передача кількох змінних через посилання:

```
<a href="pass-value.php?value=1&value2=2&value3=3">
Link text </a>
```

### 2.2 Хід роботи

1. Створіть форму з кнопкою типу Submit та двома текстовими полями, в які користувач вводить два числа. Далі після натискання кнопки Submit викликається php-скрипт, який виводить три наступні варіанти дії між цими числами: наприклад 5-2 = 3; 5 * 2 = 10; 5%2 = 1.
2. Створіть форму з двома (трьома) текстовими полями і кнопкою типу Submit, в поля користувач вводить два (три) натуральні числа. Далі після натискання кнопки Submit викликається php-скрипт, який перевіряє чи числа натуральні, якщо так, то обчислює і виводить значення виразу згідно варіанту, а також введені числа:

| № п/п | $F$ | № п/п | |
|---|---|---|---|
| 1 | $F = x + 2y$ | 16 | $F = \frac{1}{x + y + z}$ |
| 2 | $F = 2x - 3y^2$ | 17 | $F = \frac{x - y}{2xz}$ |
| 3 | $F = \frac{1 - x^2}{y^3}$ | 18 | $F = 21xy - xz + xyz$ |
| 4 | $F = 2x + 3y - z$ | 19 | $F = x^3 + x^2 + x + xy^2$ |
| 5 | $F = 3xy - y + 2x$ | 20 | $F = x + y^2 + z^{0.5}$ |
| 6 | $F = \frac{x}{y} + \frac{y^2}{1 - xy}$ | 21 | $F = 2x + 3y - z$ |
| 7 | $F = x - xy + y^2$ | 22 | $F = x^6 + x^5 y + 0.5z + x^2$ |
| 8 | $F = x + \sqrt{y}$ | 23 | $F = \frac{xz}{y}$ |
| 9 | $F = xy + xz - yz$ | 24 | $F = \frac{\sqrt{x}}{y + z}$ |
| 10 | $F = x^{y^z} + 2xz$ | 25 | $F = \frac{x - y^{0.5}}{xz}$ |
| 11 | $F = x^6 + x^5 y + 0.5z + x^2$ | 26 | $F = -3xz + \sqrt{x^3}$ |
| 12 | $F = \left(1 - \frac{x}{y}\right)^2 + z$ | 27 | $F = x^{y^z} + 2xz$ |
| 13 | $F = x^y + y^z + z^x$ | 28 | $F = \frac{\sqrt{y}}{\sqrt[3]{xz}}$ |
| 14 | $F = 1 - x - y - z$ | 29 | $F = 2x + y - \frac{z}{xy}$ |
| 15 | $F = x + y^2 + z^{0.5}$ | 30 | $F = \frac{\sqrt{x}}{y + z}$ |

3. Створіть форму з двома текстовими полями і кнопкою типу Submit, в поля користувач вводить два цілі числа. Далі після натискання кнопки Submit викликається php-скрипт, який перевіряє чи число ціле, якщо так, то обчислює і виводить значення виразу згідно варіанту, а також введені числа:

| № п/п | Варіант завдання | № п/п | Варіант завдання |
|---|---|---|---|
| 1 | $F = \begin{cases} x + y, xy > 0 \\ x, x^2 > 100 \\ x^3 + y, \text{інакше} \end{cases}$ | 16 | $F = \begin{cases} x + y, xy \neq 0 \\ x, x^3 > 100 \\ x^3 + y, \text{інакше} \end{cases}$ |

| | | | |
|---|---|---|---|
| 2 | $F=\begin{cases}x+y,(x>0)та(y>0)\\x^2+y^2,(x<3)та(y<4)\\x^3,інакше\end{cases}$ | 17 | $F=\begin{cases}2x+y,(x>0)та(y>0)\\x^2+2y^2,(x<3)та(y>4)\\x^3,інакше\end{cases}$ |
| 3 | $F=\begin{cases}x^2+y^2,(x<10)або(y>3)\\x-y+x^2,x>2\\x^3+3xy,інакше\end{cases}$ | 18 | $F=\begin{cases}2x^2-3y^2,(x<10)або(y\neq 3)\\2x-3y/x+x^2,x\neq 2\\x^3+3xy,інакше\end{cases}$ |
| 4 | $F=\begin{cases}x/y+xy^2,(x>0)та(y>0)\\y/x,(-x+y)/2>0\\x^2,інакше\end{cases}$ | 19 | $F=\begin{cases}2x/y+xy^2,(x>0)та(y\neq 0)\\y/2x,(2x+y)/2\neq 0\\x^2,інакше\end{cases}$ |
| 5 | $F=\begin{cases}x-y^2,x-y>2\\x/y,x>0\\x^3-y^2,інакше\end{cases}$ | 20 | $F=\begin{cases}x-2y^2,x-3y\neq 2\\x-y/y,x>0\\2x^2-3y,інакше\end{cases}$ |
| 6 | $F=\begin{cases}x/2y,xy<-100\\x^2-xy,xy>20\\x^3+2,інакше\end{cases}$ | 21 | $F=\begin{cases}2x/y,xy\neq -100\\x^3-2xy,xy\neq 20\\x^2+4,інакше\end{cases}$ |
| 7 | $F=\begin{cases}2xy-3,x/y>0\\xy/3,(x>0)та(y<3)\\2x-3y,інакше\end{cases}$ | 22 | $F=\begin{cases}xy-1,x/y>0\\2xy/2,(x\neq 0)та(y\neq 3)\\x-y,інакше\end{cases}$ |
| 8 | $F=\begin{cases}2x+3y,x<3y\\xy+x/3y,(y^2>2xy)або(y>0)\\x^2+y,інакше\end{cases}$ | 23 | $F=\begin{cases}10x+y,x\neq y\\2xy+4x/y,(y^2\neq 2xy)або(y<0)\\x+y^2,інакше\end{cases}$ |
| 9 | $F=\begin{cases}2\sqrt{x}\cdot y,(y>3)та(x>0)\\x/y^2,(yx^2/2>3x)або(x^2>2y)\\x^2/y,інакше\end{cases}$ | 24 | $F=\begin{cases}x^3,x^3\neq 3\\xy,(x\neq 0)та(y<0)\\2x+xy-x/2y,інакше\end{cases}$ |

| | | | |
|---|---|---|---|
| 10 | $F=\begin{cases}3xy-x, xy>0\\ 2x-3y/x, (7x>2y)або(y>2)\\ x-y, інакше\end{cases}$ | 25 | $F=\begin{cases}xy-x, 2x/y\neq 0\\ (x-2y)/x, (14x\neq y)або(2y<-4)\\ x-y, інакше\end{cases}$ |
| 11 | $F=\begin{cases}3xy, x>0\\ 2x/y, (x<0)ma(y>3/2)\\ x^2+y^3, інакше\end{cases}$ | 26 | $F=\begin{cases}x/y, x\neq 0\\ 2x/y, (x\neq 0)або(y>3/2)\\ -x^3+2y^2, інакше\end{cases}$ |
| 12 | $F=\begin{cases}x-y, xy>0\\ x^2-y/x, (y<0)ma(x>3y)\\ 3xy-x^2y^2, інакше\end{cases}$ | 27 | $F=\begin{cases}2x-y, 2xy\neq 0\\ x^2+2y/x, (y\neq 0)або(x\neq 3y)\\ x/y-xy, інакше\end{cases}$ |
| 13 | $F=\begin{cases}2\sqrt{x}\cdot y, (y>3)ma(x>0)\\ x/y^2, (yx^2/2>3x)або(x^2>2y)\\ x^2/y, інакше\end{cases}$ | 28 | $F=\begin{cases}(3/5)\sqrt{x}\cdot y, (y\neq 5)ma(x\neq 0)\\ x/y^2, (yx^2\neq 3x)або(x^3\neq 5xy)\\ x/y^2, інакше\end{cases}$ |
| 14 | $F=\begin{cases}-3x+y^3, (x>0)ma(y<0)\\ x+y-x^2y, xy-x<3\\ x^2+y^2, інакше\end{cases}$ | 29 | $F=\begin{cases}x+y^2, (x\neq 0)ma(y\neq 0)\\ x-y+x^3y^2, xy-x>=3\\ x^2-y^2, інакше\end{cases}$ |
| 15 | $F=\begin{cases}3xy-x, xy>0\\ 2x-3y/x, (7x>2y)або(y>2)\\ x-y, інакше\end{cases}$ | 30 | $F=\begin{cases}2x-y, 2xy\neq 0\\ x^2+2y/x, (y\neq 0)або(x\neq 3y)\\ x/y-xy, інакше\end{cases}$ |

4. Створіть веб-сторінку, яка показує текстове повідомлення «Виберіть, будь ласка, зображення <рандомна з 4-х назва згідно теми > » та відповідні зображення як варіанти відповіді. Після натискання на малюнок скрипт PHP перевіряє правильність відповіді та інформує про результати перевірки.

Теми :

| | | | |
|---|---|---|---|
| 1. | Птахи | 2. | Меблі |
| 3. | Побутова техніка | 4. | Комп’ютерна техніка |
| 5. | Їжа | 6. | Жіночий одяг |
| 7. | Звірі | 8. | Чоловічий одяг |
| 9. | Домашні тварини | 10. | Дитячий одяг |
| 11. | Прапори | 12. | Взуття |
| 13. | Гриби | 14. | Солодощі |
| 15. | Транспорт | 16. | Спортивні аксесуари |
| 17. | Дерева | 18. | Музичні інструменти |
| 19. | Фрукти | 20. | Транспорт |
| 21. | Овочі | 22. | Професії |
| 23. | Ягоди | 24. | Гроші |
| 25. | Косметика | 26. | Кухонна техніка |
| 27. | Кондитерські вироби | 28. | Побутова техніка |
| 29. | Двоколісний транспорт | 30. | Автомобілі |

Приклад виконання:

**Зовнішній вигляд веб-сторінки**

Запитання:

Натисніть, будь ласка, на зображення кота




Оцінка правильності:

Натисніть, будь ласка, на зображення кота




Ви вибрали равлика

Це неправильно

5. Модифікуйте попереднє завдання так, щоб випадковим чином відображалися 4-ри картинки із доступних 6.

Увага! Текстове повідомлення «Виберіть, будь ласка, зображення <рандомна з 4-х назва згідно теми >» має залишатися.

6. Виконану роботу архівуєте, назва архіву має починатися із Вашого прізвища на кирилиці, даний архів здаєте у Google classroom.

## 2.3 Контрольні запитання:

1. Які типи PHP- дескрипторів існують на даний час?
2. Поясніть суть оператора у PHP.
3. Які особливості використання пробілів?
4. Які типи коментарів використовують при програмуванні на PHP?
5. Яким чином реалізують доступ до змінних форми?
6. Що таке конкатенація рядків та як вона реалізується?
7. Поясніть зміст поняття “ідентифікатор” на PHP.
8. Які типи змінних підтримує PHP?
9. Поясніть суть поняття “змінна змінних”?
10. Яким чином оголошуються константи на PHP?
11. Поясніть суть поняття “область дії змінних”?
12. Яке призначення посилань?
13. Операції порівняння на PHP.
14. Яким чином реалізують розгалуження у програмі?
15. Які оператори для створення циклів надає PHP?

# Лабораторна робота №3. Масиви у PHP. Багатократне використання коду. Створення функцій.

Тривалість: 2 акад. години
Мета: набути практичних навичок з використання масивів та багатократного використання програмного коду.
Завдання: дослідити використання масивів, створення функцій користувача та можливості включення у програмний код раніше розроблених файлів. Виконайте завдання згідно ходу роботи.

## 3.1 Теоретичні відомості

Масиви
Функції

### 3.1.1. Використання масивів у PHP

У більшості мов програмування масиви мають числові індекси, котрі, як правило, починаються з нуля чи одиниці.

PHP дозволяє використовувати як індекси числа або рядки.

Масив створюється наступним чином:

```
$a = array('data1', 'data2', 'data3');
$b = array(1, 2, 3);
```

Відповідно, доступ до елементів масиву, отримують так:

```
echo '$a[0]='.$a[0].'<br/>';
echo '$a[1]='.$a[1].'<br/>';
echo '$a[2]='.$a[2].'<br/>';
echo '$b[0]='.$b[0].'<br/>';
echo '$b[1]='.$b[1].'<br/>';
echo '$b[2]='.$b[2].'<br/>';
```

Якщо у масиві необхідно зберегти зростаючу послідовність чисел, використовують функцію `range()`. Аналогічно функція утворює послідовність символів:

```
<?
```

```
$numbers = range(1, 10);
$symbols=range('a', 'j');
for($i=0; $i<10; $i++)
echo $numbers[$i].'---'.$symbols[$i].'<br/>';
?>
```

Даний приклад генерує масив із десяти чисел у діапазоні від 1 до 10 та послідовність символів від “a” до “j”.

Для доступу до елементів масиву зручно використовувати цикл `foreach`, який призначений, власне, для роботи з масивами.

```
<?php
$arr = range(1, 20);
foreach ($arr as $current)
echo $current.'<br/>';
?>
```

Наведений код почергово зберігає кожний елемент масиву у змінній `$current` і потім виводить її значення у вікно браузера.

При створенні масивів у наведених прикладах PHP було надано можливість присвоїти кожному елементу масиву індекс по замовчуванню, тобто перший доданий елемент став 0-м, другий – 1-м і так далі. PHP також підтримує масиви, у яких з кожним елементом можна пов’язати (асоціювати) довільний ключ (індекс).

У наступному прикладі символьні назви слугують як ключі, а числа – як значення.

```
<?php
$arr = array('the_first'=>1, 'the_second'=>2,
'the_third'=>3);
echo $arr['the_first'].'<br/>';
echo $arr['the_second'].'<br/>';
echo $arr['the_third'].'<br/>';
?>
```

Створити масив з багатьма елементами можна також, оголосивши масив з одним елементом і додаючи до нього пізніше решту потрібних елементів:

```
<?php
$arr = array('the_first'=>1);
$arr['the_second']=2;
$arr['the_third']=3;
foreach ($arr as $current)
    echo $current.'<br/>';
?>
```

Оскільки в асоціативних масивах індекси не є числами, то для роботи з такими масивами неможливо скористатися лічильником у циклі `for`. У цьому випадкові слід застосувати цикл `foreach` або конструкції `list()` та `each()`.

```
<?php
$arr = array('the_first'=>1, 'the_second'=>2,
'the_third'=>3);
foreach ($arr as $current)
echo $current.'<br/>';
?>
```

Якщо необхідно отримати доступ до індексів масиву, використовують конструкцію:

```
<?php
$arr = array('the_first'=>1, 'the_second'=>2,
'the_third'=>3);
foreach ($arr as $key => $value)
echo '$arr['.$key.']='.$value.'<br/>';
?>
```

Результат:

```
$arr[the_first]=1
$arr[the_second]=2
$arr[the_third]=3.
```

Наведений нижче код виводить вміст масиву за допомогою конструкції

each():

```
<?php
$arr = array('the_first'=>1, 'the_second'=>2,
'the_third'=>3);
while( $element = each($arr))
{
echo
'$arr['.$element['key'].']='.$element['value'].'<br/>';
}
?>
```

У цьому прикладі змінна `$element`– це масив. При викликові функції `each()` вона повертає масив з чотирма значеннями і чотирма індексами. Комірки `key` та `0` містять ключ поточного елементу, `value` та `1` – його значення.

Важливо: при використанні функції `each()` слід пам'ятати, що масив відстежує поточний елемент. Якщо в одному і тому ж сценарії необхідно двічі скористатися значеннями одного і того ж масиву, то потрібно за допомогою функції `reset()` встановити поточний елемент на початок масиву.

Наведемо ряд корисних функцій для роботи з масивами.

`sort($array)` – впорядковує елементи за алфавітом чи порядком зростання;

`asort($array)` – те саме, що і sort(), але для масивів з описовими індексами;

`ksort($array)` – впорядковує ключі у масиві з описовими індексами за зростанням;

`rsort($array)` – впорядковує елементи за алфавітом чи порядком спадання;

`rasort($array)` – те саме, що і sort(), але для масивів з описовими індексами;

`rksort($array)` – впорядковує ключі у масиві з описовими індексами за спаданням.

У PHP відсутня можливість порівняння двох масивів, тому для сортування багатовимірного масиву необхідно створювати деякий ручний метод.

`shuffle($array)` – розташовує елементи масиву випадковим чином;

`array_push($array, $data)` – додає елемент `$data` у кінець масиву `$array;`

`$array = array_reverse($array)` – змінює порядок розташування елементів на зворотній;

`next($array)` або `each($array)` – переміщують показник на елемент вперед на один елемент; `next` – спочатку змінює покажчик, потім – повертає значення, `each`–навпаки;

`end($array)` – встановлює покажчик на елемент у кінець масиву;

`prev($array)` – встановлює покажчик на елемент на одну позицію ближче до початку масиву;

`current ($array)` — повертає поточний елемент масиву

`reset($array)` – встановлює покажчик масиву на його початок.

`sizeof($array)` – підраховує кількість елементів у масиві.

3.1.2 Багатократне використання коду

Одне із завдань, які зазвичай ставлять перед собою розробники програмного забезпечення – повторне використання існуючого програмного коду замість написання нового. Таким чином вдається знизити вартість розробки, підвищити надійність та забезпечити певну уніфікацію.

PHP надає два оператори `require()` та `include()` для забезпечення повторного використання програмного коду.

Оператор `require('шлях до файлу')` слугує для включення повного коду файлу-параметру оператора у тому місці файла реципієнта, в якому знаходиться сам оператор.

При цьому розширення імені файла, який включається, ігнорується, тобто він може бути названий довільним чином. Файл, що включається, повинен мати PHP-дескриптори. Інакше вміст файла буде розглянуто як простий текст.

Оператор `require()` часто застосовується для створення шаблонів сторінок.

Оператор `include()` практично не відрізняється від оператора `require()` за винятком того, що `require()` у випадку невдалого виконання видає повідомлення про непоправну помилку, а `include()` – тільки попередження.

### 3.1.3 Створення функцій

Функцією називають незалежний модуль коду, який встановлює інтерфейс введення, виконує певну задачу і за необхідності повертає результат.

Імена функцій у PHP не чутливі до регістру.

Оголошення функції створює нову функцію. Оголошення починається ключовим словом `function`, воно присвоює функції ім'я, задає необхідні параметри і містить код, котрий виконується при кожному викликові функції.

Оголошення функції має вигляд:

```
function my_function()
{
Echo 'викликано функцію';
}
```

Виклик функції реалізується за допомогою оператора `my_function()`.

Вбудовані функції PHP доступні для усіх сценаріїв, а функції користувача – тільки у тому сценарії, у якому вони оголошені. Є сенс додати усі функції, що часто використовуються, в окремий файл і за допомогою оператора `require()` надати доступ до них.

Імена функцій користувача вибираються, враховуючи обмеження:

– функція не може мати ім'я таке саме, як у вже існуючої функції;

– ім'я функції може складатися тільки з букв, цифр та символів підкреслювання;

– ім'я функції не може починатися з цифри.

Для нормального функціонування більшість функцій вимагають передачі у них параметрів. Наприклад (виводиться таблиця з 3 рядків і 1 стовпчика):

```
<?php
function create_table($data)
{
echo '<table border=1>';
reset($data); // вказуємо початок
$value=current($data);
```

```
while($value)
{
    echo "<tr><td>$value</td></tr>\n";
    $value = next($data);
}
echo '</table>';
}
$my_array = array('Рядок 1', 'Рядок 2', 'Рядок 3');
create_table($my_array);
?>
```

Як і вбудовані функції, функції користувача можуть мати обов'язкові та необов'язкові параметри. Модифікуємо попередній приклад:

```
<?php
function create_table2($data, $border=1,
$cellpadding=4, $cellspacing=4)
{
echo "<table border=$border
cellpadding=$cellpadding"
          "cellspacing=$cellspacing>\n";
reset($data); // вказуємо початок
$value=current($data);
while($value)
{
    echo "<tr><td>$value</td></tr>\n";
    $value = next($data);
}
echo '</table>';
}
$my_array = array('Рядок 1', 'Рядок 2', 'Рядок 3');
create_table2($my_array, 3, 8, 8);
?>
```

Перший параметр функції `create_table2()` як і раніше – обов'язковий. Наступні три параметри – не обов'язкові, оскільки для них визначені значення по замовчуванню. Необов'язкові параметри можна

передавати не всі, а тільки деякі з них. Параметри присвоюються зліва-направо.

Слід пам’ятати, що пропуск необов’язкового параметру і задавання наступного після нього параметру не допускається. У наведеному прикладі це означає, що за необхідності задати `$cellpadding` пропустити `$border` не можна.

Існує можливість дізнатися, скільки та які саме параметри передано.

```
<?php
function create_table2($data, $border=1,
$cellpadding=4, $cellspacing=4)
{
echo "<table border=$border
cellpadding=$cellpadding cellspacing=$cellspacing>\n";
reset($data); // вказуємо початок
$value=current($data);
while($value)
{
    echo "<tr><td>$value</td></tr>\n";
    $value = next($data);
}
echo '</table>';
echo 'Кількість параметрів: ';
echo func_num_args().'<br/>';
$args = func_get_args();
foreach ($args as $arg)
    echo $arg.'<br/>';
}
$my_array = array('Рядок 1', 'Рядок 2', 'Рядок 3');
create_table2($my_array, 3, 8, 8);
?>
```

Функція `func_num_args()` визначає, скільки аргументів було передано функції користувача, а `func_get_args()` повертає масив, який містить ці аргументи.

Звичний спосіб передачі параметрів називається передачею за значенням. При цьому зміна значення глобального параметра функції з тіла

функції неможлива. Для вирішення проблеми використовується так звана передача за посиланням, яка реалізується шляхом додавання символу амперсанда (`&`) перед іменем параметра у визначенні функції.

Ключове слово `return` завершує виконання функції і передає виконання наступному операторові після виклику функції. Також функція завершує свою роботу, коли усі оператори виконані.

Найпоширеніша причина використання оператора `return` з метою передчасного завершення функції – умова виникнення помилки. Також `return` використовують, коли функція повинна повертати результат.

PHP підтримує рекурсивні функції. Рекурсивною функцією називають функцію, котра викликає сама себе. Такі функції особливо корисні для переміщення по динамічних структурах даних, таких як зв'язані списки та дерева.

Слід пам'ятати, що рекурсивна функція створює у пам'яті копії самої себе і, відповідно, веде до непродуктивних витрат ресурсів.

Детальнішу інформацію шукайте на сайті https://www.php.net чи будь-якому іншому.

## 3.2 Хід роботи

N - номер лабораторної роботи, a - номер варіанту (див. чат Google Classroom)

Виконати наведені нижче завдання, посилання (назва повинна містити «Завдання Номер») на файл із реалізацією кожного із завдань розмістити у файлі labN.php папки labN. Усі функції повинні бути розміщені в файлі function.php.

1. Ознайомитися із теоретичними відомостями і наведеними в них прикладами. Код функції create_table2 скопіювати в файл function.php, а виклик функції здійснити з файлу labN.php.
2. Значення масу із 10 елементів згенерувати випадковим чином з діапазону він 1 до a+10 (функція mt_rand), вивести на веб-сторінку індекси і відповідні значення генерованого масиву, а також індекси та значення мінімального, максимального та середнє значення елементів.
3. У масив внесіть 5 довільних чисел. Далі використовуючи конструкцію foreach виведіть їх і їх квадрати у вигляді: $4^2=16$ $2^2=4$ $5^2=25$ і т.д.
4. Виконайте три завдання: одне завдання згідно варіанту і два наступних (тобто, якщо номер в журналі 3, то виконати завдання 3, 4, 5), дані зберігайте у масивах, значення елементів виводьте на веб-сторінку:

| № | Завдання |
|---|---|
| 1. | Знайти та роздрукувати цілі числа, кратні трьом, та їх кількість в діапазоні від 30 до 60. |
| 2. | Знайти методом підбору корінь рівняння 4*x+5=25. Пошук почати з x=1, збільшувати кожне наступне значення x на 1. |
| 3. | Знайти та роздрукувати перші 8 чисел з ряду Фібоначчі та їх суму. |
| 4. | Знайти та роздрукувати усі парні числа ряду від 0 до 20. |
| | Годинник б'є стільки разів, яка зараз година. Знайти та роздрукувати кількість ударів годинника від 00 год. до 15 год. включно. |
| 6. | Знайти та роздрукувати ряд Фібоначчі починаючи з 5-го члену ряду і закінчуючи 20-м. Підрахувати їх кількість. |
| 7. | Знайти та роздрукувати кожне п'яте ціле число в ряді -1 до 51. |
| 8. | Спортсмен у перший день подолав 10 км дистанції. Кожного наступного дня він пробігав на 10% більше, ніж попереднього. За скільки днів спортсмен подолає 50 км? |
| 9. | Знайти та роздрукувати суму всіх парних та добуток непарних цілих чисел в діапазоні від 0 до 20. |
| 10. | Знайти та вивести на екран у вигляді таблиці значення функції y=2*x при x=[0..4] з кроком 0.5. |
| 11. | Вивести на екран лінію, яка буде складатися з 20 символів «*», замінюючи кожен третій символ на знак «?». |
| | Знайти та вивести на екран у вигляді таблиці значення квадратних коренів цілих чисел в діапазоні [1..10] |
| 13. | Знайти та роздрукувати середнє арифметичне квадратів усіх цілих чисел від a до b. Прийняти a=3, b=10. |
| 14. | Дано натуральне число N=10. Знайти суму та кількість усіх членів ряду S=1+1/2+1/3+...+1/N. |
| | Вивести в таблицю квадрати й куби таких чисел: 1, 3, 5, 9, 17. |
| 16. | Одноклітинна амеба кожні 3 години ділиться на 2 клітини, вивести в таблицю, скільки буде амеб після 3, 6, 9...24 год. |
| 17. | Щомісячна стипендія студента складає 1500 грн., а загальні витрати на місяць 2000 грн., визначте, яку суму студент візьме в борг за 10 міс., якщо загальні витрати зростають на 2% кожного місяця. |
| 18. | Знайти та вивести на екран суму та кількість елементів ряду y=1*3*5*...*(2*n-1) при n=20. |
| 19. | Знайти та вивести таблицю зведення числа 2 до ступеню 1..10. |

| | |
|---|---|
| 20. | Знайти та вивести на екран суму натуральних чисел, кратних 5, в діапазоні від 500 до 1000. |
| 21. | Дано натуральне число N=200. Одержати найбільше число $4^n$<N, де n - натуральне число. |
| 22. | Кожного дня спортсмен пробігає 5 км. Який шлях пробіжить спортсмен за 30 днів, якщо кожного наступного дня він пробігає на 0, 5 км більше за попередній. |
| 23. | Вивести на екран таблицю 10 рядків на 10 стовпчиків (усього 100 комірок), і в кожній комірці вивести наступне число (від 0 до 99). |
| 24. | Яка сума накопичиться на депозитному рахунку через 5 років, якщо вносити на рахунок кожен місяць на 20% більше, ніж було внесено попереднього місяця без урахування капіталізованих відсотків. Відсотки по внеску складають 12% річних, кожного місяця відсотки капіталізуються. Початкова сума - 200 грн. |
| 25. | Знайти суму та кількість парних чисел в діапазоні від 50 до 100. |
| 26. | Визначте число n, при якому сума квадратів натурального ряду 1..n буде менша за встановлене число N, де N=50. |
| 27. | Знайти та роздрукувати в стовпчик усі непарні числа ряду від 1 до 100. |
| 28. | Знайти та роздрукувати в таблиці цілі числа, кратні п'яти, та їх кількість в діапазоні від 100 до 300. |
| 29. | Дано натуральне число n=10. Знайти кількість натуральних чисел, більше n та менше 100, які не діляться на 2 та на 3. Виведіть ці числа. |
| 30. | Знайти і вивести на екран таблицю квадратів перших п'яти цілих додатних непарних чисел. |

5. Реалізувати наступні функції (функції реалізовувати в файлі function.php, а задавати необхідні масиви і викликати функції з файлів розміщених в папці labN):

5.1. Функція, що виводить заданий масив чисел разом із індексами в заданому і оберненому порядку. Задану послідовність вказати при виклику функції.

5.2. Для двовимірного масиву NxN, де N=(a%10+1)*2, значення згенерувати випадковим чином з діапазону від 1 до 100. Створити функцію, що виводить переданий в неї двовимірний масив у вигляді таблиці і два одновимірні масиви: елементами першого є мінімальні значення рядків вхідного масиву, а другого — числа, які знаходяться в останньому стовпці.

6. Створити файл task6.php, в якому розмістити форму, яке містить однорядкове текстове поле для введення числа із підписом «Введіть натуральне число » і кнопку для передачі даних. При умові успішної передачі даних (перевірити чи введено і передано натуральне число), викликати функцію, розміщену у файлі function.php, яка приймає введене у текстове поле число N, формує і виводить масив, елементами якого є перші N натуральних чисел, піднесених до квадрату і відповідні індекси.

Приклад виклику функції при кліку на кнопку форми:

У файл task6.php записуєте:

```
<?php
require("../config.php");
include_once("function.php");
?>
<html>
<H2>PHP. Робота з функціями</H2>
<?php
$variable=$_GET['formvariable'];
if ($variable==2){
echo_fun();
}
// інший варіант: if ($_GET['formvariable']<>""),
умову пишите по потребі

?>
<form action="task6.php" method="get">
<input type="text" name="formvariable">
<input type="submit" value="Додати">
</form>
</html>

У файлі function.php:
function echo_fun(){
echo "<br>text from function echo_fun<br>";
}
```

7. Створити файл task7.php. Даний файл підключити в кінець файлу task6.php з допомогою include_once. У файл task7.php у блок PHP додати посилання на файл task7.php (тобто посилання на самого себе). У посиланні передавати число згенероване випадковим чином від 1 до 6. З допомогою оператора switch відповідно до переданого посиланням числа вивести повідомлення «Викликати функцію func1», якщо передане значення 1, func2 при 2, func3 при 3, якщо не спрацює жоден із варіантів, вивести «Некоректні дані».

8. Здати роботу.

### 3.3. Контрольні запитання

1. Яким чином створюють масиви. Ініціалізація елементів масивів.
2. Яким чином організувати доступ до елементів масиву за допомогою циклів на PHP.
3. Асоціативні масиви. Призначення. Доступ до елементів.
4. Сортування елементів масивів.
5. Призначення операторів require() та include().
6. Яким чином створюють функції на PHP?

## Лабораторна робота № 4. Асоціативні масиви в PHP. Опрацювання параметрів форми

Тривалість: 2 акад. години
Мета: ознайомитися з прийомами застосування масивів при розв’язуванні задач та опрацюванні параметрів форми.
Завдання: ознайомитися з описом масивів та функціями опрацювання масивів у PHP та виконати всі пункти згідно ходу роботи.

### 4.1 Теоретичні відомості

#### Асоціативні масиви в PHP

В PHP індексом масиву може бути не тільки число, а й рядок. Причому на такий рядок не накладаються ніякі обмежень: він може містити пробіли, довжина такого рядка може бути будь-яка.

Асоціативні масиви особливо зручні в ситуаціях, коли елементи масиву зручніше пов'язувати зі словами, а не з числами.

Отже, масиви, індексами яких є рядки, називаються асоціативними масивами.

У наступному прикладу символьні назви слугують як ключі, а числа – як значення.

```
<?php
$arr=array('the_first'=>1, 'the_second'=>2,
'the_third'=>3);
echo $arr['the_first'].'<br/>';
echo $arr['the_second'].'<br/>';
echo $arr['the_third'].'<br/>';
?>
```

Створити масив з багатьма елементами можна також, оголосивши масив з одним елементом і додаючи до нього пізніше решту потрібних елементів:

```
<?php
$arr=array('the_first'=>1);
$arr['the_second']=2;
$arr['the_third']=3;
```

```
foreach($arr as $current)
    echo $current.'<br/>';
?>
```

Оскільки в асоціативних масивах індекси не є числами, то для роботи з такими масивами неможливо скористатися лічильником у циклі for. У цьому випадкові слід застосувати цикл foreach або конструкції list() та each().

```
<?php
$arr=array('the_first'=>1, 'the_second'=>2,
'the_third'=>3);
foreach($arr as $current)
echo $current.'<br/>';
?>
```

Якщо необхідно отримати доступ до індексів масиву, використовують конструкцію:

```
<?php
$arr=array('the_first'=>1, 'the_second'=>2,
'the_third'=>3);
foreach($arr as $key => $value)
echo '$arr['.$key.']='.$value.'<br/>';
?>
```

Результат:

```
$arr[the_first]=1
$arr[the_second]=2
$arr[the_third]=3.
```

Наведений нижче код виводить вміст масиву за допомогою конструкції each():

```
<?php
$arr=array('the_first'=>1, 'the_second'=>2,
'the_third'=>3);
```

```
    while($element=each($arr))
    {
    echo
'$arr['.$element['key'].']='.$element['value'].'<br/>';
    }
    ?>
```

У цьому прикладі змінна $element– це масив. При викликові функції each() вона повертає масив з чотирма значеннями і чотирма індексами. Комірки key та 0 містять ключ поточного елементу, value та 1 – його значення.

Важливо: при використанні функції each() слід пам’ятати, що масив відстежує поточний елемент. Якщо в одному і тому ж сценарії необхідно двічі скористатися значеннями одного і того ж масиву, то потрібно за допомогою функції reset() встановити поточний елемент на початок масиву.

Багатовимірні асоціативні масиви:

Багатовимірні асоціативні масиви можуть містити кілька ключів, відповідних конкретному індексу асоціативного масиву. Розглянемо приклад багатовимірного асоціативного масиву:

```
<?php

//Багатовимірний масив
$A["Ivaniv"]=array("name"=>"Іванів І.І.", "age"=>"25",
"email"=>"ivaniv@pnu.edu.ua");

$A["Petriv"]=array("name"=>"Петрів П.П.", "age"=>"34",
"email"=>"petriv@pnu.edu.ua");

$A["Pavliv"]=array("name"=>"Павлів С.С.", "age"=>"47",
"email"=>"pavliv@pnu.edu.ua");
?>
```

Доступ до елементів багатовимірного асоціативного масиву здійснюється таким чином:

```
echo $A["Ivaniv"]["name"]; //Виводить Іванів І.І.
echo $A["Petriv"]["email"]; //Виводить
petriv@pnu.edu.ua
```

Асоціативні багатовимірні масиви можна створювати і класичним способом, хоча це не так зручно:

```
<?php

//Багатовимірний асоціативний масив
$A["Ivaniv"]["name"]="Іванів І.І.";

$A["Ivaniv"]["age"]="25";

$A["Ivaniv"]["email"]="ivaniv@pnu.edu.ua";

$A["Petriv"]["name"]="Петрів П.П.";

$A["Petriv"]["age"]="34";

$A["Petriv"]["email"]="petriv@pnu.edu.ua";

$A["Pavliv"]["name"]="Павлів С.С.";

$A["Pavliv"]["age"]="47";

$A["Pavliv"]["email"]="pavliv@pnu.edu.ua";

//Одержуємо доступ до багатовимірного асоціативного
масиву

Echo $A["Ivaniv"]["name"]."<br>";//Виводить Іванів І.І.

Echo $A["Pavliv"]["age"]."<br>";//Виводить 47

Echo $A["Petriv"]["email"]."<br>";//Виводить
petriv@pnu.edu.ua
?>
```

Видалити окремий елемент масиву можна за допомогою функції **unset(),** а перевірити існування масиву можна за допомогою функції **isset().** Визначимо масив з 10 елементів і знищимо кожен парний елемент.

```
<?php
$arr=array(9, 8, 7, 6, 5, 4, 3, 2, 1, 0);
unset($arr[0], $arr[2], $arr[4], $arr[6], $arr[8]);
//Перевіряємо чи існують елементи масиву
for($i=0 ; $i<10 ; $i++)
{
if(isset($arr[$i]))    echo    "Елемент    $arr[$i]
```

```
визначений <br/>" ;
    else echo "Елемент $arr[$i] не визначений <br/>" ;
    }
    ?>
```

Результатом роботи скрипта будуть наступні рядки

```
елемент $arr[0] не визначений
елемент $arr[1] визначений
елемент $arr[2] не визначений
елемент $arr[3] визначений
елемент $arr[4] не визначений
елемент $arr[5] визначений
елемент $arr[6] не визначений
елемент $arr[7] визначений
елемент $arr[8] не визначений
елемент $arr[9] визначений
```

За допомогою функції **unset()** можна знищити весь масив відразу.

```
<?php
$arr=array(9, 8, 7, 6, 5, 4, 3, 2, 1, 0);
unset($arr);
if(isset($arr)) echo "Массив визначений" ;
else echo "Массив не визначений" ;
?>
```

До цього масиви виводилися за допомогою циклу, проте в PHP передбачена спеціальна функція для виведення вмісту масиву **print_r().** Функція орієнтована на вивід в консольний потік, тому при виведенні результатів у вікно браузера краще оформити її тегами <pre> і </pre>:

```
<?php
$arr[]="345";
$arr[]="mail@pnu.edu.ua";
$arr[]="http://www.softtime.ua";
$arr[]="login";
```

```
$arr[]="password";
echo "<pre>";
print_r($arr);
echo "</pre>";
?>
```

Результат роботи скрипта виглядає наступним чином:

```
Array
(
[0] => 345
[1] => mail@pnu.edu.ua
[2] => http://www.softtime.ua
[3] => login
[4] => password
)
```

Операції над масивами

1) Визначення числа елементів в масиві **count():**

Створимо масив $name:

```
<?php
$name=array('Boss', 'Lentin', 'NAV', 'Endless',
'Dragons', 'SiLeNT', 'Doctor', 'Lynx');
?>
```

Щоб визначити число елементів в масиві можна поступити наступним чином:

```
<?php
echo 'Число елементів в масиві - ' . count($name);
?>
```

результат:

```
Число елементів в масиві – 8
```

2) Об'єднання масивів

a) Створимо два асоціативних масиву $a і $b:

```
<?php
$a=array(“a” => ”aa”, “b” => “bb”);
$b=array(“c” => ”cc”, “d” => “dd”);
?>
```

Нехай необхідно створити масив $c, які буде містити як елементи масиву $a так і масиву $b:

```
<?php
$a=array("a" => "aa", "x" => "xx");
$b=array("c" => "cc", "d" => "dd");
$c=$a + $b ;
echo "<pre>" ;
print_r($c);
echo "</pre>" ;
?>
```

результат:

```
Array
(
[a] => aa
[x] => xx
[c] => cc
[d] => dd
)
```

b) Створимо два числових масиву $a і $b:

```
<?php
$a=array(10, 20);
$b=array(100, 200, 300, 400, 500);
?>
```

Їх вже не вийде об'єднати за допомогою конструкції $c=$a + $b ;. Для їх об'єднання буде потрібно скористатися функцією **array_merge():**

```
<?php
$c=array_merge($a, $b);
?>
```

3) Сортування масиву

Скористаємося масивом $name:

```
<?php
$name=array('Boss', 'Lentin', 'NAV', 'Endless',
'Dragons', 'SiLeNT', 'Doctor', 'Lynx');
?>
```

Нехай потрібно впорядкувати масив в алфавітному порядку, для цього можна скористатися наступним кодом:

```
<?php
sort($name);
echo "<pre>" ;
print_r($name);
echo "</pre>" ;
?>
```

результат:

```
Array
(
[0] => Boss
[1] => Doctor
[2] => Dragons
[3] => Endless
[4] => Lentin
[5] => Lynx
[6] => NAV
[7] => SiLeNT
```

```
)
```

Нехай необхідно з масиву $name вибрати найкоротший елемент(у якого найменша кількість символів), в цьому випадку можна скористатися кодом:

```
<?php
$name=array('Boss', 'Lentin', 'NAV', 'Endless',
'Dragons', 'SiLeNT', 'Doctor', 'Lynx');
$min=strlen($name[0]);
$nam=$name[0];
for($i=1 ; $i<count($name); $i++)
{
$len=strlen($name[$i]);
if($len<$min)
{
$nam=$name[$i];
$min=strlen($nam);
}
}
echo 'Найменша довжина - ' . $nam ;
?>
```

4) Переміщення всередині масиву

Створимо масив $num:

```
<?php
$num=array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
?>
```

Нехай потрібно відобразити елементи масиву в зворотному порядку, в цьому випадку можна скористатися кодом:

```
<?php
$end=end($num);
While($end)
{
echo $end . ' - ' ;
```

```
$end=prev($num);
}
?>
```

результат:

```
10 - 9 - 8 - 7 - 6 - 5 - 4 - 3 - 2 - 1
```

Наведений вище код можна модифікувати:

```
<?php
$num=range(1, 10);
print_r(array_reverse($num));
?>
```

Функція **range(1, 10)** створює масив(1, 2, 3, 4, 5, 6, 7, 8, 9, 10) з випадковим розташуванням елементів. Функція **array_reverse()** приймає масив і повертає елементи в зворотному порядку(10, 9, 8, 7, 6, 5, 4, 3, 2, 1)

Функція **reset()** повертає покажчик на перший елемент в масиві. Скористаємося масивом $num:

```
<?php
$num=array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);
?>
```

Нехай необхідно вивести всі елементи по порядку, і на останньому елементі масиву повернути покажчик на перший елемент масиву. Цю операцію можна здійснити за допомогою наступного коду:

```
<?php
$i=0 ; //Індекс 1 елемента
while($i<count($num))
{
echo $num[$i]. ' ' ;
$i++;
//Перевірка якщо $i рівний числу елементів в масиві
//тоді виводимо останній елемент і повертаємо
```

```
вказівник
    if($num[$i] == count($num))
    {
    echo $num[$i];
    reset($num);
    echo '<br/>' . "Кінець масиву" ;
    exit();
    }
    }
    ?>
```

результат:

```
1 2 3 4 5 6 7 8 9 10

Кінець масиву
```

5) Перемішування елементів в масиві **shuffle()**

Функція **shuffle()** перемішує значення в масиві, і якщо масив асоціативний то повертає його як список:

```
<?php
$a=array(43, 'PHP', 4, 57, 'Boss', 90);
shuffle($a);
foreach($a as $n) echo "$n " ;
?>
```

6) Випадковий елемент масиву

Якщо є готовий масив, з якого необхідно вивести один випадковий елемент, для цього необов'язково перемішувати весь масив за допомогою функції **shuffle(),** досить згенерувати випадковий індекс масиву:

```
<?php

$arr=array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);

$index=rand(0, count($arr) - 1);
```

```
echo $arr[$index];
?>
```

7) Отримання частини масиву **array_slice()**

Створимо масив $a

```
<?php
$a=array('a', 'b', '3', '5', 'f');
?>
```

Отримати частину масиву можна за допомогою наступного коду:

```
<?php
$b=array_slice($a, 2) //вивід 3, 5, f
$b=array_slice($a, 0, 3) //a, b, 3
?>
```

8) Серіалізация масиву

Функції **serialize()** і **unserialize()** дозволяють здійснювати упакування і розпакування, відповідно, масивів і об'єктів.

Зауваження. Серіалізація вперше з'явилася в об'єктно-орієнтованих бібліотеках, (першої з яких була MFC), потім серіалізація стала з'являтися в об'єктно-орієнтованих мовах (Java). Ідея серіалізації полягає в тому, що об'єкти і масиви дуже складні за своєю структурою і на збереження їх шляхом перебору кожного елемента потрібно значний обсяг коду - найпростішим рішенням є збереження таких структур у вигляді єдиної закодованої послідовності - байт-коді. У PHP функції серіалізації упаковують дані не у вигляді байт-коду, а у вигляді рядка.

```
<?php
$poll[0]=23 ;
$poll[1]=45 ;
$poll[2]=34 ;
$poll[3]=2 ;
$poll[4]=12 ;
//Упаковуємо масив в рядок
```

```
$str=serialize($poll);
echo $str . "<br/>" ;
//Розпаковуємо масив із рядка
$arr=unserialize($str);
print_r($arr);
?>
```

результат:

```
a:5:{i:0;i:23;i:1;i:45;i:2;i:34;i:3;i:2;i:4;i:12;}

Array
(
[0] => 23
[1] => 45
[2] => 34
[3] => 2
[4] => 12
)
```

Ви можете ознайомитися з усіма функціями для роботи з масивами у PHP в довіднику функцій.

**Передавання параметрів форми**

Спочатку напишемо HTML-документ, який буде містити практично всі елементи HTML-форми Параметри форми будемо передавати скрипту для наступної обробки. Отже, лістинг HTML-документа send.html:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1251">
<title> Test </title>
</Head>
<body>
<h3> Тестова форма </h3>
<form          name="form1"          method="post"
```

```
action="script.php">
    <p> <span> Текстове поле: </span>
    <input type="text" name="textfield">
    </p>
    <p> Поле введення пароля:
    <input type="password" name="pswfield">
    </p>
    <p> Приховане поле hidden
    <input   name="hidden"   type="hidden"   id="hidden"
value="Приховане_значення">
    </p>
    <hr size="1">
    <p> Незалежні перемикачі(checkbox): </p>
    <p>
    <input type="checkbox" name="checkbox1" value="1">
    Варіант перший
    <input type="checkbox" name="checkbox2" value="1">
    Варіант другий
    <input  type="checkbox"  name="checkbox3"  value="1"
checked>
    Варіант третій(за замовчуванням) </p>
    <hr size="1">
    <p> Залежні перемикачі(radio): </p>
    <p>
    <input name="radio_button" type="radio" value="yes">
    Так
    <input name="radio_button" type="radio" value="no">
    Ні </p>
    <hr size="1">
    <p> Багаторядкове текстове поле(textarea): </p>
    <p>
    <textarea name="textarea" cols="40" rows="10"> Текст
за умовчанням </textarea>
    </p>
    <hr size="1">
    <p> Список з єдиним вибором: </p>
    <p>
    <select name=day_s size=1>
```

```
<option value=1> понеділок </option>
<option value=2> вівторок </option>
<option value=3 selected> середа </option>
<option value=4> четвер </option>
<option value=5> п'ятниця </option>
<option value=6> субота </option>
<option value=7> неділя </option>
</select>
</p>
<p> Список із множинним вибором(multiple): </p>
<p>
<select name=day_m[] size=7 multiple>
<option value=1 selected> понеділок </option>
<option value=2> вівторок </option>
<option value=3> середа </option>
<option value=4> четвер </option>
<option value=5> п'ятниця </option>
<option value=6> субота </option>
<option value=7> неділя </option>
</select>
</p>
<hr size="1">
<p>
<input type="submit" value="Надіслати форму">
<input type="reset" value="Очистити форму">
</p>
</form>
</body>
</html>
```

Коли користувач натискає кнопку "Надіслати форму", браузер передасть скрипту наступні параметри:

• textfield - значення текстового поля;

• pswfield - значення поля введення пароля;

• hidden - значення прихованого поля;

• параметри checkbox: checkbox1, checkbox2 і checkbox3 будуть передані тільки в тому випадку, якщо відповідні їм незалежні перемикачі активні;

• radio_button - значення групи radio(буде передано одне із значень: Yes або No);

• textarea - вміст багаторядкової текстової області;

• day_s - значення списку з єдиним вибором;

• day_m - значення списку із множинним вибором.

Тепер перед нами стоїть завдання обробки всіх параметрів переданої форми за допомогою PHP скрипта.

Параметри textfield, pswfield і textarea обробляються досить просто. Наприклад, для відображення значення параметра textfield достатньо написати в обробному скрипті: echo $_POST['textfield'];

З параметрами checkbox1, checkbox2, checkbox3, і radio_button справа дещо складніша. Якщо перемикач не активний, то перелічені параметри взагалі не будуть передані на сервер, ніби їх взагалі не було.

Отже, при спробі звернутися в скрипті до цих параметрів, ми отримаємо повідомлення, що змінна не існує. Тому просто написати echo $_POST['checkbox1']; ми не можемо, нам необхідно спочатку перевірити існування цих параметрів в запиті.

Перевірка існування параметра здійснюється за допомогою функції isset(), яка служить для перевірки існування змінних.

```
    If (isset($_POST['checkbox1'])) echo
$_POST['checkbox1'];
    If (isset($_POST['radio_button'])) echo
$_POST['radio_button'];
```

Тільки після перевірки існування перерахованих параметрів форми можна починати роботу з змінними.

Складніше обробляти параметри списку із множинним вибором, оскільки в цьому випадку параметри передаються так:

```
day_m=01 & day_m=03 & day_m=07 ...
```

Ми опинилися в ситуації, коли один параметр має кілька значень. Це нагадує нам масив даних. В цьому випадку множинний список можна представити у вигляді масиву, а обробити його елементи за допомогою циклу

foreach. Навіть не обов'язково знати кількість елементів множини списку. Потрібно лише попередньо дати зрозуміти транслятору PHP, що ми будемо передавати масив:

```
<select name="day_m[]" size=7 multiple>
```

Квадратні дужки[] - це ознака масиву. Циклічна обробка масиву здійснюється так:

```
foreach($_POST['day_m'] as $key => $value)
echo "$key=$value ";
```

А тепер наведемо остаточний лістинг PHP скрипта, обробника нашої тестової форми:

```
<?php
//Виводимо HTML-заголовки:
echo '<html>';
echo '<head>';
echo        '<meta        http-equiv="Content-Type"
content="text/html; charset=windows-1251">';
echo '<title> Test </title>';
echo '</head>';
echo '<body>';
echo '<h3> Тестова форма </h3>';
echo "<p> Передане значення текстового поля: <b>".
$_POST['textfield']. "</b> </p>";
echo "<p> Передане значення поля пароля: <b>".
$_POST['pswfield']. "</b> </p>";
echo "<p> Передане значення прихованого поля hidden:
<b>". $_POST['hidden']. "</b> </p>";
echo '<hr size="1">';
echo   '<p>  Були  включені  наступні  незалежні
перемикачі: </p>';
if(isset($_POST['checkbox1'])) echo "<p> <b> Перший
</b> </p>";
if(isset($_POST['checkbox2'])) echo "<p> <b> Другий
```

```
</b> </p>";
    if(isset($_POST['checkbox3'])) echo "<p> <b> Третій
</b> </p>";
    echo '<hr size="1">';
    if(isset($_POST['radio_button']))
    {
    echo '<p> Був обраний незалежний перемикач з
наступним значенням:';
    if($_POST['radio_button'] === "yes") echo "<b> Yes
</b>";
    if($_POST['radio_button'] === "no") echo "<b> No
</b>";
    echo '</p>';
    }
    else echo '<p> Жоден з незалежних перемикачів не був
обраний </p>';
    echo '<hr size="1">';
    echo '<p> Значення багаторядкового текстового поля:
</p>';
    echo "<p> <b>".$_POST['textarea']. "</b> </p>";
    echo '<hr size="1">';
    echo "<p> Значення списку з єдиним вибором: <b>".
$_POST['day_s']. "</b> </p>";
    echo '<hr size="1">';
    echo '<p> Значення списку із множинним вибором:
</p>';
    foreach($_POST['day_m'] as $keys => $values)
     echo "<b> $values </b> <br>";
    echo '<hr size="1">';
    echo '</body>';
    echo '</html>';
    ?>
```

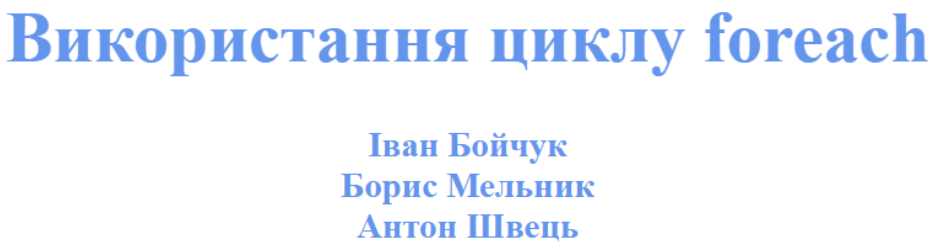
## 4.2 Хід роботи

1. Заповніть в циклах перший масив квадратами чисел від 10 до 20, а другий - кубами чисел від 1 до 10. Далі об'єднайте ці масиви і виведіть об'єднаний масив.

2. Розгляньте та реалізуйте використання циклу foreach для перебору та виведення елементів асоціативного масиву:

```
<html>
<body>
<h1> Використання циклу foreach </h1> <?php
$names["Бойчук"]="Іван";
$names["Мельник"]="Борис";
$names["Швець"]="Антон";
foreach ($names as $key => $value) {
echo "<b>$value $key</b><br>";}
?>
</body>
</html>
```

Приклад виводу:




3. В окремому файлі напишіть код створення масиву $my_topic (згідно теми із Лабораторної роботи 2) використовуючи конструкцію

```
$ім'я_масиву=array (індекс1 => значення1, індекс2 =>
значення2, ...)
```

Індексація елементів масиву повинна починатися з цифри 2. Масив повинен містити назви чотирьох елементів згідно Вашої теми. Виведіть індекси та елементи масиву. Поміняйте місцями індекси елементів масиву та їх значення. Організуйте вивід даних таким чином, щоб спочатку виводився індекс масиву, а потім член масиву.

4. Створіть і заповніть багатовимірний асоціативний масив, який містить наступні дані по трьох країнах: назва, столиця, населення (дані придумайте самостійно).

Приклад реалізації одного із елементів багатовимірний асоційований масив:

```
$country["Ukraine"]=array("name"=>"Україна",
"capital"=>"Київ", "popul"=>"45");
```

Використовуючи конструкцію **foreach** (для багатовимірного масиву одну конструкцію **foreach** вкладену в іншу)**,** виведіть з масиву:

- таблицю, яка містить три стовпці. Заголовки стовпців: «Назва», «Столиця», «Населення, млн.ч.». Кожен із стовпців містить відповідні дані по заданих країнах; К-сть рядків залежить від кількості заданих країн.
- три речення, типу «Столиця України — Київ, населення —45 млн. ч.;
- ключі першого та другого рівнів і їх значення, попередньо посортувавши елементи масиву $country:

```
Ukraine:
name=>Україна
capital=>Київ
popul=>45;
```

- виведіть даний масив у вікно браузера, використовуючи функцію **print_r().**

5. Використовуючи асоціативні масиви виконайте завдання згідно варіанту.

| № | Завдання |
|---|---|
| 1. | • У масиві з 10 елементів задана вага спортсменок двох різних команди (в кілограмах). Знайти кількість спортсменок в кожній команді, чия вага перевищує 50 кг, але не більше 57 кг. Якщо таких спортсменок немає - видати повідомлення.<br>• Визначте масив, який буде зберігати інформацію про три населених пункти (одне з них Ваше місце народження), три вулиці в них та кількість мешканців на кожній вулиці. Виведіть на екран дану інформацію у вигляді таблиці. Інформацію про вулицю з мінімальною кількістю мешканців позначте червоним кольором. Визначте і виведіть кількість мешканців у кожному населеному пункті. |
| 2. | • Сформуйте масив з трьох речень. Поміняйте місцями перше і третє речення місцями. Виведіть на екран вихідний та змінений масив.<br>• Визначте масив, який буде зберігати інформацію про три області (одна з них Ваше місце народження), три міста в них та кількість мешканців на кожному місті. Виведіть на екран дану інформацію у |

| | |
|---|---|
| | вигляді таблиці. Інформацію про місто з максимальною кількістю мешканців позначте синім кольором. Визначте і виведіть кількість мешканців у кожній області. |
| 3. | • Визначте масив, який буде зберігати інформацію про 5 десертів з різною вартістю та кількістю калорій. Дайте відповідь на питання, чи є найдорожчий десерт самим калорійним?<br>• Визначте масив, який буде зберігати інформацію про три населених пункти (одне з них Ваше місце народження), три вулиці в них та кількість мешканців на кожній вулиці. Виведіть на екран дану інформацію у вигляді таблиці. Інформацію про вулицю з максимальною кількістю мешканців позначте зеленим кольором. Визначте і виведіть кількість мешканців у кожному населеному пункті. |
| 4. | • Визначте масив, що містить інформацію про авторів (прізвище, ім'я) та кількість книг, яку кожен опублікував. Результат виведіть у вигляді таблиці. Дайте відповідь на питання, скільки авторів опублікувало більше двох книг.<br>• Визначте масив, який буде зберігати інформацію про три області (одна з них Ваше місце народження), три міста в них та кількість мешканців на кожному місті. Виведіть на екран дану інформацію у вигляді таблиці. Інформацію про місто з мінімальною кількістю мешканців позначте синім кольором. Визначте і виведіть кількість мешканців у кожній області. |
| 5. | • У масиві з 10 елементів задана вага спортсменок двох різних команди (в кілограмах). Знайти кількість спортсменок в кожній команді, чия вага перевищує 50 кг, але не більше 57 кг. Якщо таких спортсменок немає - видати повідомлення.<br>• Визначте масив, що містить інформацію про трьох студентів (прізвище) та бали з 4-х предметів, які кожен вивчив. Результат виведіть у вигляді таблиці. Інформацію про студента (-тів) з найвищою оцінкою за предмет позначте синім кольором. Визначте і виведіть середній бал кожного студента. |
| 6. | • У масиві з 10 елементів задана вага спортсменок двох різних команди (в кілограмах). Знайти кількість спортсменок в кожній команді, чия вага перевищує 50 кг, але не більше 57 кг. Якщо таких спортсменок немає - видати повідомлення.<br>• Визначте масив, який буде зберігати інформацію про три населених пункти (одне з них Ваше місце народження), три вулиці в них |

| | |
|---|---|
| | та кількість мешканців на кожній вулиці. Виведіть на екран дану інформацію у вигляді таблиці. Інформацію про вулицю з максимальною кількістю мешканців позначте зеленим кольором. Визначте і виведіть кількість мешканців у кожному населеному пункті. |
| 7. | • Сформуйте масив з трьох речень. Поміняйте місцями перше і третє речення місцями. Виведіть на екран вихідний та змінений масив.<br>• Визначте масив, що містить інформацію про трьох студентів (прізвище) та бали з 4-х предметів, які кожен вивчив. Результат виведіть у вигляді таблиці. Інформацію про студента (-тів) з найнижчою оцінкою за предмет позначте червоним кольором. Визначте і виведіть сумарний бал кожного студента. . |
| 8. | • Визначте масив, який буде зберігати інформацію про 5 десертів з різною вартістю та кількістю калорій. Дайте відповідь на питання, чи є найдорожчий десерт самим калорійним?<br>• Визначте масив, який буде зберігати інформацію про три області (одна з них Ваше місце народження), три міста в них та кількість мешканців у кожному місті. Виведіть на екран дану інформацію у вигляді таблиці. Інформацію про місто з максимальною кількістю мешканців позначте синім кольором. Визначте і виведіть кількість мешканців у кожній області. |
| 9. | • Визначте масив, що містить інформацію про авторів (прізвище, ім'я) та кількість книг, яку кожен опублікував. Результат виведіть у вигляді таблиці. Дайте відповідь на питання, скільки авторів опублікувало більше двох книг.<br>• Визначте масив, який буде зберігати інформацію про три населених пункти (одне з них Ваше місце народження), три вулиці в них та кількість мешканців на кожній вулиці. Виведіть на екран дану інформацію у вигляді таблиці. Інформацію про вулицю з максимальною кількістю мешканців позначте фіолетовим кольором. Визначте і виведіть кількість мешканців у кожному населеному пункті. |
| 10. | • Визначте масив, що містить інформацію про студентів (прізвище, ім'я) та бали з 5 предметів, які кожен вивчив. Результат виведіть у вигляді таблиці. Визначте середній бал кожного студента.<br>• Визначте масив, що містить інформацію про трьох студентів (прізвище) та бали з 4-х предметів, які кожен вивчив. Результат виведіть у вигляді таблиці. Інформацію про студента (-тів) з найнижчою оцінкою |

| | |
|---|---|
| | за предмет позначте зеленим кольором. Визначте і виведіть сумарний бал кожного студента. |

6. У HTML формі користувач вводить у п’ять різних полів: прізвище, ім'я, e-mail, пароль і повторити пароль. Після натискання кнопки ***“Готово”*** запускається PHP-скрипт, який вносить ці дані в асоціативний масив і далі виводить їх в таблицю, використовуючи конструкцію **foreach**. Перед виводом передбачити перевірку заповнення всіх полів і співпадіння паролів, якщо перевірку не пройдено, вивести відповідне повідомлення.

7. Створити анкету, яка містить чотири запитання, кожне з яких має різні види варіантів відповіді: радіокнопки, перемикачі, списки єдиного і множинного вибору. Створити скрипт, який перевіряє наявність відповіді на кожне із запитань і виводить текст запитання і вибрані до них відповіді, у випадку відсутності відповіді виводить в вікно браузера відповідне повідомлення.

## 4.1 Контрольні запитання

1. Наведіть приклад одного з можливих варіантів заповнення масиву значеннями у PHP.
2. Наведіть приклад одного з можливих варіантів заповнення асоційованого масиву значеннями у PHP.
3. Поясніть схему роботи оператора foreach у PHP.
4. Поясніть механізм передачі значень за посиланнями.
5. Наведіть приклади функцій, які можуть бути використані для опрацювання масивів у PHP.

## Лабораторна робота №5. Робота з файлами. Робота з текстом.

Тривалість: 2 акад. години

Мета: набути практичних навичок для роботи з файлами і текстом.

Завдання: дослідити можливості PHP для роботи з файлами і текстом.

### 5.1 Теоретичні відомості

Тема 7. Збереження і обробка даних в PHP

Тема 9. Робота з рядками

**Функції для роботи з багатобайтовими рядками**

#### 5.1.1. Робота з файлами

Робота з файлами - важливий інструмент PHP.

Включення зовнішніх файлів

У кожен PHP-документ можна включити файл за допомогою спеціальної конструкції мови include . Її аргумент: шлях до файлу. Цією конструкцією зручно користуватися при наявності однакових шматків коду в багатьох PHP-програмах. Вміст файлу, що включається, обробляється як простий HTML-текст. Для того, щоб вміст цього файлу оброблявся як PHP-програма, потрібно оточувати його відкриваючими і закриваючими тегами PHP.

Приклад 1

```
<html>
<head>
<title>Використання include</title>
</head>
<body>
<?php
include 'top.php';
echo "<H2> ...Основна частина... </H2>";
?>
</body>
</html>
```

Вміст файлу top.php з PHP-кодом:

```
echo “<H1> ... Загальне вітання ... </H1>”;
```

Результат прикладу 1:

... Загальне вітання ...
...Основна частина...

Якщо файл top.php містить тільки рядок HTML-тексту, результат буде тим же:

<H1> ... Загальне вітання ... </H1>

Файли, що включаються, можуть повертати значення подібно до функцій. Використання оператора return перериває виконання цього файлу так само, як і функції.

Приклад 2:

```
<html>
<head>
<title> Використання include, що повертає значення
</title>
</head>
<body>
<?php
$res = include 'top.php';
echo "<H2> Включений файл повернув $res</H2>";
?>
</body>
</html>
```

Вміст включеного файлу top.php з PHP-кодом:

```
<?php $a = 7 * 8; return $a; ?>
```

Результат прикладу 2:

```
Включений файл повернув 56
```

Інструкцію include можна використовувати всередині циклу. У циклі

include виконується при кожній ітерації. Це можна використовувати для включення декількох файлів, наприклад:

```
for($i1; $i<=5; $i++) include "incfile{$i}.html";
/*тут використано подвійні лапки замість одинарних, щоб замість змінної $i підставилися її значення, приклад назви файлу incfile1.html, incfile2.html і так до 5*/
```

Визначення імені файлу, що включається і його завантаження виконуються повторно при кожному виклику include. Це означає, що якщо вміст файлу, що включається, з моменту попереднього виклику змінився, то завантажиться новий зміст.

Конструкцію include також можна включати в тіло умовного оператора.

Конструкція include не є функцією, а є спеціальною конструкцією мови.

Для вказівки того, що файл потрібно підключати тільки один раз використовується оператор include_once

Дія констукції require аналогічна include, крім того, що у разі виникнення помилки вона видасть фатальну помилку рівня E_COMPILE_ERROR. Іншими словами, вона зупинить виконання скрипту, тоді як include тільки видасть попередження E_WARNING, яке дозволить продовження виконання скрипту.

Аналіз файлів

PHP містить множину функцій, що надають інформацію про файли. Детальнішу інформацію про наведені нижче та інші функції для роботи із файлами шукайте тут. Найбільш вживаними є:

`file_exists()` - визначає існування файлу, наприклад:

```
if(!file_exists( "aaa.php"))
echo "Увага! Файл aaa.php не знайдений!";
```

`is_file()` - визначає, чи є досліджуваний об'єкт файлом, наприклад:

```
if(is_file( "bbb.txt"))
echo "Поза всяким сумнівом, bbb.txt - це файл";
```

`is_dir()` - визначає, чи є досліджуваний об'єкт каталогом,

наприклад:

```
if(is_dir( "/tmp"))
echo "Дійсно, /tmp - це каталог";
```

`is_readable()` - визначає, чи доступний файл для читання, наприклад:

```
if(is_readable( "db.dbf"))
echo "db.dbf можна читати";
```

`is_writable()` - визначає, чи доступний файл для запису, наприклад:

```
if(is_writable("db.dbf")) echo "В db.dbf писати можна";
```

`filesize()` - визначає розмір файлу в байтах.

`filemtime()` - визначає дату і час останньої зміни файлу.

`fileatime()` - визначає дату і час останнього звернення до файлу.

Час повертається в форматі Unix, тобто являє собою кількість секунд, що пройшли після 1 січня 1970. У прикладі 3 це число перетворюється в зрозумілий для людини формат за допомогою функції date().

Приклад 3

```
<html>
<head>
<title> Інформація про файл </title>
</head>
<body>
<?php $file = "top.php";
infoFile( $file );
function infoFile( $f ){
if( !file_exists( $f ) ) { echo "$f не знайдений!";
return; }
echo "$f - ".( is_file( $f ) ? "" : "не ")."файл<br>";
```

```
echo "$f - ".( is_dir( $f ) ? "" : "не
").”каталог<br>";
echo "$f ".( is_readable( $f ) ? "" : "не ")."доступний
для читання<br>";
echo "$f ".( is_writable( $f ) ? "" : "не ")."доступний
для запису<br>";
echo "размір $f в байтах - ".(filesize($f ))."<br>";
echo "остання зміна $f - ".( date( "d MYH:i",
filemtime( $f ) ) )."<br>";
echo "останнє звернення до $f - ".( date( "d MYH:i",
fileatime( $f ) ) )."<br>"; }
    ?>
    </body>
    </html>
```

Результат прикладу 3:

```
top.php - файл
top.php - не каталог
top.php доступний для читання
top.php доступний для запису
размір top.php в байтах - 32
остання зміна top.php - 16 Feb201915:34
останнє звернення до top.php - 16 Feb201915:34
```

Управління файлами

PHP містить множину функцій управління файлами. Найбільш вживаними є:

`touch()` - створює порожній файл з заданим ім'ям. Якщо такий файл вже існує, то функція змінить дату модифікації, наприклад:

```
touch( "ex1.txt");
```

`copy()` - копіює файл.

Опис ¶

```
copy(string $from, string $to, ?resource $context =
null): bool
```

Копіює файл $from у файл з ім'ям $to.

Якщо потрібно перейменувати файл, використовуйте функцію rename().
Приклад застосування функції copy:

```
<?php
$file = 'example.txt';
$newfile = 'example.txt.bak';

if (!copy($file, $newfile)) {
 echo "не вдалося скопіювати $file...\n";
}
?>
```

unlink() - видаляє заданий файл, наприклад:

```
 <?php if(unlink('filename.txt')) {
echo "Файл видалений";
} else {echo "Помилка видалення файлу"; }
?>
```

fopen() - відкриває локальний або віддалений файл і повертає покажчик на нього. Покажчик використовується в усіх операціях з вмістом файла. Аргументи: ім'я файлу і режим відкриття.

r читання. Покажчик файлу встановлюється на його початок

r+ читання і запис. Покажчик файлу встановлюється на його початок

w запис. Покажчик файлу встановлюється на його початок. Весь старий вміст файлу втрачається. Якщо файл з вказаним ім'ям не існує, функція намагається його створити

w+ читання і запис. Покажчик файлу встановлюється на його початок. Весь старий вміст файлу втрачається. Якщо файл з вказаним ім'ям не існує, функція намагається його створити

a запис. Покажчик файлу встановлюється в його кінець. Якщо файл з вказаним ім'ям не існує, функція намагається його створити

a+ читання і запис. Покажчик файлу встановлюється в його кінець. Якщо файл з вказаним ім'ям не існує, функція намагається його створити

Наприклад:

```
    $fp = fopen( "http://www.php.net/", "r" ); // для читання
    $fp = fopen( "ex1.txt", "w" ); // для запису, покажчик файлу встановлюється на його початок
    $fp = fopen( "ex2.txt", "a" ); // для запису, покажчик файлу встановлюється на його початок
```

Якщо відкрити файл не вдалося, то можна перервати виконання програми, наприклад:

```
    $fp = fopen( "ex1.txt", "w") or die( "Не вдалося відкрити файл");
```

fclose() - закриває файл. Аргумент: покажчик файлу, отриманий раніше від функції fopen(). Наприклад:

```
    fclose($fp);
```

feof() - перевірка кінця файлу. Аргумент: покажчик файлу.

fgetc() - читання чергового символу з файлу. Аргумент: покажчик файлу.

fgets() - читання чергового рядка файлу. Аргументи: покажчик файлу і зчитування довжина рядка. Операція припиняється або після зчитування зазначеної кількості символів, або після виявлення кінця рядка або файлу.

Приклад 4

```
<html>
<head>
<title> Читання рядків з файлу </title>
 </head>
<body>
 <?php $fp = fopen( "ex1.txt", "r" ) or die( "Не вдалося відкрити файл!" );
 while( ! feof( $fp ) ) echo( fgets( $fp, 1024 ) )."<br>";
?>
</body>
```

```
 </html>
```

fread() - загальна функція читання з файлу. Аргументи: покажчик файлу і кількість зчитувальних символів.

fseek() - відступ від початку файлу. Аргументи: покажчик файлу і зсув.

Приклад 5

```
<html>
<head>
<title> Виведення на екран другої половини файлу</title>
</head>
<body>
<?php $f = "ex1.txt"; $fp = fopen( $f, "r" ) or die( "Не вдалося відкрити $f" );
$fsize = filesize( $f );
$half =(int)( $fsize / 2 );
fseek( $fp, $half );
echo( fread( $fp,($fsize - $half) ) );
?>
</body>
</html>
```

fputs() - запис рядка в файл. Аргументи: покажчик файлу і рядок.

fwrite() - повний аналог функції fputs().

Приклад 6

```
<html>
<head>
<title> Запис і додавання в файл </title>
</head>
<body>
<?php $fp = fopen( "ex2.txt", "w" ) or die( " Не вдалося відкрити файл " );
fputs( $fp, " Запис в файл \n" );
```

```
    fclose( $fp );
    $fp = fopen( "ex2.txt", "a" ) or die( " Не вдалося
відкрити файл " );
    fputs( $fp, " Додавання в кінець файлу " );
    fclose( $fp ); ?>
    </body>
    </html>
```

flock() - блокує файл, тобто не дозволяє іншим користувачам читати цей файл або писати в нього, поки той, хто наклав блокування не завершить роботу з даним файлом.

Аргументи: покажчик файлу і номер режиму блокування.

1 Можна читати, не можна писати

2 Не можна ні читати, ні писати

3 Розблокування

Приклад 7

```
    <html>
    <head>
    <title> Блокування файлу </title>
    </head>
    <body>
    <?php $fp = fopen( "ex1.txt", "a" ) or die( " Не
вдалося відкрити файл " );
    flock( $fp, 2 ); // Повне блокування
    fputs( $fp, " Запис в файл \n" );
    flock( $fp, 3 ); // Розблокування
    fclose( $fp );
    ?>
    </body>
    </html>
```

Блокування за допомогою flock() не є абсолютним. Нею будуть блокуватися лише ті програми, які теж використовують цю функцію.

file_get_contents() - прочитати весь файл або URL

file_put_contents() - записати у файл

Функція file() читає файл і повертає його вміст у вигляді масиву. Кожен елемент масиву відповідає одному рядку.

```
<?php
$arr = file( "text.txt" );
for( $i = 0 ; $i < count( $arr ); $i++)
{
echo $arr [ $i ]. "<br />" ;
}
?>
```

Завантаження файлу на сервер

Для локального серверу:

У вашому "php.ini" повинні бути наступні три параметри:

File_uploads = On — Включаємо підтримку завантаження.

Upload_tmp_dir = ПОЛНИЙ_ШДЯХ_ДО_ПАПКИ_ДЕ_БУДУТЬ_ЗБЕРІГАТИСЯ_ЗАВАНТАЖЕНІ (Тимчасові) _ФАЙЛИ

Upload_tmp_dir Наприклад = D:/сервер/PHP/завантаження

Upload_max_filesize = 2M — Максимальний розмір файлів (в нашому випадком 2 МБ).

Приклад 8.

Реалізація завантаження файлу, назва файлу upload.php, у поточній папці має бути створеним каталог files для збереження завантажених файлів:

```
<?php
if(!empty($_GET["subdir"])) {echo "<p>Ім'я переданої змінної ".$_GET["subdir"];} else {echo "<p>zminna subdir not found</p>";}
$subdir=$_GET["subdir"];
echo "subdir=$subdir";
$dir = "";
$result = move_uploaded_file($_FILES[ufile][tmp_name], $dir . $_FILES[ufile][name]);
```

```
    echo
    "<div align='center'>
    <form action='upload.php?subdir=$subdir'
method='post' enctype='multipart/form-data'>
    <input type='file' name='uploadfile'>
    <input type='submit' value='Завантажити'>
    </form></div>
    <hr/>";
    $uploaddir = "./files/$subdir/";
    echo "<p>uploaddir=$uploaddir";
    $uploadfile =
$uploaddir.basename($_FILES['uploadfile']['name']);
    echo "<p>uploadfile=$uploadfile";
    if(copy($_FILES['uploadfile']['tmp_name'],$uploadfi
le))
    {
    echo "<p>Файл завантажений на сервер";
    }
    else {echo "<p>Файл не завантажений на сервер!";}
    echo "<p>Оригінальна
назва".$_FILES['uploadfile']['name']."</p>";
    echo "<p>Тип
файлу".$_FILES['uploadfile']['type']."</p>";
    echo
"<p>Розмір".$_FILES['uploadfile']['size']."</p>";
    echo "<p>Тимчасове ім'я "
.$_FILES['uploadfile']['tmp_name']."</p>";
    ?>
```

Під час завантаження файлу в Unix-подібні операційні системи в кінці рядків можуть відображатися символи ^M. Для їх видалення в командному рядку записуємо:

```
cat name_file_in.php|tr -d '\r' > name_file_out.php
```

## 5.2 Хід роботи

Виконати наведені нижче завдання, посилання (текст посилання повинна містити «Завдання Номер») на файл із реалізацією кожного із

завдань розмістити у файлі last_name_lab5.php папки lab5.

1. Ознайомитися із теоретичними відомостями роботи з файлами і реалізувати як мінімум п’ять наведених прикладів, обов'язково реалізувати приклад 8, у якому файли повинні завантажуватися на хостинг в папку lab5/files.

2. Використовуючи PHP-скрипт і форму в рамках одного документа створити сценарій, в якому користувач буде вводити у текстове поле ім'я файла і після натискання кнопки “***Готово”*** виконається перевірка, чи існує такий файл у поточному каталозі. Якщо він не існує, виведеться повідомлення: ***“Файл з іменем Zrazok.txt у поточному каталозі не існує”***. Якщо ж файл існує, виведеться повідомлення про існування файлу, крім того дані про розмір, час створення, час останньої модифікації і вміст файла. Біля поля вкажіть як підказку назву вашого файлу із роботою last_name_lab5.php

3. У текстовому файлі lab5/files/tag1.txt в першому рядку вписати тег (без дужок <>) у другому - його опис, в третьому - інший тег, у четвертому – його опис і так далі, усього 5 - 6 тегів. Далі в PHP-скрипті організувати зчитування даних файлу по одному рядку і виведення його вмісту у вигляді:

| | |
|---|---|
| **<br>** | розриває рядок |
| **<td>** | відкриває комірку в таблиці |

4. Модифікована задача з п. 3: У текстовому файлі lab5/files/tag2.txt в першому рядку вписати тег (з дужками <>) і його опис, в другому — інший тег і його опис, і так далі, усього 5 - 6 тегів. Далі в PHP-скрипті організувати зчитування даних файлу і виведення його вмісту у вигляді:

| | |
|---|---|
| **<br>** | розриває рядок |
| **<td>** | відкриває комірку в таблиці |

Для поділу вмісту рядка на два елементи масиву можете використати функцію **explode().**

Скрипт також повинен порахувати, скільки всього тегів описано у файлі. Відповідь записати у файл lab5/files/out.txt у вигляді наступного тексту: ***“Всього у файлі tag2.txt описано тегів: 5 ”***.

Якщо даний файл відсутній, створити його для читання і запису.

Зчитати вміст з файлу lab5/files/out.txt і вивести його під таблицею.

5. Реалізувати наступні функції (текст візьміть згідно свого варіанту із наступного завдання і збережіть у файл під назвою last_name_text.txt. Файли з текстами розміщуйте у папці lab5/files, під час виконання функцій текст зчитується з відповідних файлів):

5.1. Функція, що виводить слова заданого тексту (файл last_name_text.txt) у алфавітному порядку (для розбиття тексту на окремі елементи можна скористатися функцією explode).

5.2. Задано текст (файл last_name_text.txt). Вивести список двох слів, які найчастіше зустрічаються у тексті.

5.3. Задано текст (файл last_name_text.txt). Вивести найдовше слово тексту і його довжину, якщо декілька слів мають найбільшу довжину, то вивести всі з них.

5.4. Задано текст (файл last_name_text.txt). Вивести найкоротше слово тексту і його довжину, якщо декілька слів мають найкоротшу довжину, то вивести всі з них.

5.5. Знайти в тексті (файл last_name_text.txt) всі слова, які починаються на першу літеру вашого імені. Якщо таких слів не буде, то додайте до тексту будь-яке речення, яке містить необхідні слова.

5.6. В тексті (файл last_name_text.txt) замінити всі малі літери “о” на великі “О”.

5.7. Створити PHP-сценарій, який випадковим чином виводить абзац тексту з п’яти заданих абзаців (текст візьміть зі свого варіанту + текст із наступних 4-х варіантів).

6. Виконайте завдання згідно Вашого варіанту і двох наступних, використовуючи функції обробки рядкових змінних:

| № | Умова | Текст |
|---|---|---|
| 1 | Визначити кількість слів, які починаються з символу «м». | Синтаксис мови PHP бере початок з C, Java і Perl. PHP досить простий для вивчення. Перевагою PHP є надання web- розробникам можливості швидкого створення динамічних web-сторінок. |
| 2 | Визначити кількість слів, які містять хоча б один символ «є». | Ефективність є виключно важливим чинником при програмуванні для розрахованих на багато користувачів середовищ, до яких належить і web. Дуже важлива перевага PHP полягає в його транслюючому інтерпретаторі. Такий пристрій дозволяє обробляти сценарії з |

| | | |
|---|---|---|
| | | достатньо високою швидкістю. |
| 3 | Визначити кількість слів, які містять символ «і» рівно два рази. | З точки зору типізації, PHP є мовою програмування з динамічною типізацією. Немає необхідності явного визначення типу змінних, хоча така можливість існує. В разі звернення до змінної, ядро PHP трактує її тип відповідно до контексту. За необхідності можливе приведення змінної до певного типу за допомогою відповідних конструкцій мови. |
| 4 | Визначити кількість слів у тексті і кількість літер в першому і останньому словах. | До базових типів належать булеві дані, цілі та дійсні числа із плаваючою крапкою, а також рядки. Булеві дані виражають істинність значення. Цілі числа можуть бути подані у вісімковому, десятковому та шістнадцятковому вигляді. Розмір цілого числа може змінюватись залежно від платформи, як правило, розрядність становить 32 біти. |
| 5 | Знайти та роздрукувати слово в тексті максимальної довжини. | PHP не підтримує беззнакові цілі числа. Дійсні числа із плаваючою крапкою можуть бути подані в десятковій або експоненційній формі. Рядки розділяють на два класи — рядки, що підлягають аналізу, та рядки, що не підлягають аналізу. |
| 6 | Роздрукувати в стовпчик всі різні слова заданого тексту, вказавши для кожного кількість входжень в текст. | PHP надає широкий спектр функцій для пошуку та заміни тексту в рядках. Для цього використовують як традиційний підхід, так і спеціальний підхід, що базується на використанні регулярних виразів. |
| 7 | Надрукувати заданий текст, видаливши з нього повторне входження слів. Повідомити про кількість зроблених видалень. | Константи в PHP - ідентифікатори простих значень. Можливе визначення константи, причому після її оголошення стає неможливою зміна її значення чи анулювання. Константи можуть мати лише скалярні значення. Підтримується можливість отримання значення константи за динамічним ім'ям. |
| 8 | Надрукувати заданий текст, видаливши в ньому | Оператори бувають трьох типів - унарні, бінарні та тернарні. Оператори, як і в інших |

| | | |
|---|---|---|
| | зайві пробіли (тобто з декількох поспіль пробілів залишити тільки один). | мовах характеризуються не лише дією, а й асоціативністю та пріоритетністю. |
| 9 | Здійснити заміну символів ':' на ',' в першому рядку, починаючи з п'ятої позиції. | Функції в сенсі мови є контейнерами коду: причому можливе поміщення інших функцій та класів. На цьому і базується можливість умовного визначення функції. В цьому випадку висувається вимога попередньої декларації викликаної функції, що не обов'язково в інших випадках. |
| 10 | Дано текст, що містить кілька круглих дужок. Якщо дужки розставлені правильно (тобто кожній відкриваючій відповідає одна закриваюча), то вивести число 1. Якщо ні, то вказати, кількість незакритих дужок. | Протокол HTTP, (засобами якого, як правило, обмінюються інформацією клієнт та Web-сервер не надає змогу зберегти стан сеансу взаємодії. Це випливає із того, що між клієнтом та сервером не встановлюється постійне з'єднання (і клієнт не надає жодних відомостей, що можуть виділити його з поміж інших активних в деякому околі часу). |
| 11 | Дано два рядки. Визначити кількість входжень слів з першого рядка у другий. | Мова програмування - система позначень для опису алгоритмів і структур даних. Мова програмування визначає набір лексичних, синтаксичних та семантичних правил, які задають зовнішній вигляд програми і дії, які виконує комп'ютер під її управлінням. |
| 12 | Потроїти кожне входження символу «Р» в кожному слові. | PHP інтерпретується веб-сервером в HTML-код, який передається на сторону клієнта. На відміну від таких скриптових мов програмування, як JavaScript, користувач не має доступу до PHP-коду, що є перевагою з точки зору безпеки, але значно погіршує інтерактивність сторінок |
| 13 | Вивести на екран речення у якому знаходиться слово «програмування». | У більшості візуальних середовищ програмування реалізовано функції автоматичної генерації коду. Традиційно автоматична генерація коду використовується |

| | | |
|---|---|---|
| | | для створення форм, кнопок, перемикачів та інших візуальних елементів. Деякі сучасні середовища можуть автоматично генерувати мало не будь-які фрагменти програм, а то навіть і цілі програми. |
| 14 | У тексті між словами вставити замість «,» символ «;». | Сесія, наприклад в мові програмування PHP, дуже важлива одиниця. Завдяки сесії можна передавати дані від одного файлу до наступного без явних ознак відправлення даних, тобто метод POST та GET не використовується. |
| 15 | Видалити частину тексту, яка взята в дужки. Дужки не видаляти. | PHP - мова, яка може бути вбудованою безпосередньо в html-код сторінок, які, в свою чергу коректно будуть оброблені PHP - інтерпретатором. Механізм PHP просто починає виконувати код після першої екрануючої послідовності (<?) і продовжує виконання до того моменту, коли він зустріне парну екрануючу послідовність (?>). |
| 16 | Підрахувати кількість слів у тексті. Роздрукувати текст, виводячи на екран слово «асемблер» в зворотньому порядку. | Прикладом мови низького рівня є асемблер. Мови низького рівня орієнтовані на конкретний тип процесора і враховують його особливості, тому для перенесення програми на асемблері на іншу апаратну платформу її потрібно майже повністю переписати |
| 17 | Підрахувати кількість речень у тексті і кількість слів «PHP». | PHP 3.0 був офіційно випущений в червні 1998 року після 9 місяців публічного тестування. Оновлення PHP 4 здійснювалося тільки до кінця 2007 року. |
| 18 | Визначити кількість слів, у яких довжина перевищує 7 символів. | PHP застосовувався при розробці таких CMS, як Drupal, Joomla, PHP-Nuke. З його використанням розроблялися системи для веб комерції - osCommerce і Magento, утиліти адміністрування СУБД - phpMyAdmin, галереї зображень - Gallery Project, Coppermine і багато іншого. |
| 19 | Розділити заданий текст | MySQL - це одна з найпопулярніших і |

| | | |
|---|---|---|
| | на декілька масивів, що можуть містити не більше 20 символів. | найпоширеніших СУБД в Інтернеті завдяки вдалому поєднанні користувацьких властивостей, відкритому коду і добрій технічній підтримці. Офіційний сайт - www.mysql.com |
| 20 | Перевірити текст на наявність однакових слів, вивести ці слова. | PHP є мовою програмування з динамічною типізацією. Синтаксис PHP подібний синтаксису мови Сі. Деякі елементи, такі як асоціативні масиви і цикл foreach, запозичені з Perl. PHP є мовою програмування з динамічною типізацією. |
| 21 | Замінити в тексті усі подвійні лапки на одинарні, а крапки - трьома крапками. | Розробники PHP відмовилися від доповнення про персональне використання, яке було в абревіатурі PHP/FI. Мова була названа просто PHP - абревіатура, що містить рекурсивний акронім (англ. PHP: Hypertext Preprocessor - "PHP: Препроцесор Гіпертексту"). |
| 22 | Роздрукувати в стовпчик усі слова тексту, окрім слів «PHP» та «MySQL». | PHP - мова програмування, спеціально розроблена для написання web-додатків (скриптів, сценаріїв), що виконуються на Web-сервері. Мова програмування PHP, особливо в зв'язці з популярною базою даних MySQL - оптимальний варіант для створення інтернет-сайтів різної складності. |
| 23 | Перевірити, чи є в тексті числа, визначити, перед якими словами вони розташовані. | Оновлення PHP 4 випускатимуться тільки до кінця 2007 року. До цього ж часу здійснюватиметься офіційна підтримка четвертої версії. Далі до 8 серпня 2008 року в міру необхідності з'являтимуться тільки критичні оновлення безпеки. |
| 24 | Визначити кількість речень у заданому тексті. Роздрукувати текст, виводячи на екран слово «CSS» в зворотньому порядку. | CSS (Cascading Style Sheets) — каскадні таблиці стилів, які застосовуються для візуального форматування документу в мовах розмітки. CSS використовується для того, щоб визначити кольори, шрифти, та інші аспекти вигляду сторінки. |
| 25 | Визначити, скільки слів у | Функцією називається фрагмент програмного |

| | | |
|---|---|---|
| | тексті починається з літери, з якої починається третє речення. Дану літеру визначити засобами функцій опрацювання рядків. | коду, що володіє унікальним ім'ям і призначений для вирішення конкретної задачі. Функція викликається по імені в різних точках програми, що дозволяє багато разів виконувати фрагмент з вказаним ім'ям. Перевага такого рішення полягає в тому, що блок коду пишеться лише один раз, а потім легко модифікується по мірі необхідності. |
| 26 | Замінити символи «!» на «?» у тих реченнях, які містять символ «:». Обчислити кількість здійснених замін. | Без чого подальше вивчення Web-програмування неможливе! Щоб створити сайт потрібно знати хоча б мову розмітки! Найлегшою і найпопулярнішою є мова HTML. Що таке: HTML! |
| 27 | Визначити довжину кожного речення (рядка) в тексті. | Окрім функцій та операторів PHP існують змінні. Змінна — це іменована ділянка пам'яті, де знаходяться якісь дані. Ім'я змінної починається з знака $, за яким йде буква або знак підкреслення з подальшими в будь-якій кількості буквами, цифрами або символами підкреслення. |
| 28 | Перетворіть у верхній регістр перший символ кожного слова у тексті. | JavaScript — мова програмування, за допомогою якої Ви можете створювати інтерактивні Web-сторінки. Величезною перевагою JavaScript перед іншими мовами програмування є те, що їй не потрібно ніяких сторонніх інтерпретаторів, а достатньо тільки одного браузера. |
| 29 | Змінити регістр кожного слова в тексті, якщо воно не починається з цифри. | П'ята версія PHP була випущена розробниками 13 липня 2004 року. Зміни включають оновлення ядра Zend (Zend Engine 2), що істотно збільшило ефективність інтерпретатора. Введена підтримка мови розмітки XML |
| 30 | Знайти в тексті слова «унарні» та «мови» і перенести їх у кінець тексту. | Оператором називається конструкція мови програмування, що отримує один або більше аргументів, виконує над ними операцію, та повертає результат, який може бути новим аргументом. В залежності від того, скільки |

| | | |
|---|---|---|
| | | значень приймає оператор, виділяють три виду операторів: унарні, бінарні та тернарні. |

7. Здати роботу.

## 5.3 Контрольні запитання

1. Для чого призначена спеціальна конструкція мови PHP include?
2. Для чого призначена спеціальна конструкція мови PHP require?
3. Яка різниця між include і require?
4. Які ви знаєте режими відкриття файлів?
5. Які ви знаєте функції порівняння рядків?
6. Які ви знаєте функції перетворення рядків і файлів до формату HTML і навпаки?

# Лабораторна робота № 6. PHP. Регулярні вирази

Тривалість: 2 акад. години
Мета: навчитися будувати і застосовувати регулярні вирази
Завдання: вивчити правила формування регулярних виразів та виконати всі пункти лабораторної роботи.

## 6.1 Теоретичні відомості

Лекційний матеріал на тему “Регулярні вирази”
Синтаксис регулярних виразів
Функції роботи із регулярними виразами

## 6.2 Хід роботи

1. Задано текст (файл text.txt). Виведіть з нього на веб-сторінку:
   a. весь заданий текст;
   b. тільки назви відкриваючих тегів;
   c. назви відкриваючих тегів разом і кутовими дужками;

*Примітка: теги із кутовими дужками не повинні впливати на форматування веб-сторінки.*

2. У тексті ( файл text.txt), використовуючи регулярні вирази, знайти всі входження слова незалежно від регістру (в тому числі слова, частиною яких є вказане слово):
   a. тег;
   b. HTML;
   c. частини слова, введеної в однорядкове текстове поле.

Вивести тільки ті речення, в яких дане слово присутнє, в порядку спадання числа знайдених відповідностей шаблону пошуку (першим повинно йти речення, в якому шукане слово зустрічається найчастіше), знайдені слова виділити жирним шрифтом.

3. Створіть власний фрагмент html тексту з відкриваючими і закриваючими тегами, який починається і закінчується тегом body. З допомогою функції, замініть відкриваючі і закриваючі теги на пропуски, залишивши тільки змістовну частину. Замініть подвійні пропуски, які утворилися при заміні розмітки, на один пробіл. Виведіть утворений текст.

4. Створити форму, яка міститиме наступні поля заповнення:
ім’я;
прізвище;
логін;
пароль;
повторити пароль
адреса електронної пошти (e-mail).

Використовуючи регулярні вирази і функцію preg_match(), виконати перевірку валідності інформації, введеної в поля форми (ім’я та прізвище повинно містити лише літери: перша літера велика, наступні маленькі, дозволено кирилиця і латина; логін тільки латина малими літерами; пароль: мінімум 6 символів, з яких мінімум по 1 букві і цифрі; повторити пароль: повинен співпадати із попередньо введеним; e-mail на наявність символу @ і відповідної структури. Якщо поле заповнене вірно, то навпроти поставити зображення галочки як позначення коректності введення, інакше колір тексту і рамки стає червоним, а сама рамка подвійною, збоку поля з’являється вимога до заповнення.

5. Перевірте коректність введення почтового індексу з використанням регулярних виразів за варіантом, що надано в табл. 1.

Перед полем для введення напишіть назву країни, вимоги для введення поштового індексу і зразок коректного введення. Якщо поле заповнене вірно, збоку від поля виведіть повідомлення про коректність введення, інакше - повідомлення про некоректність введення і вказівкою для повторного введення

Табл. 1 – Індивідуальні варіанти завдання

| Варіант | Країна | Алгоритм формування індексу |
|---|---|---|
| 1 | Сполучені Штати Америки | Числовий код з дев’ятьма цифрами. Після п’ятої дефіс |
| 2 | Нідерланди | Чотири цифри. Перша цифра не нуль. Після пропуску дві літери |

| | | |
|---|---|---|
| 3 | Італія | П’ять цифр. Попереду “V-“ чи “I-“. |
| 4 | Індія | Шість цифр. Перша цифра не нуль. Пропуск після третьої |
| 5 | Польща | Дві цифри, дефіс, три цифри |
| 6 | Іспанія | П’ять цифр. Після третьої цифри пропуск. Попереду “I-“. |
| 7 | Великобританія | Від 5 до 8 цифр і літер, що розділені після четвертої пропуском. Наприклад, AA9A 9AA |
| 8 | Латвія | Чотири цифри. Попереду “V-“ чи “I-“. |
| 9 | Македонія | Чотири цифри. Перша від 1 до 7 |
| 10 | Португалія | Чотири цифри, дефіс, три цифри, пропуск, 5 літер |

## 6.1 Контрольні запитання

1. Назвіть відомі Вам формати регулярних виразів.
2. Поясніть використання функцій preg_match(), preg_match_all(), preg_replace().
3. Який пошук здійснюватиме вираз:
   - “/^\w{6,9}\d{3}/s”;
   - “/ \d{3}-\d{2}-\d{2}/m”?
4. Що вказують в параметрі модифікатор?

# Лабораторна робота №7. Об'єктно-орієнтоване програмування на PHP

Тривалість: 2 акад. години
Мета: ознайомитися із основами об'єктно-орієнтоване програмування на PHP.
Завдання: вивчити та навчитися застосовувати основні поняття об'єктно-орієнтованого програмування.

## 7.1 Теоретичні відомості

Стало стандартом, що вивчення об'єктно-орієнтованого програмування (ООП) завжди починається з поняття «класу». Ідея об'єктно-орієнтованого програмування полягає в тому, що в ООП ми починаємо думати сутностями. А що таке сутність? Це все, що нас оточує, тобто це звичайні об'єкти. А об'єктом у нас може бути все що завгодно (автомобіль, будинок, комп'ютер і т.д.). Що таке клас? - Клас описує ці об'єкти, тобто є їх формальною моделлю.

Наприклад, у нас є об'єкт «автомобіль». Перш ніж його зібрати, нам потрібне креслення. У цьому кресленні має прописуватися те, що ми повинні знати про цей автомобіль - це те, що автомобіль має деякі властивості (колір, розмір, максимальна швидкість, рівень безпеки) і методи (автомобіль може рухатися вперед і назад, повертати кермо, стежити за втомою водія і т.д.).

В об'єктно-орієнтованому програмуванні для того, щоб описати таке креслення існує поняття «клас». «Клас» обов'язково повинен включати в себе опис таких понять, як властивості і методи даного об'єкту.

Отже, як же записується «клас»? Записується він у такий спосіб - спочатку йде ключове слово class, потім ім'я класу і фігурні дужки, в яких описуються властивості і методи:

```
<?php
class Car{
// Опис властивостей
// Опис методів
}
?>
```

Зверніть увагу, що імена класів чутливі до регістру.

Ґрунтуючись на цьому класі можна створювати об'єкти, в ООП вони ще називаються «екземплярами класу». Об'єкт класу створюється наступним

чином - ставиться змінна, далі йде ключове слово new і ім'я класу з круглими дужками:

```
<?php
// Опис об'єкта
class Car {
// Опис властивостей
// Опис методів
}
// Створення об'єкта
$car = new Car();
?>
```

Для одного класу (креслення) можна створювати кілька екземплярів класу або об'єктів (атомобілів):

```
<?php
// Опис об'єкта
class Car {
// Опис властивостей
// Опис методів
}
// Створення об'єкта
$car1 = new Car();
$car2 = new Car();
?>
```

Тепер давайте розберемо, що ж таке властивості і методи об'єкта. Почнемо з властивостей. Це звичайні змінні. Але не зовсім. Якщо змінна знаходиться десь в коді, то це звичайна змінна. А якщо змінна знаходиться в класі, то ця змінна - властивість.

Наприклад, у автомобіля є рік випуску. Відповідно ініціалізуємо змінну $year. Ми можемо присвоїти їй значення по замовчуванню, наприклад 2018. Також у автомобіля є максимальна швидкість, ініціалізуємо змінну $speed. Ще у автомобіля є марка, ініціалізуємо змінну $model. Всі ці змінні - це властивості об'єкта, тобто нашого автомобіля.

Однак в об'єктно-орієнтованому програмуванні перед такими змінними-

властивостями прийнято ставити модифікатори доступу. Що таке модифікатори доступу? Це ключові слова, які позначають можливість доступу або можливість звернутися з методів об'єкта або з примірників класу до властивостей цього об'єкта. Ці ключові слова записуються як public, protected, private.

Доступ до властивостей і методів класу, оголошених як public (загальнодоступний), дозволений звідусіль. Модифікатор protected (захищений) надає доступ наслідуваним і батьківським класам. Модифікатор private (закритий) обмежує область видимості так, що тільки клас, де оголошений сам елемент, має до нього доступ.

Поки що будемо використовувати модифікатор public:

```
<?php
// Опис об'єкта
class Car {
public $year = 2018;
public $speed;
public $model;
}
// Створення об'єкта
$car1 = new Car();
$car2 = new Car();
?>
```

Тепер розберемо, що у нас тут відбувається. Ми з класу створили два об'єкти або два екземпляри класу, які мають однакові властивості, два з яких не визначені ($speed, $model) і одну властивість, у якої задане значення по замовчуванню ($year = 2018).

Щоб звернутися до однієї з властивостей створеного об'єкта, необхідно написати ім'я змінної, якій присвоєно об'єкт, далі поставити ось таку стрілку -> (дефіс і знак більше) і ім'я властивості, але без знака долара:

```
<?php
// Опис об'єкта
class Car {
public $year = 2018;
public $speed;
public $model;
}
// Створення об'єкта
$car1 = new Car();
```

```
    $car2 = new Car();
    echo $car1-> year; // звернення до властивості
об'єкта
    ?>
```

Далі присвоюємо першому і другому автомобілю швидкість і модель. Так як у нас немає значення швидкості за замовчуванням і немає значення моделі, то можна зробити наступним чином:

```
    $car1-> speed = 210,
    $car1-> model = «bmw»,
    $car2-> speed = 260,
    $car2 -> model = «lexus»:
```

```
    <?php
    // Опис об'єкта
    class Car {
    public $year = 2018;
    public $speed;
    public $model;
    }
    // Створення об'єкта
    $car1 = new Car();
    $car1-> speed = 210; // присвоєння значення
властивості об'єкта
    $car1-> model = "bmw"; // присвоєння значення
властивості об'єкта
    $car2 = new Car();
    $car2-> speed = 260; // присвоєння значення
властивості об'єкта
    $car2-> model = "lexus"; // присвоєння значення
властивості об'єкта
    echo $car2-> year; // звернення до властивості
об'єкта
    ?>
```

## Методи

Як було зазначено вище, всередині класу є дві речі: перша - це властивості, і друга - це методи. Тепер детальніше зупинимося на них. Що таке методи об'єкта? Якщо говорити зовсім простою мовою, то це поведінка об'єкта.

Згадайте про об'єкт «Автомобіль». Автомобіль може рухатися вперед і назад, а також вправо або вліво, вона може розвивати певну швидкість. Все це є методами, тобто те, що може робити даний об'єкт.

Метод - це та ж сама функція! І описується він точно так само, як звичайна функція.

Тільки функція, яка лежить десь в коді, це звичайна функція, а функція, описана в класі (або по-іншому сказати, що знаходиться всередині класу) - це метод. Запам'ятайте це!

У всіх методів, так само, як і у властивостей є точно такі ж модифікатори доступу (public, protected, private). Якщо не вказати жоден з модифікаторів, то за замовчуванням буде вважатися модифікатор public. Однак, краще завжди вказувати один з них.

Давайте на прикладі розгляне, як створюється метод. Продовжимо розгляд на прикладі класу, де ми описували властивості об'єкта «автомобіль». Усередині цього класу ми опишемо найпростіший метод і називатися він у нас буде function takeSpeed().

Отримати доступ до цього методу можна точно так само, як і до властивості, наприклад: `$car1-> takeSpeed();`

```
<?php
// Опис об'єкта
class Car {
// властивості
public $year = 2018;
public $speed;
public $model;
// метод
public function takeSpeed() {
// Тут метод щось робить
echo "Швидкість автомобіля =";
}
}
```

```
// Створення об'єкта
$car1 = new Car();
// Отримуємо доступ до методу
$car1-> takeSpeed();
$car2 = new Car();
?>
```

Якщо запустити цей код, то у нас викликається наш метод і буде виведено: «Швидкість автомобіля =».

Ось тут починається найцікавіше. Ви звернули увагу, швидкість якого автомобіля ми хочемо отримати? Логічно припустити, що раз ми звертаємося з екземпляра класу car1, то і швидкість ми хочемо отримати автомобіля bmw. Однак, в методі, який у нас відповідає за отримання швидкості автомобіля, не вказано якого саме об'єкту швидкість нам потрібна. У цьому випадку нам необхідно звернутися до властивості speed класу Car. Але вона у нас одна, а автомобіля (об'єкта) два.

Так ось, нам потрібно вказати властивість (швидкість) якого об'єкта ми маємо на увазі.

Таким покажчиком в ООП є ключове слово $this.

І записується це так: `$this-> speed.`

```
<?php
// Опис об'єкта
class Car {
// властивості
public $year = 2018;
public $speed;
public $model;
// метод
public function takeSpeed() {
// Вказуємо метод, чию швидкість ми хочемо отримати
echo "Швидкість автомобіля =". $this-> speed;
}
}
// Створення об'єкта
$car1 = new Car();
```

```
    $car1-> speed = 210; // присвоєння значення
властивості об'єкта
    // Отримуємо доступ до методу
    $car1-> model = "bmw"; // присвоєння значення
властивості об'єкта
    $car1-> takeSpeed();
    $car2 = new Car();
    $car2-> speed = 260; // присвоєння значення
властивості об'єкта
    // Отримуємо доступ до методу
    $car2-> model = "lexus"; // присвоєння значення
властивості об'єкта
    ?>
```

Тепер у нас вже виведеться: Швидкість автомобіля = 210. Тобто ми отримали швидкість об'єкта $car1 (bmw).

Тут важливо запам'ятати - щоб звернутися з методу до властивості об'єкта, ми звертаємося через ключове слово $this.

Давайте розглянемо ще один важливий момент - припустимо у нас в класі Car крім методу takeSpeed(), є метод paintSpeed(). Ми зробимо так, що метод paintSpeed() буде виводити швидкість автомобіля, а метод takeSpeed() буде викликати метод paintSpeed(). Це досить поширена ситуація, що показує - для того щоб звернутися з методу до іншого методу, також використовується ключове слово $this. Поглянемо на код:

```
    <?php
    // Опис об'єкта
    class Car {
    // властивості
    public $year = 2018;
    public $speed;
    public $model;
    // метод
    public function takeSpeed() {
    // Звертаємося з методу takeSpeed() до методу
paintSpeed()
    $this-> paintSpeed();
```

```
    }
    // метод
    function paintSpeed() {
    // Вказуємо методу, чию швидкість ми хочемо отримати.
Йде звернення з методу до властивості
    echo "Швидкість автомобіля =". $this-> speed;
    }
    }
    // Створення об'єкта
    $car1 = new Car();
    $car1-> speed = 210; // присвоєння значення
властивості об'єкта
    // Отримуємо доступ до методу
    $car1-> model = "bmw"; // присвоєння значення
властивості об'єкта
    $car1-> takeSpeed();
    $car2 = new Car();
    $car2-> speed = 260; // присвоєння значення
властивості об'єкта
    // Отримуємо доступ до методу
    $car2-> model = "lexus"; // присвоєння значення
властивості об'єкта
    $car2-> takeSpeed();
    ?>
```

Отже, звернення з методу до методу і звернення з методу до властивості здійснюється через ключове слово $this.

Дивіться, що тут відбувається. Ще нічого серйозного не зробили, а вже якось багато всього понаписувано. Щось тут не так і взагалі це йде в розріз з ідеологією ООП. Чи не можна це скоротити? Можна, можливо. І надалі теж будемо скорочувати кількість коду. У зв'язку з чим, ми може це зробити?

На даному етапі вивчення об'єктно-орієнтованого програмування з'являється таке поняття, як «конструктор класу». Що ж це таке?

Конструктор класу - це спеціальний метод, який автоматично викликається в момент створення об'єкту. У PHP конструктор класу має спеціальну назву - подвійне підкреслення construct (function __construct).

Давайте з вами перевіримо, як викликається конструктор для кожного

створеного об'єкта. Запустіть цей код і Ви побачите, що для кожного об'єкта викликається конструктор:

```
<?php
class Users {
public $name;
public $login;
public $password;
// Створення конструктора
function __construct () {
echo "<p> Конструктор викликався автоматично!";
}
// Створюємо метод getInfo ()
function getInfo () {
echo "<p> Name:". $this-> name. "<br>";
echo "Login:". $this-> login. "<br>";
echo "Password:". $this-> password. "<br>";
}
}
$user1 = new Users ();
$user1-> name = "Vasya";
$user1-> login = "vas";
$user1-> password = 123;
// Виводимо метод getInfo ()
$user1-> getInfo ();
$user2 = new Users ();
$user2-> name = "Petya";
$user2-> login = "pet";
$user2-> password = 321;
// Виводимо метод getInfo ()
$user2-> getInfo ();
$user3 = new Users ();
$user3-> name = "Vova";
$user3-> login = "vov";
$user3-> password = 456;
// Виводимо метод getInfo ()
$user3-> getInfo ();
?>
```

Коли ми створювали об'єкт: $user1 = new Users (); ми ставили круглі дужки біля об'єкту Users(). Логіка нам підказує, що раз є круглі дужки, то вони для чогось потрібні. Так ось, ці круглі дужки біля об'єкту - це дужки самого конструктора. Тобто, якщо ми передамо в круглих дужках об'єкта якийсь параметр, то він потрапить в конструктор. Давайте краще подивимося, як це виглядає в коді. Ми в конструкторі вкажемо параметр у вигляді змінної і будемо її виводити, а коли створюємо об'єкт, в ньому вкажемо числа (1,2,3). Запустіть цей код і подивіться, що вийде:

```
<?php
class Users {
public $name;
public $login;
public $password;
// Створюємо конструктор з параметром
function __construct ($number) {
echo "<p> Конструктор викликався автоматично $number!";
}
// Створюємо методод getInfo ()
function getInfo () {
echo "<p> Name:". $this-> name. "<br>";
echo "Login:". $this-> login. "<br>";
echo "Password:". $this-> password. "<br>";
}
}
$user1 = new Users (1);
$user1-> name = "Vasya";
$user1-> login = "vas";
$user1-> password = 123;
// Виводимо метод getInfo ()
$user1-> getInfo ();
$user2 = new Users (2);
$user2-> name = "Petya";
$user2-> login = "pet";
$user2-> password = 321;
```

```
// Виводимо метод getInfo ()
$user2-> getInfo ();
$user3 = new Users (3);
$user3-> name = "Vova";
$user3-> login = "vov";
$user3-> password = 456;
// Виводимо метод getInfo ()
$user3-> getInfo ();
?>
```

Запам'ятайте - все, що ми передаємо в дужки об'єкта, потрапляє в конструктор.

Поняття “конструктор” є у всіх мовах програмування, але в PHP він позначається саме, як нижня __construct. Це метод автоматично викликається. Для чого використовується конструктор? Для ініціалізації чого-небудь. Наприклад, коли створюється новий об'єкт, ми хочемо щоб, щось відбувалося (щось в сесію записувалося, в файл що-небудь записувалося). Тобто, щось відбувалося автоматично. Це дуже схоже на автозагрузку операційної системи. Завантажилася операційна система і разом з нею сталися якісь дії, завантажилися програми і т.д. Тут те ж саме.

Однак, поряд з конструктором у нас є ще один метод, який використовується, коли закінчується код, тобто, коли код відпрацював і об'єкти видаляються. Звичайно ми можемо видалити об'єкт і раніше, ніж закінчиться код. Наприклад, функцією unset (). Але нам потрібно, щоб у разі вилучення об'єктів, неважливо примусово ми видаляємо їх чи вони видаляються після завершення коду, щось відбувалося, як і при створенні об'єктів.

Для цього у нас є метод, який називається деструктор. Цей метод автоматично викликається при видаленні об'єкта. У PHP він позначається, як подвійне нижнє підкреслення `destruct (__destruct):`

```
<?php
class Users {
public $name;
public $login;
public $password;
```

```
    // Створення конструктора
    function __construct ($number) {
    echo "<p> Конструктор зголосився автоматично
$number!";
    }
    // Створюємо деструктор
    function __destruct () {
    echo "<p> Об'єкт видалився!";
    }
    // Створюємо методод getInfo ()
    function getInfo () {
    echo "<p> Name:". $this-> name. "<br>";
    echo "Login:". $this-> login. "<br>";
    echo "Password:". $this-> password. "<br>";
    }
    }
    $user1 = new Users (1);
    $user1-> name = "Vasya";
    $user1-> login = "vas";
    $user1-> password = 123;
    // Виводимо метод getInfo ()
    $user1-> getInfo ();
    $user2 = new Users (2);
    $user2-> name = "Petya";
    $user2-> login = "pet";
    $user2-> password = 321;
    // Виводимо метод getInfo ()
    $user2-> getInfo ();
    $user3 = new Users (3);
    $user3-> name = "Vova";
    $user3-> login = "vov";
    $user3-> password = 456;
    // Виводимо метод getInfo ()
    $user3-> getInfo ();
    ?>
```

Запустивши цей код, ми бачимо, що при видаленні об'єктів, а вони видаляються у нас тут автоматично, викликається деструктор із записом -

Об'єкт пішов. У нас було три об'єкти - відповідно виводяться три рази запис - «Об'єкт видалився!».

Зверніть увагу, що деструктор має круглі дужки, але передавати параметри в ці круглі дужки не можна! Ще один важливий момент - порядок видалення об'єктів ніколи не визначений. Тобто у нас є три об'єкти і їх треба видалити. Так ось, в якому порядку вони будуть видалятися невідомо. Тому є таке правило - з деструктора не звертатися до інших об'єктів, так як на момент виклику об'єкта, він може бути вже видалений.

**Копіювання і передача об'єктів за посиланням**

Тепер давайте поговоримо ось про що - копіювання і передача об'єктів за посиланням, яке так чи інакше завжди присутній в ООП. У чому ідея всього цього? Дивіться: $object2 = $object1. Як Ви думаєте, що це означає? Тут все залежить від того, якої версії PHP використовують — 4 чи >=5. Бо в 4 версії PHP це означало копіювання. Тобто об'єкт 2 - це копія об'єкта 1. А якщо ми хотіли передати об'єкт за посиланням, то ми писали ось таку річ — $object2 = & $object1. Що ж це означає - передати об'єкт за посиланням? Я думаю, що всі знають, що таке ярлик на робочому столі в Windows - це посилання на файл. Так ось тут те ж саме. Якщо більш зрозуміліше, то це два імені одного і того ж об'єкта. Все це стосується тільки 4 версії PHP.

А що ж у нас 5+ версії? А тут у нас ось цей вираз `$object2 = $object1` вже буде посиланням. Тоді виникає питання, а як же мені тоді скопіювати об'єкт? А копіюється він за допомогою чарівного слова clone: `$object2 = clone $object1.`

Зверніть увагу, що при копіюванні об'єкта, конструктор не викликається. Але ж при копіюванні створюється новий об'єкт і у нас може виникнути необхідність, щось автоматично виконати.

У цьому випадку ми можемо описати спеціальний метод - подвійне нижнє підкреслення `clone (__clone)`, який буде автоматично викликатися при копіюванні об'єктів:

```
<?php
class Users {
public $name;
public $login;
public $password;
```

```
// Створення конструктора
function __construct ($name, $login, $password) {
$this-> name = $name;
$this-> login = $login;
$this-> password = $password;
}
// Створюємо метод clone
function __clone () {
echo "<p> Об'єкт скопійований!";
}
// Створюємо метод getInfo ()
function getInfo () {
echo "<p> Name:". $this-> name. "<br>";
echo "Login:". $this-> login. "<br>";
echo "Password:". $this-> password. "<br>";
}
}
$user1 = new Users ( "Vasya", "vas", "123");
// Виводимо метод getInfo ()
$user1-> getInfo ();
$user2 = new Users ( "Petya", "pet", "321");
// Виводимо метод getInfo ()
$user2-> getInfo ();
$user3 = new Users ( "Vova", "vov", "456");
// Виводимо метод getInfo ()
$user3-> getInfo ();
// Об'єкт $user4 копія об'єкта $user3
$user4 = clone $user3;
?>
```

## 7.1 Хід роботи

Завдання 1:

1. створити клас Student з властивостями name, suname, group;
2. створити три об'єкти класу Student;
3. задати довільні значення властивостям для кожного з об'єктів.

Завдання 2:

1. в класі Student потрібно описати метод getInfo ();
2. метод getInfo () повинен виводити значення властивостей об'єкта;
3. викликати метод getInfo () для кожного об'єкта.

Завдання 3:

1. в класі Student необхідно описати конструктор;
2. конструктор повинен задавати початкові значення властивостей name, suname, group;
3. створити крім існуючих трьох об’єктів класу Student, додатково ще три об'єкти з використання конструктора.

Завдання 4:

1. в класі Student описати метод __clone ();
2. метод __clone повинен задавати початкові значення властивостей за замовчуванням при копіюванні об'єктів;
3. створити сьомий об'єкт, скопіювавши один з наявних об'єктів.

Завдання 5:

Створіть клас для виведення таблиці множення для вказаного числа (передавати в конструкторі). Створити окремий метод для обчислення. Далі створити кілька об'єктів даного класу для демонстрації працездатності класу. Вивід оформити у вигляді таблиці.

Завдання 6:

Створіть клас країни в якому будуть поля: назва країни, населення і назва столиці (дані вказуйте тих країн, перша літера англійської назви яких співпадає з першою літерою Вашого прізвища, якщо такі країни відсутні, тоді імені, потім по-батькові). Створіть масив об'єктів, виведіть кожний з них у таблицю в три рядки по дві комірки в кожному. У лівій комірці ім'я елемента, в правій - його значення.

Завдання 7:

Створіть клас користувача, з полями: прізвище, ім'я, вік і e-mail.

У HTML формі користувач вводить в чотири різні поля: прізвище, ім'я, вік і e-mail. Після натискання клавіші кнопки ГОТОВО створюється об'єкт

користувача, з методом, який вносить ці дані в поля об'єкту і далі виводить їх використовуючи другий метод класу користувача.

У формі передбачити перевірку, що всі поля перед відправленням не порожні.

## 7.3. Контрольні запитання

1. Що таке клас в ООП і як він описується?
2. Що таке об'єкт в ООП і як він описується?
3. Що таке метод в ООП і як він описується?
4. Що таке конструктор в ООП і як він описується?
5. Що таке деструктор в ООП і як він описується?

# Лабораторна робота №8. PHP. Робота із СУБД

Тривалість: 2 акад. години.

Мета: набути практичних навичок для роботи з PHP і СУБД.

Завдання: дослідити взаємне використання бази даних і PHP

## 8.1. Теоретичні відомості.

Уроки SQL

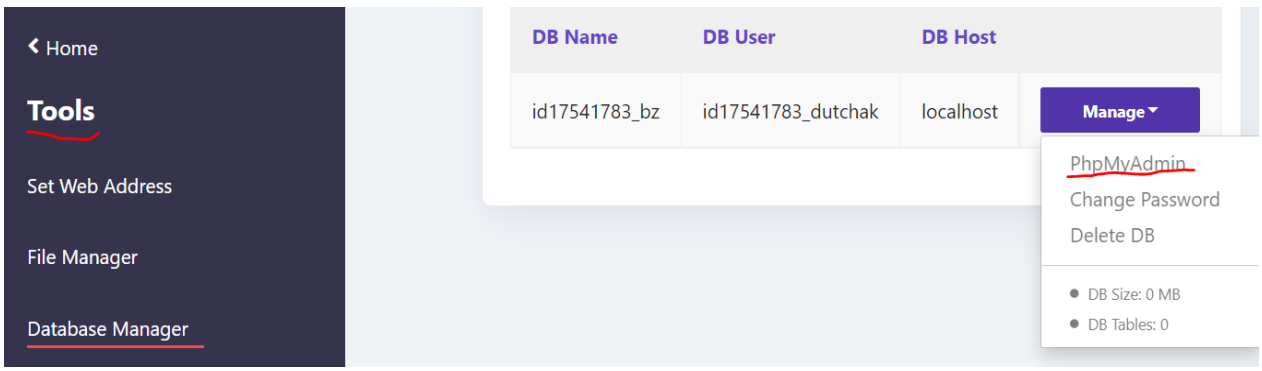
Взаємодія PHP-додатків з базами даних MySQL

## 8.2 Хід роботи

Увага! Для наступних завдань створіть посилання, при кліку на які будуть виконуватися наступні завдання.

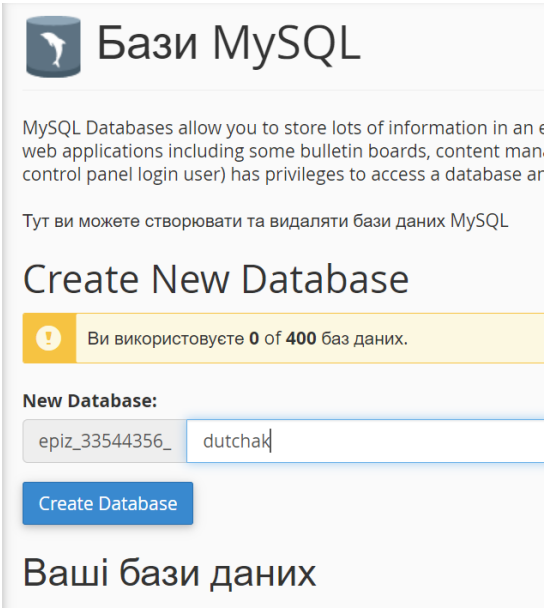
1. Скопіювати на свій вебхостинг файли з папки PHP/labBD (якщо ви цього не зробили під час виконання Лабораторної роботи 1), ввести свої дані для з'єднання із базою даних, розібрати і забезпечити коректне відображення запропонованих викладачем прикладів взаємодії PHP і СКБД MySQL, результат їх виконання можете подивитися тут). Назва Вашої бази даних, ім'я користувача і пароль знайдіть на Вашому веб хостингу). Перейменувати папку labBD на lab8_BD.
2. Перейти в phpMyAdmin. Дослідити меню додатка і створену таблицю.

**https://www.000webhost.com/**:

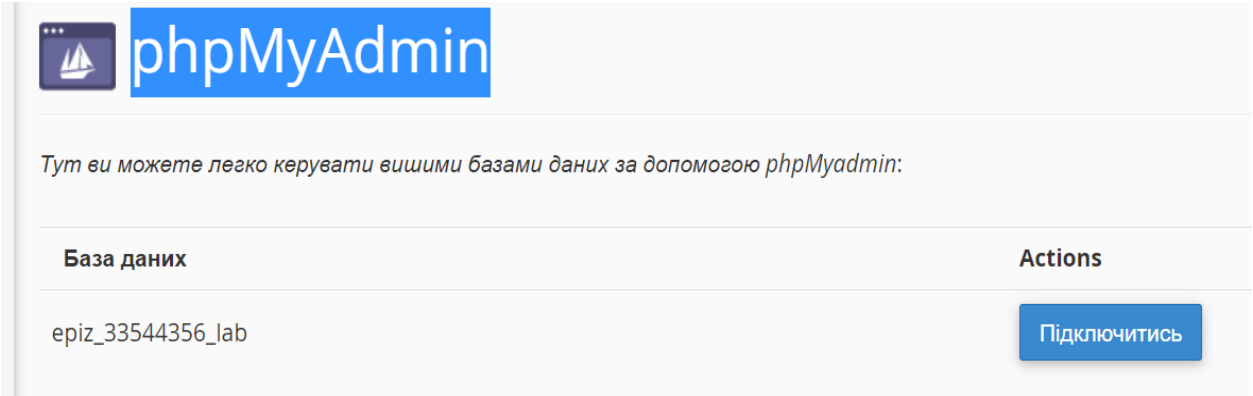



**https://infinityfree.net/**:

Для створення бази даних, перейдіть у Control panel -> Бази MYSQL, далі:




Для створення переходу в phpMyAdmin перейдіть у Control panel -> phpMyAdmin, далі оберіть потрібну базу даних і клікніть “Підключитись”:




3. Перейдіть у закладку SQL, додайте і видаліть кілька записів у таблицю з дисциплінами за допомогою мови запитів SQL.

4. У базі даних створити таблицю користувачів (обов'язковими полями є: логін, пароль - інші поля на свій розсуд). Розробіть PHP-сценарій обробки форми реєстрації і запису даних про користувача в таблицю користувачів. Занесіть мінімум трьох користувачів (себе і двох одногрупників, наступних по списку, в якості логіну вводьте прізвище на латині). Напишіть php-код, який виведе список всіх внесених у базу даних користувачів.

5. Створіть таблицю Склад, і внесіть в неї дані мінімум про три найменування товару згідно варіанту: найменування, назва файлу із зображенням товару, саме зображення, ціна за одиницю товару (шт., кг. і т.д.), наявна кількість. Виведіть на сторінку всю інформацію про доступні товари із

таблиці Склад. При кліку на зображення чи назву товару повинна відкритися сторінка, на яку виведіть всю інформацію про обраний товар і лічильник вибору кількості товару. Додайте кнопку «Купити», після кліку на яку зменшити наявну кількість товару на відповідну кількість купленого товару і вивести змінені дані з таблиці склад бази даних.

6. На сторінку покупки товару, поруч із кнопкою «Купити», додати кнопку «Поповнити склад», яка навпаки буде збільшувати наявну кількість товару на вказану кількість.

1. Птахи
2. Їжа
3. Домашні тварини
4. Гриби
5. Транспорт
6. Дерева
7. Фрукти
8. Овочі
9. Ягоди
10. Косметика
11. Кондитерські вироби
12. Двоколісний транспорт
13. Меблі
14. Комп’ютерна техніка
15. Жіночий одяг
16. Чоловічий одяг
17. Дитячий одяг
18. Взуття
19. Солодощі
20. Спортивні аксесуари
21. Музичні інструменти
22. Транспорт
23. Кухонна техніка
24. Побутова техніка
25. Автомобілі
26. Комп’ютерна техніка
27. Птахи
28. Їжа

29. Домашні тварини
30. Транспорт

## 8.1 Контрольні запитання

1. Що таке База Даних?
2. Що таке SQL?
3. Синтаксис команди вибору окремих полів.
4. Назвіть призначення операторів: ORDER BY, WHERE, LIKE
5. Що таке СКБД MySQL?
6. Назвіть призначення і параметри функції mysqli.

## Лабораторна робота №9. PHP. Робота з базою даних. Створення сайту із новинами

Тривалість: 2 акад. години.
Мета: набути практичних навичок для роботи з PHP і базою даних.
Завдання: дослідити взаємне використання бази даних і PHP та виконати завдання відповідно до ходу роботи.

Увага! Для наступних завдань створіть посилання, при кліку на які будуть виконуватися наступні завдання.

### 9.1 Хід роботи

1. Створити таблицю last_name_news, із полями що містять порядковий номер, тематику, заголовки, контент, дата. Задати автоматичне додавання порядкового номеру (create sequence), поле із заголовками зробити унікальним.

Приклад:

```
mysqli_query($db_server,"drop table duscup_id");
mysqli_query($db_server,"drop sequence duscup_seq");
mysqli_query($db_server,"create sequence duscup_seq minvalue 1000 maxvalue 1999 start 1000");
mysqli_query($db_server,"select setval('duscup_seq',1000)");
mysqli_query($db_server,"create table duscup_id (id integer default nextval('duscup_seq') primary key, name_d varchar(100) unique, key_concepts text[])");
```

2. Занести у дану таблицю новини, розміщені у файлі file/news.txt. Зверніть увагу, що поля розділені «~», а записи «&». Внесення даних перевірте через phpMyAdmin.

Відкриття файлу для читання: `fopen($file,"r");`

закриття файлу `fclose($file;`

```
$file="file/news.txt";
$fdata_my = fopen($file,"r") or die ("Can't open file $file!");
$mas = fread($fdata_my,filesize($file));
```

```
    $mas=explode("&",$mas);
    for ($i=0;$i<count($mas);$i++){
    echo "mas[$i]=$mas[$i]<p>";
    $mas_vidm=explode("~",$mas[$i]);
    for ($j=0;$j<count($mas_vidm);$j++){
    echo "<p>mas_vidm[$j]=$mas_vidm[$j]<p>";
    }
    $res=mysqli_query($db_server,"insert into
last_name_news(tema,zagol,content,chas)
values('$mas_vidm[0]', '$mas_vidm[1]', '$mas_vidm[2]',
'$mas_vidm[3]')");
```

3. Додати в таблицю last_name_news по кожній тематиці мінімум по одній новині за поточний рік. Вивести на сторінку всі внесені дані із таблиці last_name_news у вигляді таблиці.

4. По зразку сайту ukr.net, вивести спочатку три свіжих новини із заголовком Головне, а потім посортовані по тематиці із заголовком, що відповідає тематиці. При кліку по заголовку тематики, відкриваються всі новини відповідної тематики, а при кліку по заголовку новини відкривається сторінка із відповідною новиною, що містить: заголовок, дату, контент. Створити файл file/out.txt і записати в нього загальну кількість новин;

5. Написати запит, який би видалив з таблиці last_name_news новину, яка має порядковий номер 5. Вивести на сторінку всі записи із таблиці last_name_news у вигляді таблиці, окрім даних поля з контентом.

6. Створити форму для додавання новини у таблицю last_name_news.

## 9.1 Контрольні запитання

1. Що таке База Даних?
2. Що таке SQL?
3. Синтаксис команди вибору окремих полів.
4. Назвіть призначення операторів: ORDER BY, WHERE, LIKE.
5. Що таке СКБД MySQL?
6. Назвіть призначення і параметри функції mysqli
7. Назвіть призначення функцій fopen, fclose, fread

# РЕКОМЕНДОВАНА ЛІТЕРАТУРА


1. Гаевский А.Ю. Самоучитель по созданию Web-сторінок HTML JavaScript Dynamic HTML.К.: А.С.К., 2002.
2. Джейсон Ленгсторф. PHP и jQuery для профессионалов — М.: «Вильямс», 2010. —352. С
3. Дмитрий Котеров, Алексей Костарев. PHP. — СПб.: «БХВ-Петербург», 2005. — С. 1120. — (В подлиннике). — ISBN 5-94157-245-X.
4. Квентин Зервас. Web 2.0: создание приложений на PHP / Practical Web 2.0 Applications with PHP. — М.: «Вильямс», 2009. — 544 С.
5. Костарев А. Ф. PHP 5. — СПб.: «БХВ-Петербург», 2008. — С. 1104. — ISBN 978-5-9775-0315-0.
6. Кузнецов М, Симдянов И. PHP 5/6. — СПб.: «БХВ-Петербург», 2009. — С. 1024.
7. Кузнецов М, Симдянов И. PHP. Практика создания Web-сайтов. — 2-е изд. перераб. и доп. — СПб.: «БХВ-Петербург», 2008. — С. 1264. — ISBN 978-5-9775-0203-0.
8. Кузнецов М, Симдянов И. Самоучитель PHP 5/6. — 3-е изд., перераб. и доп. — СПб.: «БХВ-Петербург», 2009. — С. 672. — ISBN 978-5-9775-0409-6.
9. Кузнецов Максим, Симдянов Игорь. PHP на примерах. — 2-е изд. перераб. и доп. — СПб.: «БХВ-Петербург», 2011. — С. 400. — ISBN 978-5-9775-0445-4
10. Матвієнко О.В. Internet-технології: проектування Web-сторінки: Навч.посібник. К. : ЦУЛ, 2004.
11. С. Суэринг, Т. Конверс, Д. Парк. PHP и MySQL. Библия программиста / PHP 6 and MySQL 6 Bible. — 2-е издание. — М.: «Диалектика», 2010. — С. 912.
12. Юринець В.Є. Комп’ютерні мережі. Інтернет: навч. посіб. Львів: ВЦ ЛНУ, 2006.

# ІНФОРМАЦІЙНІ РЕСУРСИ

1. Інтернет. Матеріал з Вікіпедії — вільної енциклопедії. [Електронний ресурс]. – Режим доступу: http://uk.wikipedia.org/wiki/Інтернет

2. PHP. Матеріал з Вікіпедії — вільної енциклопедії [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/PHP

3. PhpStorm. Матеріал з Вікіпедії — вільної енциклопедії [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/PhpStorm

4. PHP Manual. – [Електронний ресурс]. Режим доступу: https://www.php.net/manual/en/index.php

5. Уроки SQL – [Електронний ресурс]. Режим доступу: http://moonexcel.com.ua/уроки-sql_ua