МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
“НАЦІОНАЛЬНИЙ ГІРНИЧИЙ УНІВЕРСИТЕТ”

Л.І. Цвіркун
Р.В. Липовий

ГЛОБАЛЬНІ КОМП’ЮТЕРНІ МЕРЕЖІ

ПРОГРАМУВАННЯ МОВОЮ PHP

Під загальною редакцією професора Л.І. Цвіркуна

Рекомендовано Міністерством освіти і науки України
як навчальний посібник для студентів вищих навчальних закладів

Дніпропетровськ
НГУ
2013

УДК 004.432:004.738(075.8)
ББК 32.973.26-018.1я73
Ц28

Рекомендовано Міністерством освіти і науки України як навчальний посібник для студентів вищих навчальних закладів (лист за № 1/11-8155 від 14 травня 2013 р.)

Рецензенти:

І.В. Жуковицький, доктор технічних наук, професор, завідувач кафедри електронних обчислювальних машин Дніпропетровського національного університету залізничного транспорту імені академіка В. Лазаряна;

М.І. Горбійчук, доктор технічних наук, професор, завідувач кафедри комп'ютерних систем та мереж Івано-Франківського національного технічного університету нафти та газу.

Цвіркун Л.І.

Ц28 Глобальні комп'ютерні мережі. Програмування мовою PHP: навч. посібник / Л.І. Цвіркун, Р.В. Липовий; під заг. ред. Л.І. Цвіркуна. – Д.: Національний гірничий університет, 2013. – 239 с.

ISBN 978–966–350–417–9

Наведено елементи мови HTML, основні поняття глобальних комп'ютерних систем, мови програмування PHP, а також методи розробки програм. Подано лабораторний практикум з основ програмування мовою PHP.

Для студентів вищих навчальних закладів, які навчаються за напрямами підготовки "Системна інженерія" та "Комп'ютерна інженерія".

УДК 004.432:004.738(075.8)
ББК 32.973.26-018.1я73

ISBN 978–966–350–417–9

ЗМІСТ

Присвячується 100-річчю
з дня народження професора
Іванова Анатолія Олександровича –
першого завідувача
кафедри автоматизації та
комп'ютерних систем
Державного ВНЗ
"Національний гірничий
університет"

ПЕРЕДМОВА

Навчальна дисципліна „Глобальні комп'ютерні мережі" належить до професійно-практичного циклу вибіркової частини плана і займає важливе місце у підготовці спеціалістів напрямів підготовки "Системна інженерія" та "Комп'ютерна інженерія".

Посібник відповідає навчальним планам нормативної дисципліни „Глобальні комп'ютерні мережі", яка понад п'ятнадцять років викладається авторами у Державному ВНЗ "Національний гірничий університет" (Державний ВНЗ "НГУ", м. Дніпропетровськ, Україна).

Навчальний посібник може бути використаний для самостійного навчання. Для цього в одній книжці разом з основними поняттями глобальних комп'ютерних мереж, мовою HTML, теорією програмування мовою PHP подано лабораторний практикум, а також наведено багато прикладів з реальними доробками програм.

Як мова програмування у навчальному посібнику застосовується PHP 5.0.

Посібник, написаний викладачами Державного ВНЗ "НГУ" Л.І. Цвіркуном, кандидатом технічних наук, професором кафедри автоматизації та комп'ютерних систем (АКС) та асистентом кафедри АКС Р.В. Липовим, складається з 5-ти частин і 19-ти розділів. Розділи 1–3, 9–14 і 17–19 та загальне редагування виконано Л.І. Цвіркуном, а розділи 4–8, 15 і 16 написані Р.В. Липовим.

Зауваження і побажання стосовно матеріалів книги будуть із вдячністю прийняті за адресою: кафедра автоматизації та комп'ютерних систем, просп. К. Маркса, 19, Державний ВНЗ "Національний гірничий університет", м. Дніпропетровськ, Україна, 49005, або за e-mail: TsvirkunL@gmail.com.

ВСТУП

У 1957 році в СРСР запускається перший у світовій історії штучний супутник Землі. Ця подія вважається початком технологічної гонки між СРСР і США і змушує вчених цих країн розробляти нову систему зв'язку. Так у 1969 р. було створено єдину мережу США – ARPANET, а потім у 1986 р. – глобальну комп'ютерну мережу й Інтернет [1].

Всесвітня комп'ютерна мережа не була першим відкриттям подібного роду. У свій час створення радіо відкрило цілу епоху в спілкуванні між людьми, а поява Інтернету дала поштовх для розвитку нових технологій і спричинило глубокі зміни у світі [2]. Сьогодні Інтернет розвивається за такою самою схемою, як і його попередники: телеграф, радіо і телебачення. Однак, на відміну від них, він об'єднав у собі їхні достоїнства, став не тільки корисним для зв'язку, але й загальнодоступним засобом для одержання інформації.

Сучасне програмування бере початок ще в 40-х роках минулого століття, коли з'явилися перші обчислювальні машини [3, 4]. Однак мова програмування PHP виникла і почала розвиватися тільки в 1990-х роках після появи і бурхливого розвитку глобальних компьютерних мереж та Інтернету.

Спочатку мова PHP була розроблена як інструмент для вирішення суто практичних завдань. Її творець, Расмус Лердорф, бажав знати кількість людей, які читають його online-резюме, і написав для цього просту CGI-оболонку мовою Perl [5]. Цей набір програм був призначений винятково для певної мети – збору статистики стосовно відвідувань на його веб-сторінці.

Як відомо CGI (Common Gateway Interface – загальний інтерфейс шлюзів) є стандартом, що призначений для створення серверних програм, які працюють за протоколом HTTP. Такі програми запускаються сервером у режимі реального часу, який передає запити користувача CGI-програмі, що їх обробляє і повертає результат своєї роботи на екран користувача. Таким чином, відвідувач одержує динамічну інформацію, що змінюється в результаті впливу різних факторів. Сама CGI-програма може бути написана на різних мовах програмування – Ci/C++, Fortran, Perl, TCL, UNIX Shell, Visual Basic, Python та ін.

Через деякий час з'ясувалося, що оболонка має невисоку продуктивність. З цієї причини довелося переписати її заново, але вже мовою Сі. Після цього вихідні тексти програми були подані на сайті для загального огляду, виправлення помилок і доповнень. Відвідувачі сайту, де була розташована програма з першою версією PHP, зацікавилися цим інструментом. З'явилися бажаючі її використати. Незабаром мова PHP перетворилася в самостійний проект, і на початку 1995 року вийшла перша відома версія продукту, що називалася Personal Home Page Tools – засоби для персональної домашньої сторінки [6, 7].

Засоби ці були більш ніж скромними. До їхнього складу входили:

– аналізатор коду, що розуміє всього лише кілька спеціальних команд;

– набір утиліт, корисних для створення гостьової книги, лічильника відвідувань, чата і т.п.

До середини 1995 року після ґрунтовної переробки з'явилася друга версія продукту. Вона одержала назву PHP/FI (Personal Home Page / Forms Interpreter – персональна домашня сторінка / інтерпретатор форм). Цей продукт включав набір базових можливостей нинішньої мови PHP:

– автоматична обробка html-форми;

– вбудовування в html-коди.

Синтаксис PHP/FI сильно нагадував синтаксис Perl, але був більше простим.

У 1997 році вийшла друга версія Сі-реалізації PHP – PHP/FI 2.0. До того моменту в 1% Інтернет-сайтів в усьому світі застосовувалася мова PHP.

Офіційно версія PHP/FI 2.0 вийшла тільки в листопаді 1997 року, проіснувавши до цього в основному в тестових сайтах. Незабаром її замінила PHP 3.0.

PHP 3.0 була першою версією, що нагадувала нинішню PHP і дуже сильно відрізнялася від PHP/FI 2.0. Її також створено, як інструмент для вирішення конкретного прикладного завдання.

Энді Гутманс (Andi Gutmans) і Зів Сураскі (Zeev Suraski) – її творці – в 1997 році переписали заново код PHP/FI. Це відбулося з тієї причини, що він здався їм непридатним для розробки програм електронної комерції, над яким вони працювали. Для того щоб одержати допомогу в реалізації проекту від розроблювачів PHP/FI, Гутманс і Сураскі вирішили об'єднатися з ними й оголосити PHP 3.0 офіційним спадкоємцем PHP/FI. Після об'єднання розробка PHP/FI була повністю припинена.

Одна із сильних сторін PHP 3.0 – можливість розширення ядра програми. Саме властивість розширюваності PHP 3.0 привернуло увагу багатьох розроблювачів, які бажали додати свій модуль розширення. Крім того, технологія PHP 3.0 надавала широкі можливості для взаємодії з базами даних, різними протоколами та API.

Вирішальним кроком до успіху виявилася розробка нового, повного і набагато могутнішого синтаксису з підтримкою об'єктно-орієнтованого програмування.

З моменту появи PHP 3.0 змінилася не тільки функціональність і внутрішній склад мови, але і назва. В абревіатурі PHP більше не було згадування про персональне використання. PHP стало скороченням від “Php: Hypertext Preprocessor”.

До кінця 1998 року число користувачів PHP зросло. Сотні тисяч веб-сайтів повідомляли про те, що вони працюють із використанням цієї мови. Майже 10% Інтернет-сайтів застосовували мову PHP 3.0.

Офіційно PHP 3.0 вийшла у червні 1998 року, після 9 місяців публічного тестування. А вже до зими Энді Гутманс і Зів Сураскі почали переробку ядра PHP. Вони прагнули збільшити продуктивність роботи складних додатків і поліпшити модульність коду програми.

Нове ядро вперше з’явилося в середині 1999 року. Воно одержало назву «Zend Engine» (від імен творців Zeev і Andi).

Проте робота тривала далі. Так, майже через два роки після свого попередника (PHP 3.0), у травні 2000 року з’явилася версія PHP 4.0, заснована на цьому ядрі, разом із додатковими функціями.

Крім поліпшення продуктивності, PHP 4.0 мала ще кілька ключових нововведень. Ось деякі з них:

– підтримка сесій;
– буферизація виводу;
– більш безпечні способи обробки інформації, що вводить користувач;
– кілька нових мовних конструкцій.

У цей час ведуться роботи з поліпшення Zend Engine і впровадження нововведень у PHP 5.0. Одне з істотних змін відбулося в об'єктній моделі мови.

Сьогодні мова PHP використовується сотнями тисяч розроблювачів. Кілька мільйонів сайтів написані цією мовою.

Творці PHP говорять, що мова PHP може все.

Однак можна виділити три основні сфери її застосування.

Перша – створення додатків, які виконуються на стороні сервера.

PHP має широкі можливості саме для створення такого роду додатків. При цьому для написання PHP-коду додатка застосовується текстовий редактор, а для використання цих додатків необхідні:

– PHP-парсер (програма-оброблювач php-програм);

– веб-сервер (програма для взаємодії з користувачем);

– браузер (програма для перегляду результатів роботи додатка).

PHP-парсер поширюється у вигляді CGI-програми або серверного модуля. Установлення PHP-парсера, веб-сервера і браузера на комп'ютер буде розглянуто далі.

У цій книзі подано і проаналізовано приклади використання мови PHP при створенні саме серверних додатків.

Друга – створення програм, що виконуються в командному рядку.

Іншими словами за допомогою PHP можна створювати такі додатки, які будуть виконуватися поза залежністю від веб-сервера і браузера. Для такої роботи буде потрібно тільки PHP-парсер. Даний підхід називають інтерпретатором командного рядка (cli – command line interpreter).

Цей спосіб роботи підходить, наприклад, для програм, які повинні виконуватися регулярно за допомогою різних планувальників завдань або для вирішення завдань простої обробки тексту.

Третя – створення GUI-додатків (графічних інтерфейсів), що виконуються на стороні клієнта.

До речі, це не найкращий спосіб використати PHP, особливо для початківців. На нього варто звернути увагу, якщо ви досконально вивчили PHP. Такі можливості мови можуть виявитися досить корисні. Для застосування PHP у цій області потрібний спеціальний інструмент – PHP-GTK, що є розширенням PHP.

Отже, області застосування PHP досить великі й різноманітні. Проте існує безліч інших мов програмування, здатних вирішувати схожі завдання.

Чому все ж таки варто вивчати PHP?

По-перше, мова PHP дуже проста у вивченні. Досить ознайомитися лише з основними правилами синтаксису і принципами його роботи, і можна починати писати власні програми. При цьому вирішення подібних завдань на іншій мові вимагало б серйозної підготовки.

По-друге, PHP підтримується майже на всіх відомих платформах, майже в усіх операційних системах і на найрізноманітніших серверах. Це теж дуже важливо. Адже навряд чи комусь захочеться перемінити звичну операційну систему або веб-сервер тільки для того, щоб вивчити ще одну мову програмування.

В PHP сполучаються дві найпопулярніші парадигми програмування – об'єктна і процедурна.

В PHP 4.0 більше підтримується процедурне програмування, але є можливість писати програми і в об'єктному стилі.

У перших пробних версіях PHP 5.0 вже усунута більшість недоліків PHP 4.0, існуючих у реалізації об'єктно-орієнтованої моделі мови. Таким чином, можна вибрати найбільш звичний стиль роботи при використанні мови PHP.

Якщо говорити про можливості теперішньої мови PHP, то вони виходять далеко за рамки тих, що були реалізовані в її перших версіях. За допомогою PHP можна створювати зображення, PDF-файли, флеш-ролики. Вона може підтримувати велику кількість сучасних баз даних, має вбудовані функції для роботи з текстовими даними будь-яких форматів, у тому числі XML, і функції для роботи з файловою системою. PHP підтримує взаємодію з різними сервісами за допомогою відповідних протоколів.

Для роботи з текстовою інформацією PHP успадкувала механізми роботи з регулярними конструкціями з мови Perl і UNIX-систем. Для обробки XML-документів можна використати як стандарти DOM і SAX, а для XSLT-трансформацій – API.

Існує ряд корисних функцій, таких як Cybercash, CyberMUT, VeriSign Payflow Pro і CCVS. Їх застосовують для створення додатків електронної комерції.

ЧАСТИНА 1. ГЛОБАЛЬНІ КОМП'ЮТЕРНІ МЕРЕЖІ
1. ОСНОВИ ГЛОБАЛЬНИХ КОМП'ЮТЕРНИХ МЕРЕЖ

Навчальною метою розділу є ознайомлення студентів з основами глобальних комп'ютерних мереж.

У результаті вивчення даного розділу студенти повинні знати:

- основні принципи побудови мережі Інтернету;
- історичні факти розвитку комп'ютерних мереж;
- поширені протоколи передачі даних;
- тенденції розвитку мережі Інтернет.

Інтернет – глобальна комп'ютерна мережа, що охоплює всю територію земної кулі, яка працює за протоколом TCP/IP. Однак це лише частина відповіді на питання «що таке Інтернет?». Інтернет – це не тільки величезна кількість комп'ютерів, але ще і неймовірна кількість людей, для яких мережа є принципово новим засобом спілкування, що майже не має аналогів у світі.

Людина – істота соціальна, і спілкування із собі подібними – одна з найперших її потреб. Мабуть, дотепер ще жодний технічний винахід, окрім телефону, не викликав такого перевороту.

Звичайно, що саме зацікавить вас в мережі Інтернет у першу чергу – люди або комп'ютери, залежить тільки від вас. Але не буде перебільшенням сказати, що, одержуючи доступ до мережі Інтернет, ви робите для себе доступним цілий світ.

Винахід і вдосконалювання модемів – спеціальних пристроїв, що дозволяють комп'ютеру посилати інформацію зі звичайної телефонної лінії, відкрило двері в Інтернет величезній кількості людей, у яких немає ніякого спеціального устаткування, а є лише персональний комп'ютер і телефонна розетка поблизу.

Всі комп'ютери мережі Інтернет можна розділити на два типи: сервери і клієнти. Комп'ютери користувачів є комп'ютерами-клієнтами мережі Інтернет, тому що вони використовують ресурси Інтернету. Сервери утворюють основу мережі, надають свої ресурси у використання іншим комп'ютерам.

Коли говорять, що комп'ютер підключений до мережі Інтернет, це означає, що цей комп'ютер за допомогою одного з основних засобів зв'язку – модему (Dial-Up підключення) або мережної карти – з'єднаний із провайдером (службою доступу в Інтернет) і може звернутися до будь-якого сервера мережі Інтернет. А під терміном Інтернет у цьому випадку розуміють безліч серверів, до яких комп'ютери користувачів мають доступ і ресурсами яких можуть користуватися.

У мережі Інтернет користувачі підключаються до різних серверів і одержують необхідну інформацію. Інтернет – це складна структура зв'язаних між собою комп'ютерних мереж, що дозволяє їм мати доступ до всіх комп'ютерів мережі.

1.1. Історія створення глобальних комп’ютерних мереж

Інтернет починався аналогічно більшості сучасних технологій як військова програма, спрямована на підвищення стійкості системи оборони США. Майже 50 років тому, після запуску першого радянського штучного супутника Землі, американський центр RAND Corporation був поставлений перед складною стратегічною проблемою керування країною під час і після ядерної війни.

Країні, що могла одержати ядерний удар, потрібна була надійна мережа передачі даних, яка буде справно функціонувати при втраті значної частини устаткування цієї самої мережі.

У 1958 році при Міністерстві оборони США було створене Агентство Передових дослідницьких проектів – Advanced Research Projects Agency (ARPA). ARPA, зокрема, проводило дослідження в галузі безпеки зв'язку і комунікацій у ході обміну ядерними ударами.

В 1963 році керівник комп'ютерної лабораторії ARPA Джон Ліклідер пропонує першу детально розроблену концепцію комп'ютерної мережі.

А в 1964 р. RAND опублікував свої пропозиції, які полягали в тому, що:

- мережа не повинна бути централізованою;
- вона має складатися з окремих сегментів.

Таким чином, кожен вузол мережі повинен бути незалежним від інших вузлів, тобто самостійно відповідати за прийом/передачу повідомлень.

В основу інформаційного обміну був покладений принцип комутації пакетів: будь-яке інформаційне повідомлення ділиться на частини, які називаються пакетами, кожен пакет забезпечується адресою. Пакети передаються по мережі й збираються в повідомлення у вузлі-одержувачі. Якийсь із пакетів може і втратитися, але повідомлення в цілому має більші шанси знайти адресата. Із самого початку передбачалося, що для прийому/передачі інформації можуть використовуватися будь-які канали зв'язку (радіо, телефонні, виділені лінії і т.п.).

На початку 60-х років мережа, яка заснована на комутації пакетів, об'єднала RAND, Масачусетський Технологічний Інститут і Каліфорнійський університет. А в 1968 р. до неї приєдналася ще й Національна фізична лабораторія Великобританії.

У 1969 р. агентство перспективних досліджень Міністерства оборони США вирішило об'єднати суперкомп'ютери оборонних, наукових і керуючих центрів у єдину мережу – ARPANET. На той час у мережі було тільки чотири комп'ютери (хоста), але на початку 1971 року – чотирнадцять, а наприкінці – тридцять сім.

Цього ж року інженер однієї з американських компаній Рей Томлісон розробляє систему електронної пошти й відправляє з одного комп'ютера на інший послання "QWERTYUIOP" у мережі Інтернет.

Сімдесяті роки минулого століття – це період росту і налагодження технологій Інтернет. Згодом з'ясувалося, що основне навантаження в мережі

становлять комунікаційні повідомлення (пошта і новини). Отже, починає інтенсивно розвиватися система електронної пошти і телеконференцій.

Передбачалося, що комп'ютерна мережа ARPANET об'єднає внутрішні мережі ряду дослідницьких лабораторій і університетів США, що працюють на оборону. У рамках цього проекту американським дослідником Вінтоном Серфом (Vinton Cerf) був розроблений первісний варіант протоколів TCP (Transmission Control Protocol, протокол керування передачею даних) та IP (Internet Protocol, мережевий протокол). Перший описував спосіб поділу інформаційного повідомлення на пакети і їхню передачу, а другий управляв адресацією в мережі. Ці два протоколи дали назву всьому сімейству протоколів міжмережевого обміну, розробленому в рамках Інтернету, – сімейство протоколів TCP/IP. Комп'ютерна мережа ARPANET швидко розвивалася і у 1977 році число її хостів досягло ста.

Так, у 1974 році відкрита перша комерційна версія ARPANET – мережа Telenet.

А вже через два роки Роберт Меткалф, співробітник дослідницької лабораторії компанії Xerox, створює Ethernet – першу локальну комп'ютерну мережу.

У 1980 році письменник Алвін Тофлер опублікував книгу “The Third Wave”. У ній був описаний постіндустріальний світ, у якому основну роль грають інформаційні технології. Тофлер А. оцінив перспективи розвитку комп'ютерних мереж і зробив припущення, що в майбутньому така мережа зможе об'єднати увесь світ. При цьому користувачі комп'ютерної мережі, як всі власники телевізорів, зможуть дивитися однакову передачу. За прогнозом Тофлера А. інформаційні технології дадуть людям незрівнянно більше можливостей, ніж звичайне телебачення.

У 1982 році ARPA створила єдиний мережевий протокол TCP/IP, що був покладений в основу сучасного Інтернету.

З 1984 року національний науковий фонд США почав вкладати значні кошти в наукову комп'ютерну мережу NSFNET. Ця мережа об'єднала в собі наукові центри та університети США. Як основа мережі були обрані протоколи сімейства TCP/IP. У цей час до NSFNET примкнули NASA, DOE і National Institutes of Health, число хостів мережі перевищило 1000.

Реально становлення глобальної комп'ютерної мережі США Інтернет відбулося в 1986 році.

А через три роки на іншому березі Атлантики сталася інша важлива для Інтернету подія – була утворена організація RIPE (Reseaux IP Europeans), покликана забезпечити адміністрування і технічне координування Інтернету у рамках Європи. Число хостів мережі перевищило 10000.

У 1991 році Європейська фізична лабораторія CERN розробила сервіс www (world wide web). Хоча він був створений, насамперед, для обміну інформацією серед фізиків, ним стали користуватися й інші.

Вже через деякий час виходять на ринок графічні переглядачі (браузери) веб-сторінок Mosaic (1993) та Netscape Navigator (1994).

Надалі стане ясно, що своєму нинішньому розквіту Інтернету безперечно зобов'язаний початку практичного використання www.

Поява www, супутника Інтернету, революційно змінило відношення масового користувача до мережі. Цю подію можна зрівняти з появою графічного інтерфейсу на тлі засилля малозрозумілих текстових.

Інтернет став демонструвати дивовижні темпи росту. Ємність Інтернет виросла від 50 веб-серверів на початку 1993 до 600 веб-сторінок і 2 млн хостів уже до кінця року.

У Росію, а потім і в Україну (у той час республіки СРСР) Інтернет уперше проникнув на початку 90-х років. Ряд університетів і дослідних інститутів приступили в цей час до побудови своїх комп'ютерних мереж, які були зв'язані із закордонними. Особливо слід зазначити Інститут Атомної Енергії ім. І.В. Курчатова. На базі ІАЕ склалися дві найбільші комерційні компанії (Релком і Демос), що надавали послуги з підключення до Інтернету.

Користувачі отримували в основному сервіси електронної пошти і телеконференцій з використанням протоколу UUCP.

В 1996 році у світі існувало 12,8 млн хостів мережі Інтернет і 500 тис. веб-сторінок, а у 2002 ця мережа зв'язує 689 млн користувачів і 172 млн хостів.

Сьогодні вже розробляються нові технології Інтернету, які повинні замінити "старий Інтернет", тобто розширити його функції.

1.2. Сервіси та тенденції розвитку глобальних комп'ютерних мереж

1.2.1. Перші сервіси Інтернету

Перші сервіси глобальних комп’ютерних мереж повнісю відповідали новим розробкам протоколів, які з’являлись і починали працювати в комп’ютерних мережах.

Розглянимо ці сервіси.

Telnet

Одним з перших застосувань Інтернету став так званий віддалений доступ до комп'ютера.

Користувач одного з комп’ютерів мережі міг підключитися (віддалено) до іншого комп’ютера, який теж підключений до мережі, при цьому операційна система надавала користувачу всі ті послуги, які він отримував на цій машині.

Програма та протокол для такого віддаленого доступу у межах мережі були названі telnet.

Telnet – найпростіший і тому найуніверсальніший інструмент Інтернету. Можна взяти будь-яку програму і, призначивши цій програмі ім'я і пароль на одному з вузлів (серверів), оголосити вільний доступ до неї через telnet. Після цього будь-який користувач Інтернету, що знає ім'я і пароль цієї програми, зможе працювати з нею віддалено. Єдина вимога – ця програма повинна вміти працювати в текстовому режимі, без використання графіки.

Саме так було організовано в 70 – 80-ті роки минулого століття безліч баз даних з вільним доступом через Інтернет та інших різноманітних інформаційних ресурсів.

Сам по собі telnet – просто засіб зв'язку, який не має ні власного інтерфейсу, ні можливостей пошуку. Зв'язавшись з допомогою telnet з віддаленою машиною користувач вводив за її запитом свої ім'я і пароль і далі спілкувався саме з цією машиною і програмами на ній. Telnet дбає лише про підтримку зв'язку між комп'ютером користувача і сервером: передає те, що набирає користувач на клавіатурі й повертає назад все, що генерують програми з комп'ютера сервера (звичайно, можливості такого спілкування сильно обмежені текстовим режимом дисплея).

FTP

Практично вся інформація в комп'ютерному світі зберігається у вигляді файлів. Комп'ютерам працювати з файлами зручніше, а для користувачів написано безліч програм, які виводять на дисплей вміст файлів у красивій, зручній і зрозумілій формі. Втім повністю ізолювати користувача від спілкування з «файлами як такими» нікому ще не вдавалося, та це й не потрібно.

Однак перш ніж звертати увагу на вміст файлів, потрібно забезпечити найважливіше – можливість копіювання цих файлів з одного вузла Інтернету на інший.

Через це на самому початку розвитку Інтернету з'явився спеціальний засіб для обміну файлами у мережі – протокол FTP (що, власне, і розшифровується як File Transfer Protocol – протокол передачі файлів).

Порівняно з telnet, FTP є вже більш складний інструмент, із власним набором команд. Це й зрозуміло, адже перш ніж одержати файл із віддаленої машини, непогано б спочатку переконатися, що він там є, а ще раніше перейти в той каталог, де цей файл знаходиться. Програма-клієнт FTP має особливі команди не тільки для цих базових дій (перехід у каталог на віддаленій машині, перегляд його вмісту, одержання файлу), але і для деяких допоміжних функцій.

Кількість файлів, які доступні на вузлах усього світу на анонімних FTP-серверах, виміряється астрономічними цифрами і постійно зростає. Протокол FTP має одне важливе достоїнство: щоб зробити файл доступним усім бажаючим, від адміністратора потрібен мінімум зусиль – він повинен просто покласти цей файл в один з каталогів, який бачать анонімні користувачі.

E-mail

Електронна пошта (e-mail) – також одна з перших служб Інтернету, що з'явилася в середині 70-х років. Основна концепція, що лежить в основі електронної пошти, досить проста: ви виходите в комп'ютерну систему, набираєте й адресуєте текстове повідомлення користувачеві іншої системи. Потім повідомлення курсує по лабіринту взаємозалежних комп'ютерних систем доти, поки не буде доставлене адресатові.

Хоча основні принципи роботи не змінилися, сучасні програми електронної пошти мало чим нагадують системи 70 – 80-х років.

Зараз електронна пошта дозволяє посилати текстові повідомлення, але тепер у повідомлення можна вкладати файли різних типів і шифрувати повідомлення для запобігання несанкціонованого доступу. Зараз в Інтернеті існують безкоштовні служби, розроблені винятково для роботи електронної пошти, які можна використовувати в тих випадках, якщо Інтернет застосовується тільки як можливість передачі електронних повідомлень.

UseNet

UseNet, або групи новин **(**телеконференції**),** являють собою службу, що чимось нагадує електронну пошту, але в ній повідомлення посилають не одній конкретній особі, а спрямовуються в сферу спільного доступу, тобто телеконференції, з якої повідомлення можуть получити багато користувачів і дати свою відповідь.

UseNet почала працювати в 1979 році як сервіс, що з'єднує комп'ютери університетів Дьюка (Duke) і Північної Кароліни (North Caroline). У цей час UseNet була надзвичайно популярною службою Інтернет, що охоплювала більше 4000 тем, на які користувачі відсилали повідомлення й одержували відповіді. Це могли бути найрізноманітніші теми: обговорення комп'ютерних і технічних питань, а також соціальних, релігійних і політичних дискусій, музики, книг і кіно.

Телеконференції являють собою щось подібне до клубів за інтересами, що об'єднують користувачів певної програми, шанувальників відомих кінорежисерів або популярних телеведучих, інших людей, пов'язаних спільними нтересами і поглядами.

Якщо для електронної пошти не складно знайти аналогію в «матеріальному» світі, то система телеконференції UseNet схожа відразу на все й у той же час не схожа ні на що. Деякі порівнюють її з газетою, інші – з дошкою оголошень, треті – з дискусійним клубом. І якщо «паперова» пошта під час відсутності Інтернету могла б частково замінити електронну, то багатомільйонної аудиторії UseNet у світі без комп'ютерів просто б не існувало.

World wide web (www)

WWW – це гіпертекстова (гіпермедіа) система, яка призначена для інтеграції різних мережних ресурсів у єдиний інформаційний простір.

Найменшою частинкою всесвітньої гіпертекстової павутини є окремий файл з яким-небудь текстом і посиланнями, розмічений мовою HTML.

HTML (Hyper Text Markup Language) – мова гіпертекстової розмітки. Файл, розмічений мовою HTML, за форматом є звичайним ASCII-файлом. Такий файл має власну унікальну URL-адресу, і його можна посилати по мережі тільки цілком. Звичайно ці файли називають просто HTML-файлами або HTML-документами.

Однак для практики важливіше інший рівень побудови www, на якому весь уміст цієї системи складається з www-сторінок (sities).

Цей рівень розподілу – більш значний, ніж формальний: до одної «сторінки» відносять набір файлів різних форматів, основним з яких є HTML.

Безліч таких сторінок, які розташовані на одному комп'ютері та тісно переплетені взаємними посиланнями й притому зв'язані за змістом (наприклад, що містять інформацію про одну фірму, одну людину тощо), утворять www-сервер.

Кожний www-сервер має свій «вхід» або «домашню сторінку» («home page») – HTML-документ, у якому зібрані посилання на головні складові частини сервера і адреса якого розповсюджується через каталоги, рекламу й т.ін. як адреса цього сервера.

Сторінки одного www-сервера пов'язані зі сторінками інших серверів Інтернету. Таким чином, взаємозалежні www-сервери утворять єдиний інформаційний ресурс Інтернет.

Компанія Microsoft розробила операційну систему Windows з урахуванням потреби користувачів у послугах Інтернету і включила в пакет програм кілька спеціальних утиліт для забезпечення доступу до Інтернету. Зокрема, програму-браузер Microsoft Інтернет Explorer.

Браузер – це програма для перегляду ресурсів мережі Інтернет, тобто переглядач, що, за своєю природою, являє собою програму-клієнт.

Строго кажучи, програма-переглядач – це www-клієнт. Але www-клієнт не обов'язково повинен бути переглядачем – це може бути наприклад, програма-робот, що збирає необхідну інформацію в мережі Інтернет.

Якщо бути більш точним, переглядач – це клієнт www, який призначений для інтерактивної роботи з користувачем (а іноді виконуючий і деякі сторонні функції, наприклад обробку електронної пошти).

Зараз на ринку існує безліч програм-переглядачів від різних фірм-виробників, але одними з перших і найпопулярніших у всіх країнах світу були Mosaic і Netscape Communicator.

1.2.2. Спілкування в Інтернет

Будь-які засоби комунікації служать для спілкування людей між собою. Не є винятком і комп'ютерні мережі взагалі, й Інтернет зокрема. Саме можливість для будь-якого користувача Всесвітньої мережі легко спілкуватися з іншими людьми, що навіть знаходяться на іншому кінці планети, зробила її настільки популярною. Для реалізації такого спілкування існує безліч програмних засобів, що розрізняються функціональними можливостями, доступністю використання, необхідними мережними ресурсами й іншими параметрами.

IRC

IRC (Інтернет Relay Cat – трансляція розмови в Інтернеті) – надзвичайно популярний сервіс Інтернету. Як припускає назва, IRC являє собою систему, що дозволяє користувачам спілкуватися один з одним, підключившись до одного сервера IRC. Однак у ході спілкування користувачі не говорять, а набирають свої репліки на клавіатурі.

Якщо ще років п’ять тому інтерактивні бесіди виділялися в окрему службу Інтернету, то зараз така можливість надається користувачам багатьох Web-вузлів.

Не слід розраховувати на те, що в IRC часто і густо ведуться високоінтелектуальні дискусії за участю нобелівських лауреатів і відомих учених у галузі космічних досліджень. Значна частина дискусійних груп (які називаються ще chat rooms – кімнати для бесід) формується навколо останніх спортивних подій, політичних скандалів й інших тем, обговорюваних, звичайно, у колі компанії.

ICQ

ICQ – це також дуже популярний засіб «онлайнового» спілкування користувачів через Інтернет. Назва цієї програми є омонімом й утворена зі співзвуччя слів «I seek you» («Я шукаю тебе») і трибуквеного сполучення ICQ.

IP-телефонія

IP-телефонія – послуга, яка заснована на перетворенні голоса людини в пакети, які можуть бути передані через Інтернет.

Сьогодні розвиток цієї технології досяг такої стадії, що послуги IP-телефонії стали досить надійними та якісними, щоб розглядатися як альтернатива послугам традиційного телефонного зв'язку.

Більшість клієнтів використовують нову технологію для зниження вартості послуг телефонного зв'язку при збереженні, якщо можна, його якості без введення додаткових обмежень для користувачів. Економія коштів відбувається, головним чином, за рахунок компресії голосу, інтеграції мереж передачі голосу й даних і за рахунок більш привабливих (на сьогодні) тарифів Інтернету порівняно з послугами традиційних телефонних операторів.

Послугами IP-телефонії особливо часто користуються ті, кому доводиться вести часті розмови із заокеанськими співрозмовниками.

Для зручності застосування нової технології використовуються спеціальні IP-телефони. Маючи такий телефон, не потрібно ні комп'ютер (процесор і модем уже вбудовані в нього), ні спеціальне програмне забезпечення, ні особливий канал, крім Інтернет.

Таким чином, міжнародні/міжміські дзвінки стають значно дешевшими (іноді треба платити тільки за доступ до Інтернету).

DNS

Сервіс DNS, або система доменних імен, забезпечує можливість використання для адресації вузлів мережі мнемонічні імена замість числових адрес.

1.2.3. Сучасні сервіси

Сучасні сервіси Інтернету пов'язані з подальшим використанням його у всіх галузях життя, включаючи роботу, навчання і відпочинок. Більшість цих сервісів базуються на сервісі www або на його інтеграції з іншими, користувачі отримують їх за допомогою програми-переглядача.

Слід також зазначити, що збільшилася різноманітність пристроїв, які використовують Інтернет. Це тепер не тільки комп'ютери і ноутбуки, а з'явилися мобільні телефони з графічним кольоровим інтерфейсом і підтримкою багатьох сервісів Інтернету, КПК (кишенькові персональні комп'ютери), планшети, гаджети й т.п.

Веб-портал

Розвиток веб-серверів привело до появи так званих веб-порталів. Відмінністю їх є те, що доступ до всієї або частини інформації забезпечується для деякої вибраної групи користувачів. Ця група може визначатися спільним місцем роботи або відпочинку, за видом одержуваної інформації і т.п.

Пошук інформації

В Інтернеті інформація не впорядкована і не існує єдиного каталогу збереженої інформації або файлів. Для пошуку необхідної інформації в мережі Інтернет існують пошукові портали. Іноді функція пошуку інформації є одним з можливих сервісів веб-порталу.

Функції пошуку інформації виконують такі найбільш відомі портали:

- http://www.bigmir.net;
- http://www.google.com;
- http://www.meta.ua;
- http://www.rambler.ru;
- http://www.yandex.ru;
- http://www.ya.ru;
- http://www.yahoo.com.

E-leaning (е-навчання)

Основна концепція, що лежить в основі електронного навчання (е-навчання), досить проста: Інтернет, як правило, використовується для виклику розміщеної на серверах інформації у формі файлів, які користувачем, так званим „клієнтом", будуть переглянуті, прослухані, роздруковані, збережені, перероблені або запущені до виконання.

Це також можливо, коли, наприклад, у межах курсу е-навчання матеріал у придатній формі „викладено в мережі". Таким чином, з багатьох місць, у тому числі і з вищих навчальних закладів, можна буде дістатися таких матеріалів, як конспекти лекцій у вигляді файлів на веб-серверах. При цьому мова може йти про інтерактивний процес, коли користувач (наприклад студент) має відповісти на запитання або вирішити поставлені завдання, а відповіді у режимі он-лайн надіслати назад.

Результати потім за необхідності можуть бути збережені у формі файлів або записів у базі даних, вручну або (якщо можна) автоматично перевірені, а оцінка або корективи знов направлені користувачу. Звісно, кроки навчання до того ж здійснюватимуться залежно від успішного проходження заданих рівнів або інших дій користувача так, щоб загалом вийшла єдина інтерактивна навчальна система.

Навчання у реальних системах

Лабораторні випробування, що базуються на Інтернеті, як частину концепції е-навчання неодмінно використовують описані механізми, але також ідуть на один істотний крок далі [8]. Тут мова йде про інтерактивний доступ до пристроїв, у тому числі таких складних систем, як роботи, верстати, приводи, пристрої регулювання та керування, системи обробки зображень, мікроскопи, пристрої діагностики, пристрої обробки тощо (рис. 1.1).

Різноманітності тут не встановлено жодних меж. На основі експериментів користувач здатен ці пристрої через Інтернет ввести у дію, обслуговувати, спостерігати, програмувати, ініціювати виконання деяких технологічних операцій, збирати дані в різних формах (системні дані, фізичні величини, дані зображень, дані вимірювань, часові величини тощо), інтерпретувати дані й характеристики системи.

За допомогою інтерактивної роботи з телевипробуваннями отримані знання можуть бути застосовані та у реальності перевірені, теорія і практика – порівняні, а нові знання здобуті та закріплені. Дія як важливий елемент успішного навчання стоїть тут на передньому плані.

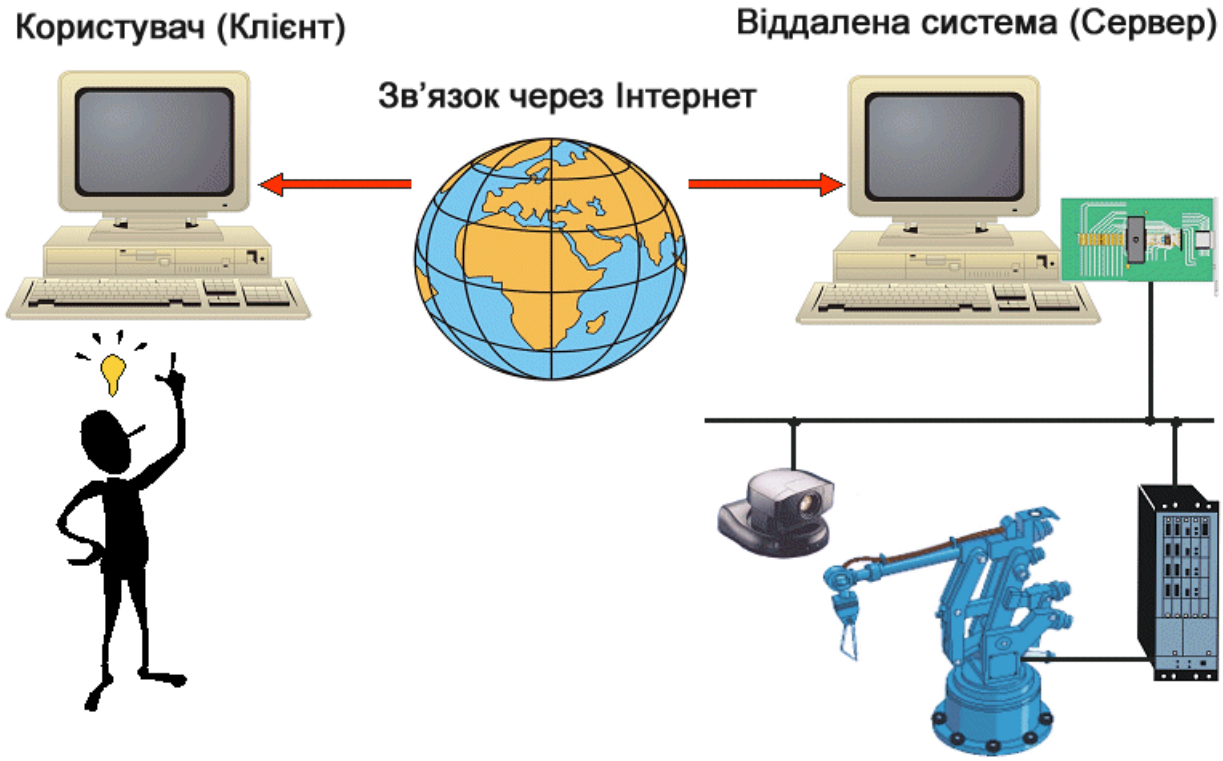


Рис. 1.1. Навчання у реальних системах через Інтернет

Стандартні порти компонент сервісів

Для стандартних сервісів також стандартизується й інтерфейс взаємодії із протоколами транспортного рівня. Зокрема, за кожним програмним сервером резервуються стандартні номери TCP- і UDP-портів, які лишаються незмінними незалежно від особливостей тієї або іншої фірмової реалізації як компонентів сервісу, так і транспортних протоколів. Номери портів клієнтського програмного забезпечення так жорстко не регламентуються. Це пояснюється такими факторами:

– по-перше, на користувацькому вузлі може функціонувати кілька копій клієнтської програми, і кожна з них повинна однозначно ідентифікуватися транспортним протоколом, тобто за кожною копією повинен бути закріплений свій унікальний номер порту;

– по-друге, клієнтові важлива регламентація портів сервера, щоб знати, куди направляти запит, а сервер зможе відповісти клієнтові, впізнавши адресу із запиту, що надійшов.

У наведеній нижче таблиці вказані стандартні номери портів для основних сервісів.

Таблиця 1.1

Стандартні номери портів серверів для основних сервісів

Компонент сервісу	Номер порту	Транспортні протоколи
Telnet		
Telnet-сервер	23	TCP
	FTP	
FTP-сервер	20, 21	TCP
E-mail		
SMTP-сервер	25	TCP
POP3-сервер	110	TCP
IMAP-сервер	143	TCP
WWW		
HTTP-сервер	80	TCP
DNS		
DNS-сервер	53	TCP, UDP

1.2.4. Тенденції розвитку

Інтернет – глобальна комп'ютерна мережа, що об'єднує мільйони комп'ютерів і локальних мереж у всьому світі. Більшість комп'ютерів в Інтернеті самі є частиною більш дрібних мереж, що належать різним компаніям, університетам, урядовим організаціям і т.д.

Глобальна мережа Інтернет була створена для забезпечення обміну інформацією між віддаленими комп'ютерами. З появою Інтернету у мільйонів людей з'явилася можливість одержувати інформацію з компетентних джерел, обмінюватися файлами і просто спілкуватися незалежно від місцезнаходження.

24 жовтня 1995 року Федеральна мережна рада (FNC) США затвердила визначення терміна Інтернету. Він розроблявся за участі фахівців у галузі мереж і в галузі прав на інтелектуальну власність.

Федеральна мережна рада США визнала, що таке формулювання відображає зміст терміна Інтернету:

Інтернет – це глобальна інформаційна система, яка:

– логічно взаємопов’язана простором глобальних унікальних адрес, заснованих на IP протоколі або на наступних розширеннях або спадкоємцях IP;

– здатна підтримувати комунікації з використанням сімейства протоколів TCP/IP або його наступних розширень/спадкоємців і/або сумісних IP-протоколів;

– забезпечує, використовує або робить доступною на суспільній або приватній основі сервіси високого рівня, які надбудовані над описаною тут комунікаційною та іншою пов'язаною з нею інфраструктурою.

За десятиліття свого існування Інтернет як мережа зазнала кардинальних змін. Вона зароджувався в епоху ламповихЕОМ, але зуміла вижити в часи панування персональних комп'ютерів, однорангових мереж, систем клієнт/сервер і мережних комп'ютерів. Проектувалася до перших локальних комп'ютерних мереж, але спромоглася застосувати цю нову мережну технологію, як і ті сервіси комутації пакетів і кадрів, що з’явилися пізніше. Вона підтримувала широкий спектр функцій, від розділення файлів і віддаленого входу до розподілу ресурсів та спільної роботи, породивши електронну пошту і, у більш пізній період, Інтернет. Але важливіше за все те, що мережа, яка створювалася спочатку як об'єкт діяльності невеликого колективу спеціально виділених дослідників, виросла до комерційно вигідного підприємства, у яке щорічно вкладаються мільярди доларів.

Не слід думати, що всі зміни Інтернету завершено. За назвою і географічно Інтернет є мережею, але це породження інформаційної, а не традиційної телефонної або телевізійної індустрії.

Щоб передовий рівень Інтернету зберігався, зміни повинні тривати, і вони будуть продовжені, розвиток і далі буде йти в темпі, характерному для комп'ютерної індустрії. Зміни, які відбуваються в наші дні, спрямовані так, щоб надавати нові сервіси, як передача даних у реальному масштабі часу з метою підтримки, наприклад, аудіо- і відеопотоків з більш кращою якістю.

Проведені дослідження темпів росту переданих через Інтернет даних у період з грудня 2001 по грудень 2006 року з 6-місячним інтервалом показали, що, за аналогією із Законом Мура, Інтернет подвоюється в обсязі приблизно кожні 5 років.

До початку XX століття обсяг накопичених людством знань подвоювався кожні 100 років. Тепер сумарний обсяг людських знань подвоюється кожні два–три роки, і найважливішу роль у цьому відіграє Інтернет. Досить сказати, що 70 відсотків усієї інформації з'явилося з народженням Інтернету. І чим більше людство знає, тим швидше воно поповнює запас своїх знань.

Очікується, що в 2013 році буде створено 1,2 зетабайта унікальної інформації (для запису такого обсягу даних потрібні були б 250 млрд DVD-дисків).

Розробка нової версії системи адресації в Інтернеті дозволяє надати користувачам 340 282 366 920 938 463 463 374 607 433 768 211 456 IPv6-адрес. У перерахунку на кожного жителя нашої планети це складе 52 трильйони IPv6-адрес.

Повсюдна доступність Інтернету в сполученні з потужними, компактними й доступними за ціною обчислювальними й комунікаційними засобами – це новий крок на шляху розвитку мобільних обчислень і комунікацій, хмарних технологій.

Нові режими доступу і нові форми обслуговування породять нові додатки, які у свою чергу стануть рушійною силою подальшого розвитку самого Інтернету.

Висновки

У даному розділі розглянуто такі основні питання:

- історія розвитку Інтернету;
- перші сервіси Інтернету;
- спілкування в Інтернеті;
- сучасні сервіси;
- перспективи розвитку Інтернету.

Контрольні питання

1. Що таке Інтернет?
2. Які відомі визначні дати історії Інтернету?
3. Які сервіси Інтернету були першими?
4. Яким протоколом здійснюється віддалений доступ до комп'ютера?
5. Які сервіси забезпечують спілкування в Інтернеті?
6. Який сервіс Інтернету забезпечує адресацію в мережі мнемонічними іменами замість числових адрес?
7. Які відомі сучасні сервіси Інтернету?
8. Що таке веб-портал?
9. Як забезпечується пошук інформації в Інтернеті?
10. Які тенденції розвитку Інтернету?

2. ЕЛЕМЕНТИ МОВИ HTML

Навчальною метою розділу є ознайомлення студентів із синтаксисом мови PHP.

У результаті вивчення даного розділу студенти повинні знати:

- структуру програми мовою HTML;
- теги форматування документів;
- стилі html-документів;
- особливості гіпертекстових посилань.

2.1. Структура програми мовою HTML

Hyper Text Markup Language (HTML) являє стандартну мову, призначену для створення гіпертекстових документів.

HTML-документ (програма) − це звичайний текстовий файл, що можна переглядати й редагувати в будь-якому текстовому редакторі, який не виконує форматуваня тексту. HTML-документ містить для розмічування спеціальні символи (теги).

Тег (мітка) є ключовим словом, за допомогою якого користувач може позначати інформацію в програмі або тексті. Важлива особливість тегів − можливість позначати один елемент кількома різними мітками.

У більш вузькому застосуванні тег − це ідентифікатор для категоризації, опису, пошуку даних і завдання внутрішньої структури.

Більшість документів мають стандартні елементи, такі як заголовок, параграфи або списки.

Теги HTML вказують веб-браузеру, як необхідно подати вказану інформацію на екрані дисплея. Перевага HTML полягає в тому, що документ може бути переглянутий на веб-браузерах різних типів різноманітних операційних систем.

HTML документ складається із заголовної частини й тіла документа.

Відповідно існують HTML-теги двох категорій:

- теги, що описують загальні властивості документа, які використовуються в заголовній частині документа;
- теги, що визначають, як буде відображатися веб-браузером тіло документа в цілому.

До того ж існує тег, який указує, що даний документ або його частина подані HTML-мовою.

Всі теги HTML починаються із символа “<” і закінчуються символом “>”. Як правило, існує стартовий і завершальний теги.

Завершальний тег виглядає так само, як і стартовий, і відрізняється від нього наявністю прямого слеша перед текстом усередині кутових дужок.

Веб-браузер відразу після одержання документа визначає, як і з використанням якої мови документ повинен бути інтерпретований.

Найперший тег, що зустрічається в HTML-програмі, повинен бути тегом <HTML>.

Даний тег повідомляє веб-браузеру, що документ написаний з використанням HTML. Мінімальний HTML-документ буде виглядати так:

```
<HTML> ... тіло документа ... </HTML>
```

HTML не реагує на регістр символів, що описують тег, і наведений раніше приклад може мати такий вигляд:

```
<html> ... тіло документа ... </html>
```

Деякі теги, такі як <P> (тег, що визначає абзац), не вимагають завершального тега, але його використання поліпшує читаність вихідного тексту документа та його структуру.

Тег заголовної частини документа повинен бути використаний один раз після тега <html>. Даний тег являє собою загальний опис документа.

Стартовий тег <head> поміщається безпосередньо перед тегом <title> й іншими тегами, що описують документ, а завершальний тег </head> розміщається відразу після закінчення опису документа.

Теги заголовка визначають текст, що знаходиться всередині стартового й завершального тегів заголовка, які описують документ, і мають такий вигляд:

```
<title> Заголовок документа </title>
```

У даному прикладі тег <title> вказує веб-браузеру про необхідність використання формату заголовка, а тег </title> – про завершення тексту заголовка.

Більшість веб-браузеров відображають уміст тега <title> у заголовку вікна, що містить документ, й у файлі закладок, якщо він підтримується веб-браузером. Заголовок документа не з'являється при відображенні самого документа у вікні. Заголовок, обмежений тегами <title> й </title>, розміщається всередині <head>-тегов, як показано на прикладі.

```
<html>
<head>
<title> Заголовок документа </title>
</head>
...
</html>
```

Теги тіла документа ідентифікують відображувані у вікні компоненти HTML-документа. Тіло документа може містити посилання на інші документи, текст й іншу форматовану інформацію.

Тіло документа повинне перебувати між тегами <body> й </body>. Це та частина документа, що відображає текстову й графічну (значеннєву) інформацію документа.

Технічно стартові й завершальні теги типу <html>, <head> й <body> не обов'язкові. Однак рекомендується їх використовувати, тому що дані теги дозволяють веб-браузеру впевнено розділити заголовну частину й безпосереднє тіло документа.

Мова HTML дозволяє вставляти в тіло документа коментарі, які не відображаються браузером.

Синтаксис коментарю:

```
<!--Комментар -->
```

Тепер можна написати перший HTML-документ, що виводить на екран у браузері слова "Привіт, Україно!". Для цього в будь-якому редакторі документів, який не виконує форматування текста, наприклад WordPad, наберемо такий текст.

```
<html>
<heard>
<title> Hello </TITLE>
</head>
<body>
Привіт, Україно!
</body>
</html>
```

2.2. Основні теги форматування документа

2.2.1. Форматування текстів

Текст HTML-документа структурно ділиться на просто текст, заголовки частин тексту, заголовки більш високого рівня й т.ін.

Перший рівень заголовків (найбільший) позначається цифрою 1, другий – 2 і т.д. Більшість браузерів підтримує інтерпретацію шістьох рівнів заголовків, призначаючи кожному з них власний стиль. Заголовки вище шостого рівня не є стандартними і можуть не підтримуватися браузером. Заголовок найвищого рівня має ознаку "1".

Для розмітки заголовків використаються теги <Hn> і </Hn>, де n – рівень заголовка від 1 до 6. Синтаксис заголовка рівня n такий:

```
<Hn> Заголовок першого рівня n </Hn>
```

В HTML-документі звичайно ігноруються символи повернення каретки. Фізичний розрив абзацу може знаходитися в будь-якому місці вихідного тексту документа. Однак браузер виділяє абзаци тільки при наявності тегу <p>. Тег <p> не вимагає завершального тегу.

Без використання тегу <p> документ буде виглядати, як один великий абзац.

Тег <p> має додаткові параметри, за допомогою яких текст можна вирівняти по лівому або правому краю чи центру:

```
<p align=left|center|right>
```

Тег <pre> дозволяє вставляти текст із специфічним форматуванням на екрані. Попередньо сформатований текст закінчується завершальним тегом </pre>. Усередині попередньо сформатованого тексту дозволяється використовувати:

– переведення рядка;
– символи табуляції (зсув на 5 символів праворуч);
– непропорційний шрифт, який установлений браузером.

Використання тегів, що визначають формат абзаців, таких як <Hx>, буде ігноруватися браузером при розміщенні їх між тегами <pre> й </pre>.

Складемо з використанням раніше розглянутих тегів html-документ.

Приклад 2.1

```
<html>
<heard>
<title> Список телефонів кафедри АКС </title>
</head>
<body>
<h2> Список телефонів співробітників кафедри АКС </h2>
<h3> Складений для відділу кадрів 10.01.12 </h2>
Список містить <p> ПІБ, дати народження, домашні адреси й телефони <p> всіх співробітників кафедри АКС.
</body>
</html>
```

Екран дисплея (рис. 2.1) буде відображати таке:

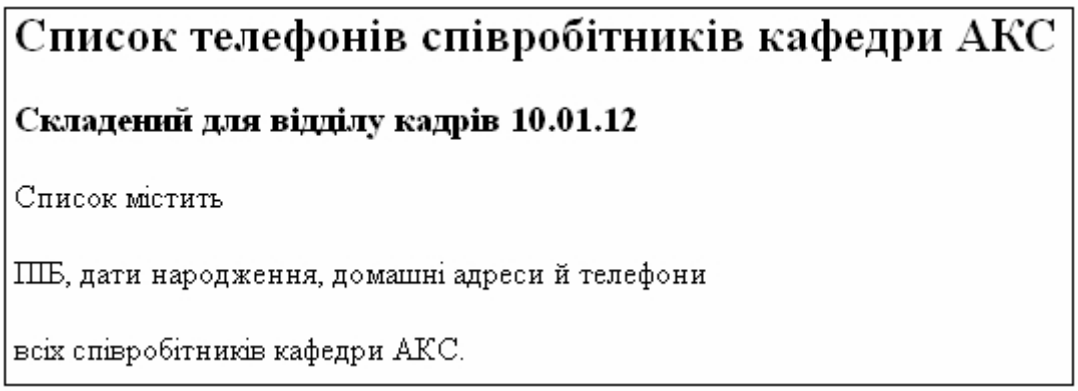


Рис. 2.1. Результат виводу на екран html-документа прикладу 2.1

Для розриву рядка використовується тег <br>. Перепишемо попередній html-документ із використанням цього тегу.

Приклад 2.2

```
<html>
<heard>
<title> Список телефонів кафедри АКС </title>
</head>
<body>
<h2> Список телефонів співробітників кафедри АКС </h2>
<h3> Складений для відділу кадрів 10.01.12 </h2>
Список містить <br> ПІБ, дати народження, домашні адреси й
телефони <br> всіх співробітників кафедри АКС.
</body>
</html>
```

Екран дисплея (рис. 2.2) буде відображати таке:

Рис. 2.2. Результат виводу на екран html-документа прикладу 2.2

Для того щоб неможливо було переносити рядки, використовуються теги <nobr> й </nobr>.

У цьому випадку браузер не буде переносити рядок, навіть якщо він виходить за межі екрана (замість цього браузер дозволить горизонтально прокручувати вікно).

Для позначення в документі цитати з іншого джерела використовується тег **<**blockquote**>**. Текст, що позначається таким тегом, виводиться з відступом 5 пробілів від лівого краю документа.

Перепишемо попередній html-документ із використанням цього тегу.

Приклад 2.3

```
<html>
<heard>
<title> Список телефонів кафедри АКС </title>
</head>
<body>
Список містить <blockquote> ПІБ, дати народження, домашні
адреси й телефони <br>
всіх співробітників кафедри АКС.
</blockquote>
</body>
</html>
```

Екран дисплея (рис. 2.3) буде відображати таке:

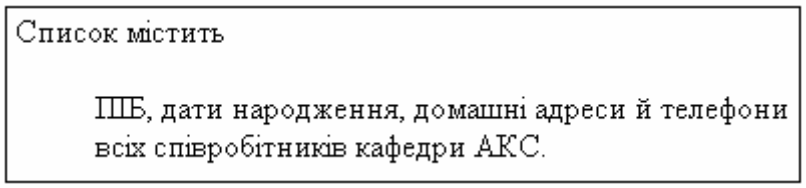


Рис. 2.3. Результат виводу на екран html-документа прикладу 2.3

2.2.2. Спеціальне форматування

Розглянуті далі теги дозволяють створити більш функціональні html-документи.

Тег `<address>` використовується для виділення автора документа і його адреси (наприклад, e-mail). Синтаксис має такий вигляд:

```
<address> Адреса автора </address>
```

Деякі символи є в HTML і не можуть прямо використовуватися в документі:

- ліва кутова дужка "<";
- права кутова дужка ">";
- амперсанд "&";
- подвійні лапки " " ".

Щоб скористатися ними, необхідно замінити їх escape-послідовностями (табл. 2.1).

Таблиця 2.1

Таблиця заміни керуючих символів

Керуючий символ	Використовувані escape-послідовності
<	&lt;
>	&gt;
&	&amp;
“	&quot;

Існує велика кількість escape-послідовностей для позначення спеціальних символів, що з'явилися в HTML 2.0.

Наприклад "©" для позначення знака © і “®” для значка ®.

Одна з особливостей – заміна символів у другій частині символьної таблиці (після 127-го символу) на escape-послідовності для передачі текстових файлів з національними мовами з перекодуванням у 7-бітні коди.

Escape-послідовності чутливі до регістру. Тому не можна використовувати “<” замість “<”.

2.2.3. Форматування списків

Існує чотири основні види списків в HTML-документах:

– непронумеровані;
– пронумеровані;
– вкладені;
– визначень.

Для непронумерованих списків браузер звичайно використовує маркери для позначення елемента списку. Вид маркера, як правило, налаштовує користувач браузера.

Непронумерований список починається стартовим тегом <ul> і завершується тегом . Кожний елемент списку починається з тега <li>.

Приклад 2.4

```
<html>
<heard>
<title> Автори навчального посібника </title>
</head>
<body>
<h2> Автори навчального посібника <br> з програмування </h2>
<ul>
<li> Цвіркун Л.І.
<li> Євстігнєєва А.А.
<li> Панферова Я.В.
</ul>
</body>
</html>
```

Екран дисплея (рис. 2.4) буде відображати таке:

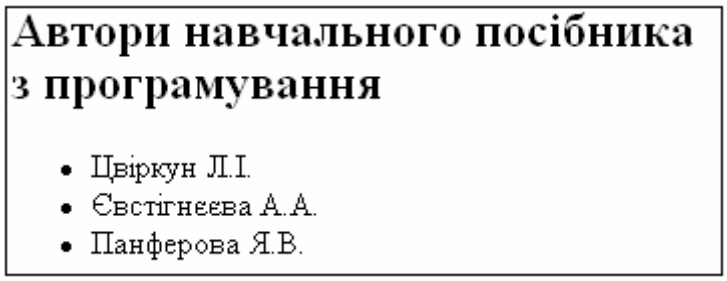


Рис. 2.4. Результат виводу на екран html-документа прикладу 2.4

Тег <ul> може мати параметри:

```
<ul type=disc|circle|square>
```

Тип тегу <ul> обумовлює зовнішній вигляд маркера як вигляд за умовчанням (disc), круглий (circle) або квадратний (square).

Приклад 2.5

```
<html>
<heard>
<title> Автори навчального посібника </title>
</head>
<body>
<h2> Автори навчального посібника <br> з програмування </h2>
<ul type=square>
<li> Цвіркун Л.І.
<li> Євстігнєєва А.А.
<li> Панферова Я.В.
</ul>
</body>
</html>
```

Екран дисплея (рис. 2.5) буде відображати таке:

Автори навчального посібника з програмування

- Цвіркун Л.І.
- Євстігнєєва А.А.
- Панферова Я.В.

Рис. 2.5. Результат виводу на екран html-документа прикладу 2.5

У пронумерованому списку браузер автоматично вставляє номери елементів один за одним.

Це означає, що якщо видалити один або кілька елементів пронумерованого списку, то інші номери автоматично будуть перелічені.

Пронумерований список починається стартовим тегом <ol> і завершується тегом . Кожний елемент списку починається з тегу <li>.

Приклад 2.6

```
<html>
<heard>
<title> Автори навчального посібника </title>
</head>
<body>
<h2> Автори навчального посібника <br> з програмування </h2>
<ol>
<li> Цвіркун Л.І.
<li> Євстігнєєва А.А.
<li> Панферова Я.В.
</ol>
</body>
</html>
```

Екран дисплея (рис. 2.6) буде відображати таке:

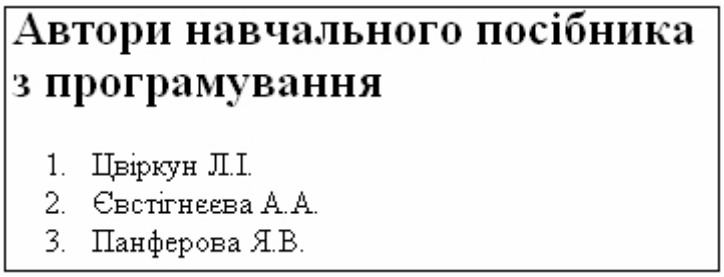


Рис. 2.6. Результат виводу на екран html-документа прикладу 2.6

Тег <ol> може мати такі параметри:

```
<ol type=A|a|I|i|1 start=n>
```

де type – вигляд лічильника:

A – великі латинські літери (A, B, C ...);
a – маленькі латинські літери (a, b, c ...);
I – великі римські цифри (I, II, III ...);
i – маленькі римські цифри (i, ii, iii ...) ;
1 – арабскі цифри (1, 2, 3 ...);

start=n – число, з якого починається відлік.

Приклад 2.7

```
<html>
<heard>
<title> Автори навчального посібника </title>
</head>
<body>
<h2> Автори навчального посібника <br> з програмування </h2>
<ol type=i start=11>
<la> Цвіркун Л.І.
<la> Євстігнєєва А.А.
<la> Панферова Я.В.
</ol>
</body>
</html>
```

Екран дисплея (рис. 2.7) буде відображати таке:

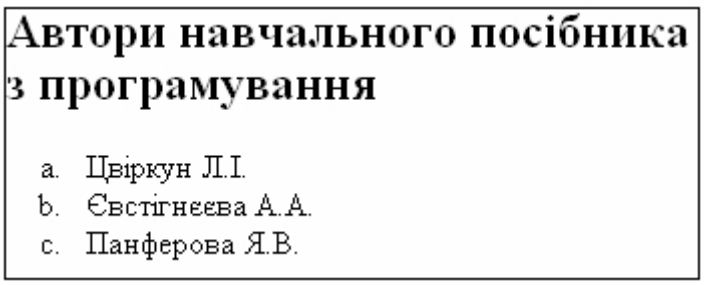


Рис. 2.7. Результат виводу на екран html-документа прикладу 2.7

Вкладені списки створюються, коли використовуються різні теги списків, або коли повторюються одні всередині інших. Для цього просто необхідно розмістити одну пару тегів (стартовий і завершальний) усередині іншої.

Чи будуть елементи вкладеного списку мати ті самі маркери, що позначають елемент списку, залежить від браузера.

Приклад 2.8

```
<html>
<heard>
<title> Список викладачів кафедри АКС </title>
</head>
<body>
<h2> Список викладачів кафедри АКС </h2>
<ul type=square>
<li> Професори
<ol>
<li> Ткачов В.В.
<li> Кожевніков В.Л.
<li> Цвіркун Л.І.
</ol>
<li> Доценти
<ol>
<li> Пушкар М.С.
<li> Ткаченко С.М.
</ol>
<li> Ст. викладачі
<ol>
<li> Надточий В.В.
</ol>
</ul>
<p>
<h3> Складений для відділу кадрів 10.01.12 </h2>
</body>
</html>
```

Екран дисплея (рис. 2.8) буде відображати таке:

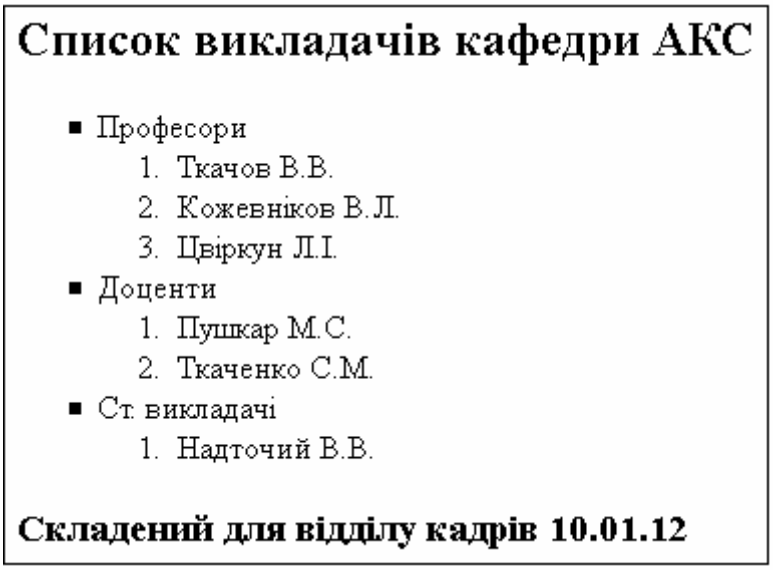


Рис. 2.8. Результат виводу на екран html-документа прикладу 2.8

Тег <li> може мати різні параметри, оскільки це залежить від того, у якому тегу знаходиться даний елемент:

Для <ul> – <li type=disc|circle|square> , а

для <ol> – <li type=A|a|I|i|1 value=n>

Приклад 2.9

```
<html>
<heard>
<title> Список викладачів кафедри АКС </title>
</head>
<body>
<h2> Список викладачів кафедри АКС </h2>
<ul>
<li type=square> Професори
<ol>
<li> Ткачов В.В.
<li> Кожевніков В.Л.
<li> Цвіркун Л.И.
</ol>
<li> Доценти
<ol>
<li type=1 value=4> Кожевніков А. В.
<li> Пушкар М.С.
<li> Ткаченко С.М.
</ol>
<li type=circle> Ст. викладачі
<ol>
<li type=1 value=7> Надточий В.В.
</ol>
</ul>
<h3> Складений для відділу кадрів 10.01.12 </h2>
</body>
</html>
```

Екран дисплея (рис. 2.9) буде відображати таке:

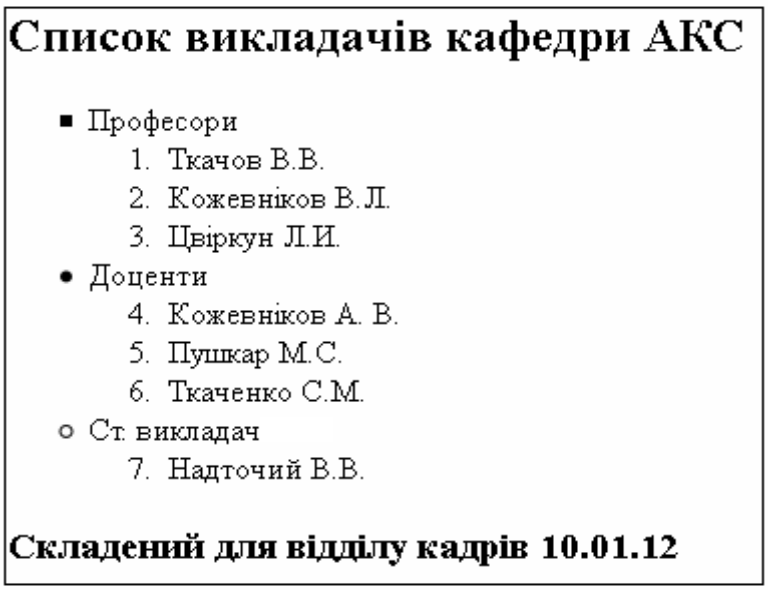


Рис. 2.9. Результат виводу на екран html-документа прикладу 2.9

Список означень починається з тега <dl> і завершується тегом </dl>. Даний список служить для створення списків типу "термін"-"опис". Кожний термін починається тегом <dt> , а опис – тегом <dd>.

Приклад 2.10

```
<html>
<heard>
<title> Мехатроніка </title>
</head>
<body>
<dl>
<dt> <b> Мехатронний об'єкт </b>
<dd> Являє собою інтеграцію механічних, електротехнічних,
     <br> мікроелектронних і комп'ютерних компонентів
<dt> <b> Мехатронний модуль </b>
<dd> Призначений, як правило, для реалізації рухів за однією
     координатою
<dt> <b> Мехатронний агрегат </b>
<dd> Включає декілька модулів і призначений для реалізації
     <br> заданих законів руху за декількома координатами в
     <br> умовах взаємодії із зовнішнім середовищем
<dt> <b> Мехатронний комплекс </b>
<dd> Складається з декількох агрегатів або агрегату і ряду
     <br> окремих модулів, зв'язаних між собою і діючих як
     одне ціле
<dl>
</body>
</html>
```

Екран дисплея (рис. 2.10) буде відображати таке:

Мехатронний об'єкт
Являє собою інтеграцію механічних, електротехнічних, мікроелектронних і комп'ютерних компонентів
Мехатронний модуль
Призначений, як правило, для реалізації рухів за однією координатою
Мехатронний агрегат
Включає декілька модулів і призначений для реалізації заданих законів руху за декількома координатами в умовах взаємодії із зовнішнім середовищем
Мехатронний комплекс
Складається з декількох агрегатів або агрегату і ряду окремих модулів, зв'язаних між собою і діючих як одне ціле

Рис. 2.10. Результат виводу на екран html-документа прикладу 2.10

2.2.4. Форматування таблиць

Таблиці в HTML-документах організуються як набір стовпців і рядків. Комірки таблиці можуть містити будь-які html-елементи. Наприклад, заголовки, списки, абзаци, фігури, графіки, а також елементи форм.

Теги <table>...</table> є основними для опису таблиці. Інші елементи (не основні) повинні знаходитися всередині цих двох тегів.

За умовчанням таблиця не має обрамлення й роздільників. Обрамлення додається атрибутом border.

Кількість рядків таблиці визначається кількістю пар, що зустрічаються, тегів <tr> ...</tr>. Рядки можуть мати атрибути `align` й `valign`, які описують візуальне розташування вмісту рядків у таблиці.

Для опису стандартної комірки таблиці використовуються теги <td>...</td>.

Комірка таблиці може бути описана тільки всередині рядка таблиці. Кожна комірка повинна мати номер колонки, для якої вона описується. Якщо в рядку не має одної або кілька комірок для деяких колонок, то браузер відображає порожню комірку.

Розташування даних у комірці за умовчанням визначається атрибутами align=left й valign=middle. Таке розташування може бути виправлене як на рівні опису рядка, так і на рівні опису комірки.

Для опису заголовка таблиці використовуються теги <th>...</th>.

Комірка заголовка таблиці має ширину всієї таблиці, текст у даній комірці має атрибут bold й align=center.

Назва таблиці описується тегами <caption>...</caption> .

Тег <caption> повинен бути присутнім усередині <table>...</table>, але зовні опису якого-небудь рядка або комірки. За умовчанням <caption> має атрибут align=top, але він може бути явно встановлений в align=bottom.

Атрибут align визначає, у якому місці (зверху або знизу) таблиці буде поставлена назва. Назва завжди центрована в рамках ширини таблиці.

Таблиця може мати атрибути, наведені в табл. 2.2.

Таблиця 2.2

Основні атрибути таблиці

Ім'я атрибута	Опис атрибута
`border`	Даний атрибут використовується в тегі `table`. Якщо даний атрибут присутній, межа таблиці прорисовується для всіх комірок і для таблиці в цілому. Атрибут border може набувати числове значення, що визначає ширину межі, наприклад `border=3`
`align`	Якщо атрибут `align` є всередині тегів `<caption>` і `</caption>`, то він визначає положення назви таблиці (зверху або знизу). Якщо не призначено інше, то `align=top`

Продовження табл. 2.2

Ім'я атрибута	Опис атрибута
	Якщо атрибут `align` зустрічається всередині `<tr>`, `<th>` або `<td>`, він керує положенням даних у комірках по горизонталі. Може набувати значення `left` (ліворуч), `right` (праворуч) або `center` (по центру)
`valign`	Даний атрибут зустрічається всередині тегів `<tr>`, `<th>` й `<td>`. Він визначає вертикальне розміщення даних у комірках. Може набувати значення `top` (угорі), `bottom` (унизу), `middle` (посередині) і `baseline` (всі комірки рядка притиснуті догори)
`nowrap`	Даний атрибут визначає те, що дані в комірці не можуть логічно розбиватися на кілька рядків і повинні бути подані одним рядком
`colspan`	Даний атрибут визначає кількість комірок, які будуть об'єднані по горизонталі для зазначеної комірки. Якщо не призначено інше, то `colspan`=1
`rowspan`	Даний атрибут визначає кількість комірок, які будуть об'єднані по вертикалі для зазначеної комірки. Якщо не призначено інше, то `rowspan`=1
`colspec`	Даний параметр дозволяє задавати фіксовану ширину колонок у символах або відсотках, наприклад `colspec="20%"`

Приклад 2.11

```
<html>
<heard>
<title> Приклад таблиці </title>
</head>
<body>
<table border=2>
<caption> Правила дій з числами </caption>
<tr> <th rowspan=2>1-ше число</th> <th rowspan=2> 2-ге число
</th> <th colspan=2>Результат додавання </th></tr>
<tr><td>Значення і-го разряду</td><td>Перенесення</td></tr>
<tr><td align=center>0</td><td align=center>0</td><td
align=center>0</td><td align=center >0</td></tr>
<tr><td align=center>0</td><td align=center>1</td><td
align=center>1</td><td align=center >0</td></tr>
<tr><td align=center>1</td><td align=center>0</td><td
align=center>1</td><td align=center>0</td></tr>
<tr><td align=center>1</td><td align=center>1</td><td
align=center>0</td><td align=center>1</td></tr>
</table>
</body>
</html>
```

Екран дисплея (рис. 2.11) буде відображати таке:

Правила дій з числами

1-ше число	2-ге число	Результат додавання	
		Значення і-го разряду	Перенесення
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Рис. 2.11. Результат виводу на екран html-документа прикладу 2.11

2.3. Використання стилів в html-документі

HTML дозволяє використовувати різні стилі шрифтів для виділення текстової інформації в документах. Більшістю браузерів підтримують такі стилі:

– bold (жирний);

– italic (похилий);

– mono spaced (type writer – з використанням фіксованих шрифтів).

Можливо комбінувати різні види стилів, наприклад, жирний і похилий.

У табл. 2.3 указані теги, які потрібно використовувати для вибору певного стилю.

Таблиця 2.3

Теги для різних стилів

Позначення стилю	Назва стилю	Теги
Bold	Жирний	…
Italic	Похилий	…
Monospaced	З непропорційним шрифтом	<tt> …</tt>
Big	Великий	<big> …</big>
Small	Маленький	…
Sub	Підрядковий	_…
Sup	Надрядковий	[…]

За допомогою тегів й можна змінювати розмір шрифту.

Тег < font size > може мати такі параметри:

```
<font size=+|- n>
```

Шрифт може мати розмір від 1 до 7. Можна безпосередньо вказати розмір шрифту цифрою або вказати зсув відносно базового значення (якщо не призначено інше, то дорівнює 3) у позитивну або негативну сторону.

Базове значення можна змінити за допомогою тегу:

```
<basefont size=n>
```

Використаємо розглянуті теги.

Приклад 2.12

```
<html>
<heard>
<title> Приклад зі зміною розміру шрифту </title>
</head>
<body>
<p>Базовий розмір шрифту,якщо не призначено інше, то дорівнює
3 <p>
Установлення розміру шрифту від 1 до 7 і від 7 до 1 <br>
зсувом відносно базового:<p>
<font size=-2>1</font><font size=-1>2</font>3<font
size=+1>4</font><font size=+2>5</font><font size=+3>6</font><font
size=+4>7</font>
<font size=+3>6</font><font size=+2>5</font><font
size=+1>4</font>3<font size=-1>2</font><font size=-2>1</font><p>
<basefont size=5>Установлення базового розміру<br>шрифту
дорівнює 5 </font>
</body>
</html>
```

Екран дисплея (рис. 2.12) буде відображати таке:

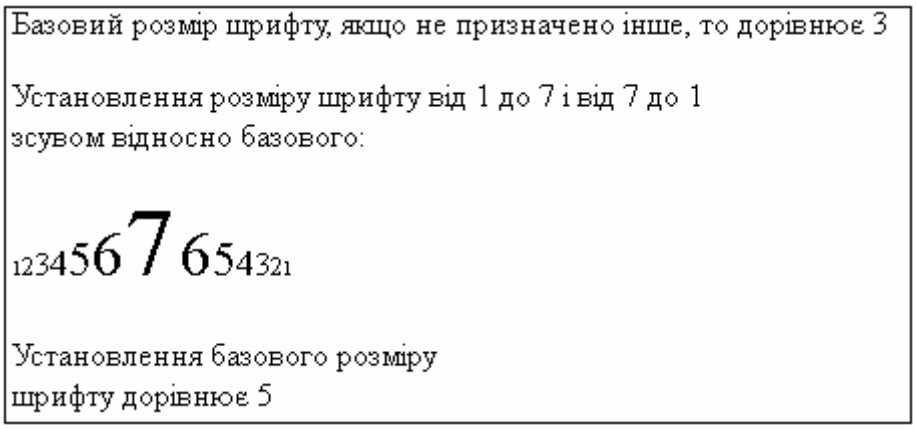


Рис. 2.12. Результат виводу на екран html-документа прикладу 2.12

За допомогою тегів <font color> й можна змінити колір шрифту.

Тег <font color> має такий вигляд:

```
<font color="#xxxxxx">
```

Колір вказується в RGB-форматі (Red-Green-Blue) із зазначенням розмірності кожного компонента кольору в шістнадцатирічному форматі. В html-документах кольори позначаються цифрами в шістнадцатирічному коді.

Колірна система базується на трьох основних кольорах – червоному, зеленому й синьому. Для кожного кольору задається значення від 00 до FF. Потім ці значення об'єднуються в одне число, перед яким ставиться символ #.

Для простоти в HTML визначено 16 стандартних кольорів. Табл. 2.4 демонструє ці кольори і їх шістнадцатирічний код.

Таблиця 2.4

Коди кольорів, які використовуваються в html-документах

Колір	Код кольору	Колір	Код кольору
black	#000000	green	#008000
silver	#C0C0C0	lime	#00FF00
gray	#808080"	olive	#808000
white	#FFFFFF	yellow	#FFFF00
maroon	#800000	navy	#000080
red	#FF0000	blue	#0000FF
purple	#800080	teal	#008080
fuchsia	#FF00FF	aqua	#00FFFF

2.4. Гіпертекстові посилання

Гіпертекстові посилання – ключові компоненти, що надають веб-документам привабливості для користувачів. Такі посилання роблять набір документів зв'язаним і структурованим, що дозволяє користувачеві отримувати необхідну інформацію максимально швидко й зручно.

Посилання мають стандартний формат, що дозволяє браузеру інтерпретувати їх і виконувати необхідні функції (викликати методи) залежно від типу посилання. Посилання можуть вказувати на інший документ, спеціальне місце даного документа або виконувати інші функції. Наприклад, запитувати файл по FTP-протоколу для відображення його браузером.

HTML використовує URL (Uniform Resource Locator) для подання гіпертекстових посилань і посилань на мережні сервіси всередині html-документа.

URL може вказувати на спеціальне місце в абсолютному шляху доступа, або вказувати на документ у поточному шляху, що часто використовується при організації великих структурованих веб-сайтів.

Перша частина URL (до двокрапки) описує метод доступу або мережний сервіс. Інша частина URL (після двокрапки) інтерпретується залежно від методу доступу. Звичайно, два прямі слеша після двокрапки позначають ім'я машини:

```
метод://ім'я_машини/шлях/ім'я_файлу
```

Наприклад, для методу доступу html і виклика документа index.html із сервера nmu.org.ua необхідно вказати такий рядок:

```
http://nmu.org.ua/index.html
```

Загальний формат URL має вигляд:

```
метод://ім'я_сервера:порт/шлях/ім'я_файлу#рядок документа.
```

Опис компонент URL поданий в табл. 2.5.

Таблиця 2.5

Компоненти Uniform Resource Locator

Ім'я компоненти	Що являє собою компонента URL	Примітка
Метод	Ім'я операції, що буде виконуватися при інтерпретації даного URL	–
Ім'я_сервера	Необов'язковий параметр, що описує повне мережне ім'я машини	Якщо ім'я сервера не указане, то посилання вважається локальним, повний шлях, зазначений далі в URL, обчислюється на тій машині, з якої взятий html-документ, що містить дане посилання. Замість символьного імені машини може бути використана IP-адреса (не рекомендується через можливе перетинання з фіксованими локальними адресами внутрішньої мережі)
Порт	Номер порту TCP, на якому функціонує веб-сервер	Номер порту TCP, на якому функціонує веб-сервер. Якщо порт не зазначений, то використовується порт 80. Даний параметр (порт) не використовується в переважній більшості URL
Шлях	Частковий або повний шлях до документа, що повинен викликатися в результаті інтерпретації URL	Веб-сервери сконфігуровані по-різному для інтерпретації шляху доступу до документа. Якщо використовуються виконавчі програми, то вони звичайно збираються в одному або декількох виділених каталогах

Продовження табл. 2.5

Ім'я компоненти	Що являє собою компонента URL	Примітка
		Шлях до цих каталогів записується в спеціальних параметрах веб-сервера. Для даних каталогів веб-сервером виділяється спеціальний логічний шлях, що і використовується в URL. Якщо веб-сервер бачить даний шлях, то запитуваний файл інтерпретується як виконувана програма. У противному разі запитуваний файл інтерпретується просто як файл даних, навіть якщо він є виконуваною програмою
Ім'я_файлу	Ім'я викликуваного документа	Якщо після мережного імені машини відразу йде ім'я документа, то він повинен знаходитися в кореневому каталозі на віддаленій машині або (що частіше) у каталозі, виділеному веб-сервером як кореневий. Якщо ж URL закінчується мережним ім'ям машини, то як документ запитується документ із кореневого каталогу віддаленої машини з ім'ям, установленим у налаштуваннях веб-сервера (як правило, це `index.html`)
Рядок документа	Даний елемент є посиланням на рядок (точку) усередині html-документа	Більшість браузерів, зустрічаючи після імені документа даний елемент, розміщають документ на екрані таким чином, що указаний рядок документа поміщається у верхній рядок робочого вікна браузера. Точки, на які посилається `#рядок документа`, вказуються в документі за допомогою тега `name`, як це буде описано далі

Можливі варіанти компоненти URL “метод” подані в табл. 2.6.

Таблиця 2.6

Опис можливих варіантів компоненти URL “метод”

Можливі варіанти компоненти URL “метод”	Опис
file	Читання файлу з локального диска. Ім'я файлу інтерпретується для локальної машини користувача. Даний метод використовується для відображення якого-небудь файлу, що знаходиться на машині користувача. Наприклад: `file:/home/alex/index.html` – відображає файл `index.html` з каталогу `/home/alex` на комп’ютері користувача
http	Доступ до веб-сторінки в мережі з використанням http-протоколу. Це найбільше часто використовуваний метод доступу до якого-небудь html-документа в мережі. Наприклад: `http://www.nmu.org.ua/` – доступ до home-сторінки Державного ВНЗ “Національний гірничий університет”
ftp	Запит файлу з анонімного ftp-сервера. Наприклад: `ftp://hostname/directory/filename`
mailto	Активізує поштову сесію із указаним користувачем і сервером. Наприклад: mailto:TsvirkunL@gmail.com – активізує сесію посилки повідомлення користувачеві `TsvirkunL` на сервері `gmail.com`, якщо браузер підтримує запуск електронної пошти. При цьому метод `mailto` не вимагає зазначення слешей після двокрапки (як правило, після двокрапки вказується електронна адреса користувача)
telnet	Звертання до служби `telnet`

Для того, щоб браузер відобразив посилання на URL, необхідно відзначити URL тегами <A HREF="URL"> й </A> в html-документі.

Текст, що знаходиться між даними двома тегами, позначається спеціальним способом веб-браузером.

Звичайно цей текст відображається підкресленим і виділеним синім (або іншим заданим користувачем) кольором. Текст, що позначає URL, не відображається браузером, а використовується тільки для виконання запропонованих ним дій при активізації посилання.

Приклад 2.13

```
<html>
<heard>
<title> Приклад 2.13 </title>
</head>
<body>
```

```
Інформацію про Академію Cisco НГУ можна одержати <br>
на сайті Державного ВНЗ <A HREF=http://cisco.nmu.org.ua/>
“Національний гірничий університет”</A>
</body>
</html>
```

Екран дисплея (рис. 2.13) буде відображати таке:

Інформацію про Академію Cisco НГУ можна одержати
на сайті Державного ВНЗ “Національний гірничий університет”

Рис. 2.13. Результат виводу на екран html-документа прикладу 2.13

Можливо зробити посилання на різні ділянки або розділи того самого документа, використовуючи спеціальний прихований маркер для цих розділів. Це дозволяє швидко переходити від розділу до розділу всередині документа, не використовуючи скролінг екрана.

Для створення такого посилання необхідно виконати таке:

– створити маркер розділу. Синтаксис даного маркера має вигляд:

```
<A NAME=”ім'я_рядка”>
Текст,_який_відобразиться_в_першому_рядку_браузера </A>
```

– створити посилання на даний маркер:

```
<A HREF=”#им’я_рядка”> Текст"</A>
```

Приклад 2.14

```
<html>
<heard>
<title> Приклад 2.14 </title>
</head>
<body>
<p><b>Зміст</b>
<ul>
<li><a href="#r1">Розділ 1</a></li>
<li><a href="#r2">Розділ 2</a></li>
</ul>
<p><a name="r1"><b>Розділ 1</b></a>
 <ul>Текст розділу 1</ul>
<p><a name="r2"><b>Розділ 2</b></a>
 <ul>Текст розділу 2</ul>
</body>
</html>
```

Екран дисплея (рис. 2.14) буде відображати таке:

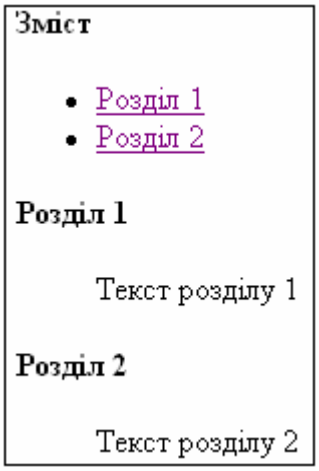


Рис. 2.14. Результат виводу на екран html-документа прикладу 2.14

Символи "#r1" повідомляють браузеру, що необхідно знайти в даному html-документі маркер з ім'ям "r1".

Коли користувач клацне клавішею миші на рядку "Розділ 1", браузер перейде відразу до розділу 1.

Якщо маркер розділу поставити в іншому документі, а в позначці рядка вказати ім'я цього документа (наприклад "2_16.html#r1"), то браузер здійснить загрузку іншого документа й перейде до вказаного для нього розділу.

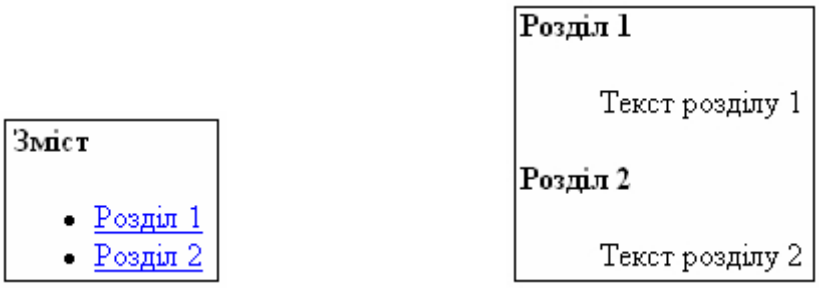
Приклад 2.15

```
<html>
<heard>
<title> Приклад 2.15 </title>
</head>
<body>
<p><b>Зміст</b>
<ul>
<li><a href="2_16.html#r1">Розділ 1</a></li>
<li><a href="2_16.html#r2">Розділ 2</a></li>
</ul>
</body>
</html>
```

Приклад 2.16

```
<html>
<heard>
<title> Приклад 2.16 </title>
</head>
<body>
<p><a name="r1"><b>Розділ 1</b></a>
<ul> Текст розділу 1 </ul>
<p><a name="r2"><b>Розділ 2</b></a>
<ul> Текст розділу 2 </ul>
</body>
</html>
```

Екран дисплея (рис. 2.15) буде відображати таке:



а) б)

Рис. 2.15. Результат виводу на екран прикладів html-документів:
а – вихідна сторінка; б – після переходу по маркеру розділу

Висновки

У даному розділі розглянуто такі основні питання:
- основи синтаксису мови PHP;
- структура програми мовою HTML;
- теги форматування документів;
- стилі html-документів:
- особливості гіпертекстових посилань.

Контрольні питання

1. Що являє собою HTML, документ або програму?
2. Що таке тег мови HTML?
3. Який тег застосовується в HTML-програмі першим?
4. Які типи тегів відомі для форматування текстів?
5. Які теги застосовуються для спеціального форматування?
6. Які види списків існують в HTML-документах?
7. Які html-елементи можуть містити комірки таблиць?
8. Яким атрибутом додається обрамлення до таблиць в HTML-документах?
9. Які стилі шрифтів застосовуються в HTML-документах?
10. Що застосовується в HTML для гіпертекстових посилань?

ЧАСТИНА 2. ОСНОВИ ПРОГРАМУВАННЯ МОВОЮ PHP
3. СЕРЕДОВИЩЕ ПРОГРАМУВАННЯ МОВОЮ PHP

Навчальною метою розділу є ознайомлення студентів із серверним програмним забезпеченням, необхідним для побудови PHP-програм.

У результаті вивчення даного розділу студенти повинні знати:

- визначення веб-сервера;
- визначення інтерпретатора PHP;
- принципи установлення серверного програмного забезпечення для ОС Windows та Linux;
- базові налаштування веб-сервера Apache;
- механізм написання і запуску PHP-програм.

Можливості мови були обговорені, сфери застосування розглянуті, історія створення вивчена. Тепер можна приступати до установлення необхідних програм. Починати треба з веб-сервера та інтерпретатора PHP. Виберемо, наприклад, Apache, як найбільш популярний серед програмістів веб-сервер. Для перегляду результатів роботи програм нам знадобиться веб-браузер, наприклад Firefox.

3.1. Установлення веб-сервера Apache

Для того, щоб встановити веб-сервер Apache на персональний комп'ютер, скопіюємо відповідне програмне забезпечення (ПЗ). Програму для установлення Apache можна знайти на офіційному сайті http://www.apache.org. Наприклад, файл httpd-2.2.22-win32-x86 apache_1.3.29-win3x86-no_src.exe – це автоматичний установник (інакше – wizard) веб-сервера Apache версії 1.3.29 під операційну систему Windows XP. Цей файл допоможе встановити на комп'ютер програмне забезпечення сервера. Після запуску файлу установника на екрані монітора з'явиться повідомлення, рис. 3.1.

Для продовження установлення веб-сервера Apache версії 1.3.29 на комп'ютері потрібно клацнути лівою клавішею миші при встановленому курсорі на кнопці Next (далі писатимемо коротше, тобто просто “натиснути на кнопку Next”).

Ця ж програма дозволяє змінити або видалити раніше встановлений веб-сервер Apache.

Після натиснення на кнопку Next програма запропонує погодитися з умовами ліцензії (рис. 3.2).

Наступний крок − введення ім'я мережного домену, ім'я сервера та e-mail адміністратора. Програма спробує автоматично визначити домен і сервер згідно з налаштуванням комп'ютера (рис. 3.3).

Після того як дані введені в зазначену форму, потрібно вибрати тип установлення (рис. 3.4), повний (встановлюються всі компоненти сервера) або визначуваний користувачем (можна вибрати компоненти для установлення).

Рекомендується вказати тип повного установлення.

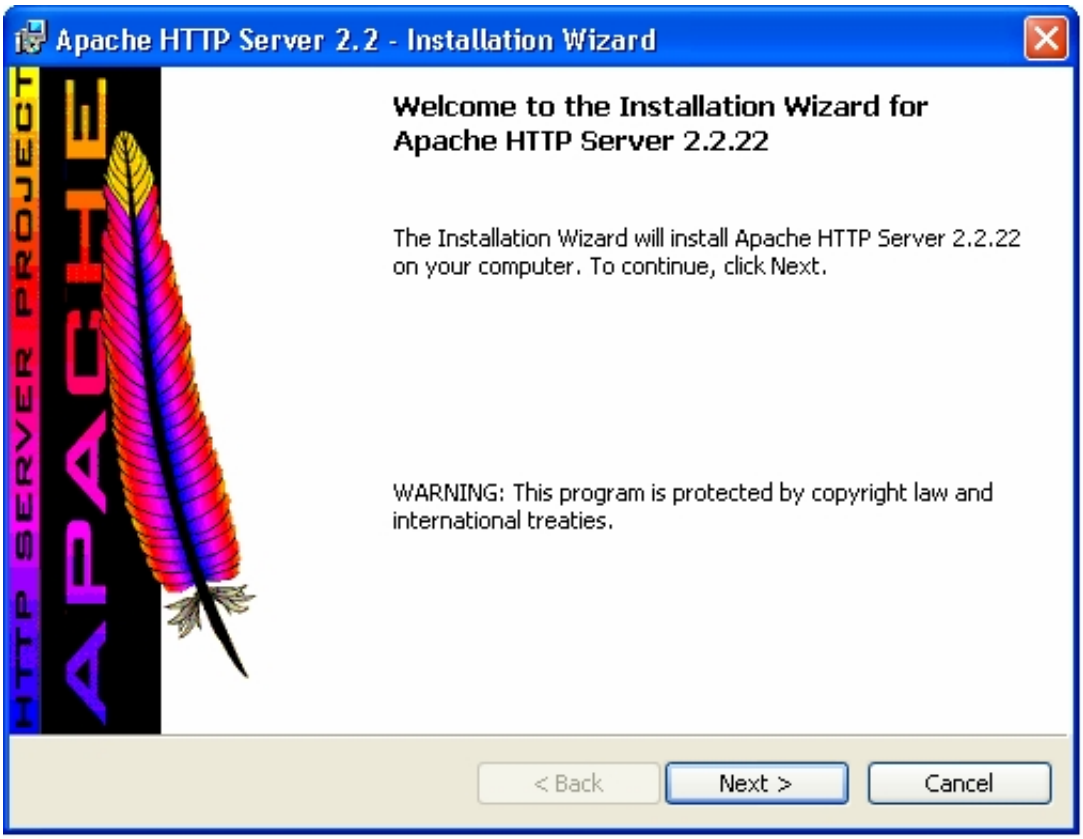


Рис. 3.1. Перше повідомлення автоматичної установки сервера Apache

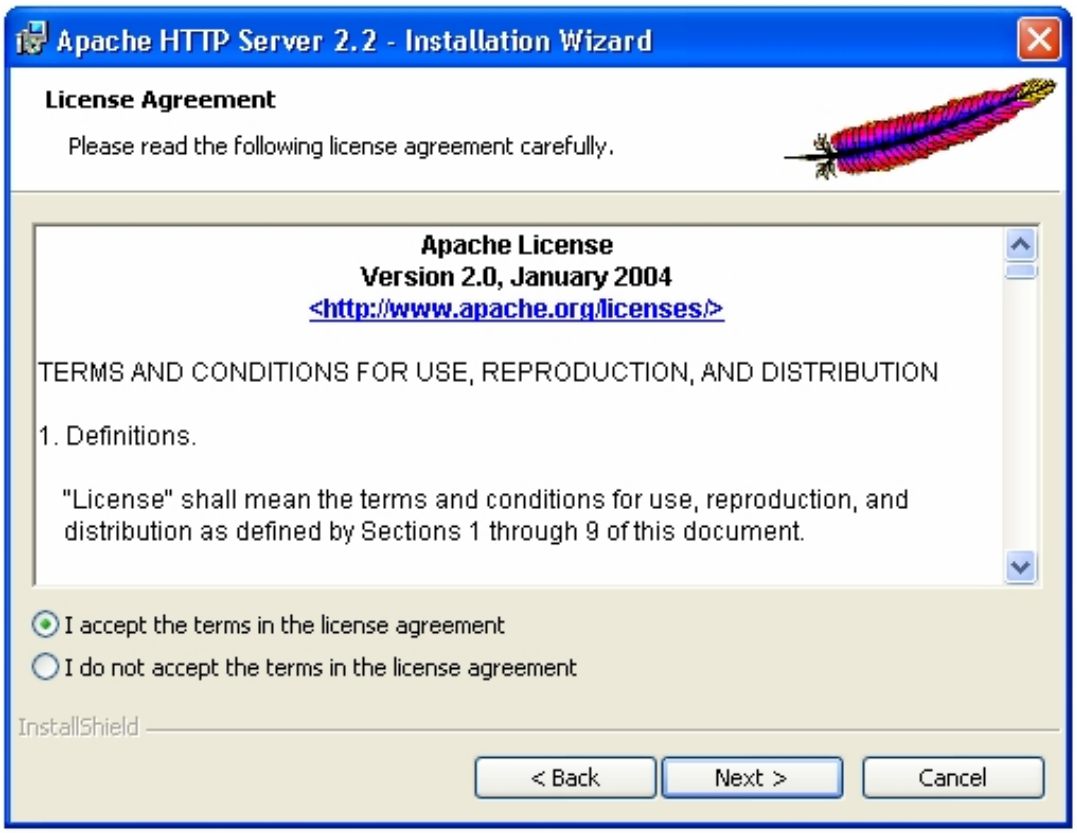


Рис. 3.2. Ліцензійна угода

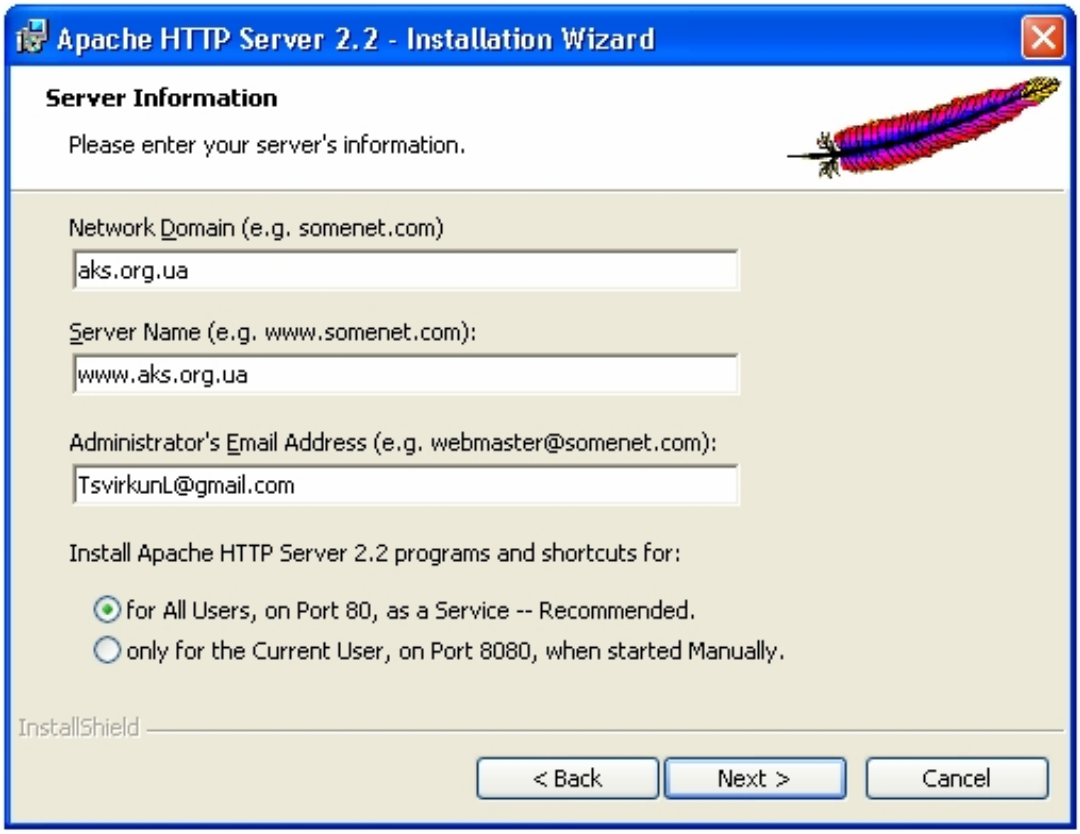


Рис. 3.3. Основна інформація про сервер

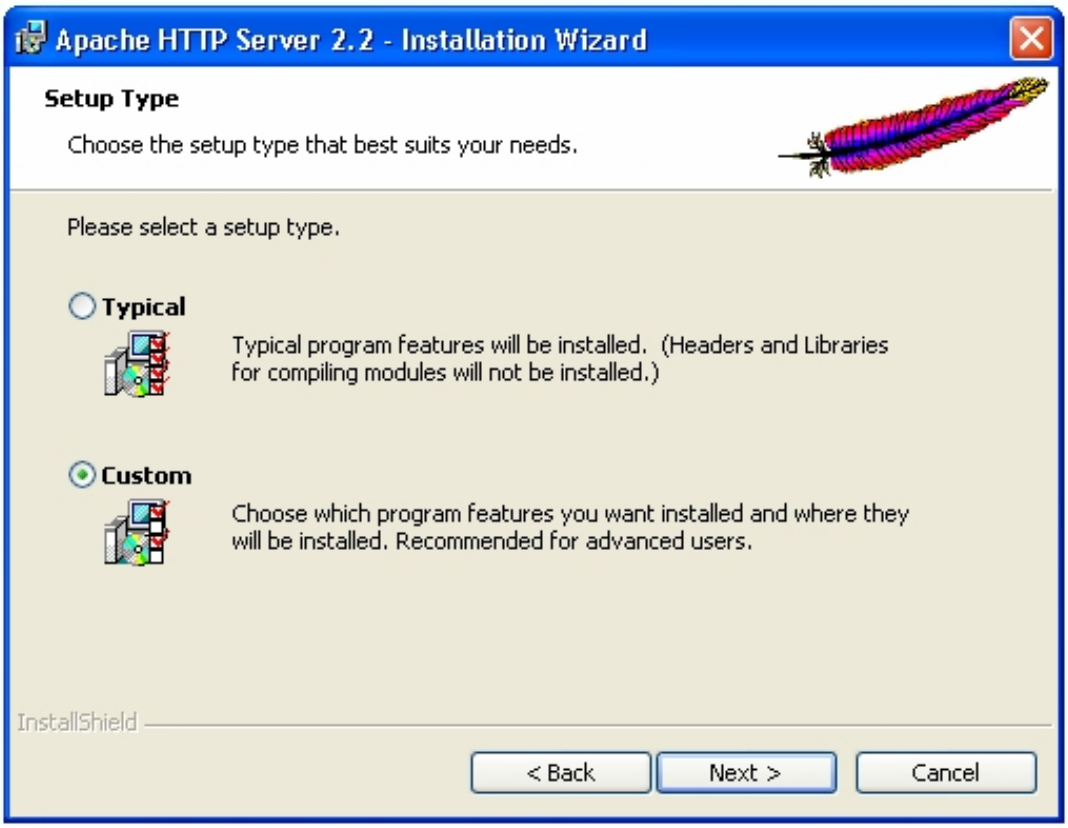


Рис. 3.4. Тип установлення

Тоді на наступному кроці програма запропонує вибрати компоненти сервера і папку, в яку буде встановлений сервер (рис. 3.5).

Якщо не вказано інше, сервер встановлюється в каталог C:\Program Files\Apache Software Foundation\Apache2.2\.

Вибір компонент сервера не дуже великий – це інструменти, необхідні для роботи сервера, і документація до нього (рис. 3.5).

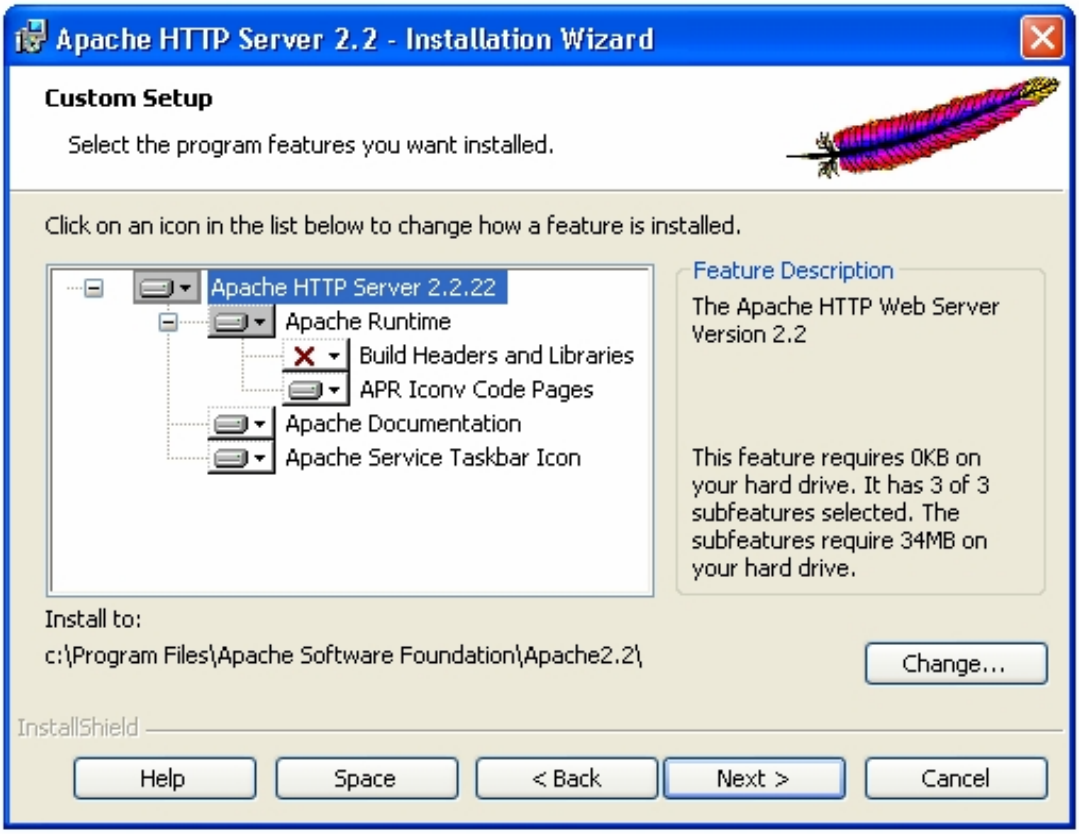


Рис. 3.5. Вибір компонент призначений для установки користувачем

Тепер потрібно підтвердити правильність введених даних (рис. 3.6). Далі натисканням на кнопки Install можна розпочати безпосереднє установлення сервера. Від користувача ніяких додаткових дій більше не вимагається.

По закінченні на екрані з`явиться повідомлення – рис. 3.7.

На будь-якому кроці установлення, включаючи і цей, можна повернутися назад і змінити введені раніше дані.

Файли, які повинні бути оброблені сервером, можна зберігати або в корені сервера, або в директоріях користувачів. Місцезнаходження кореневого каталогу сервера і директорій користувачів записане в налаштуваннях сервера, а точніше – у файлі конфігурації httpd.conf (знайти його можна в C:\Program Files\Apache Software Foundation\Apache2.2\conf).

Щоб змінити ці дані, потрібно відредагувати відповідні рядки у файлі конфігурації сервера.

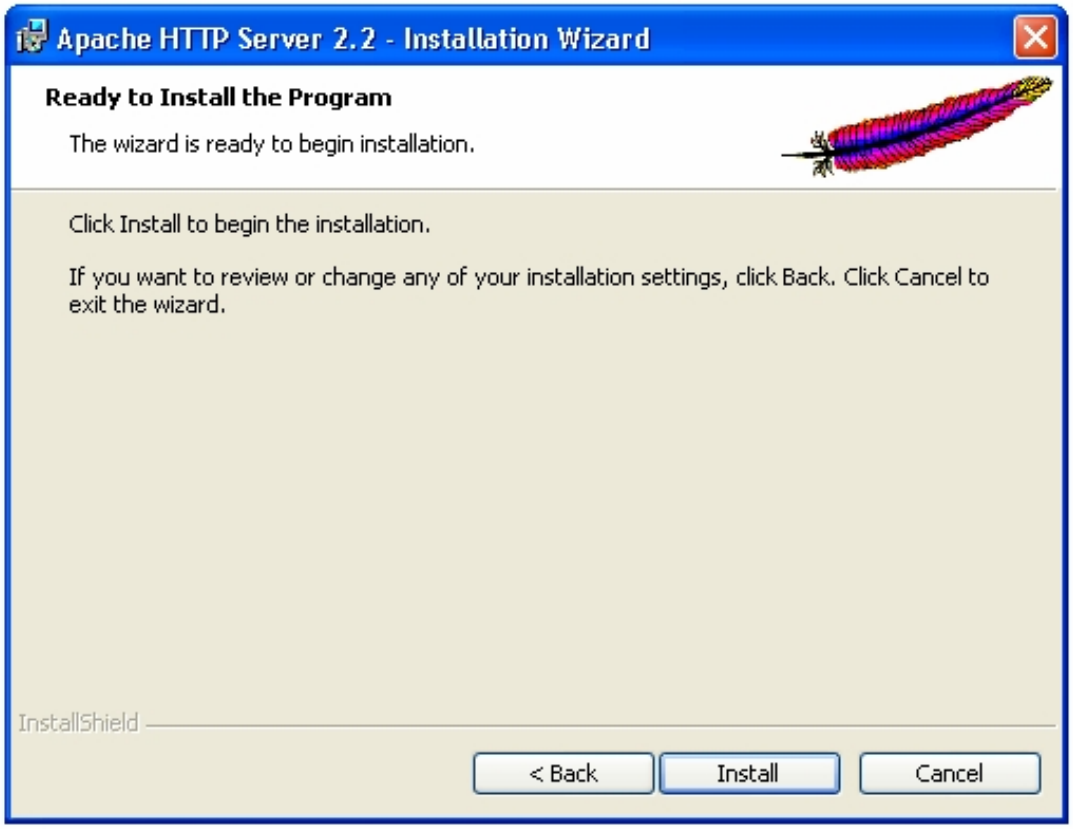


Рис. 3.6. Початок установлення

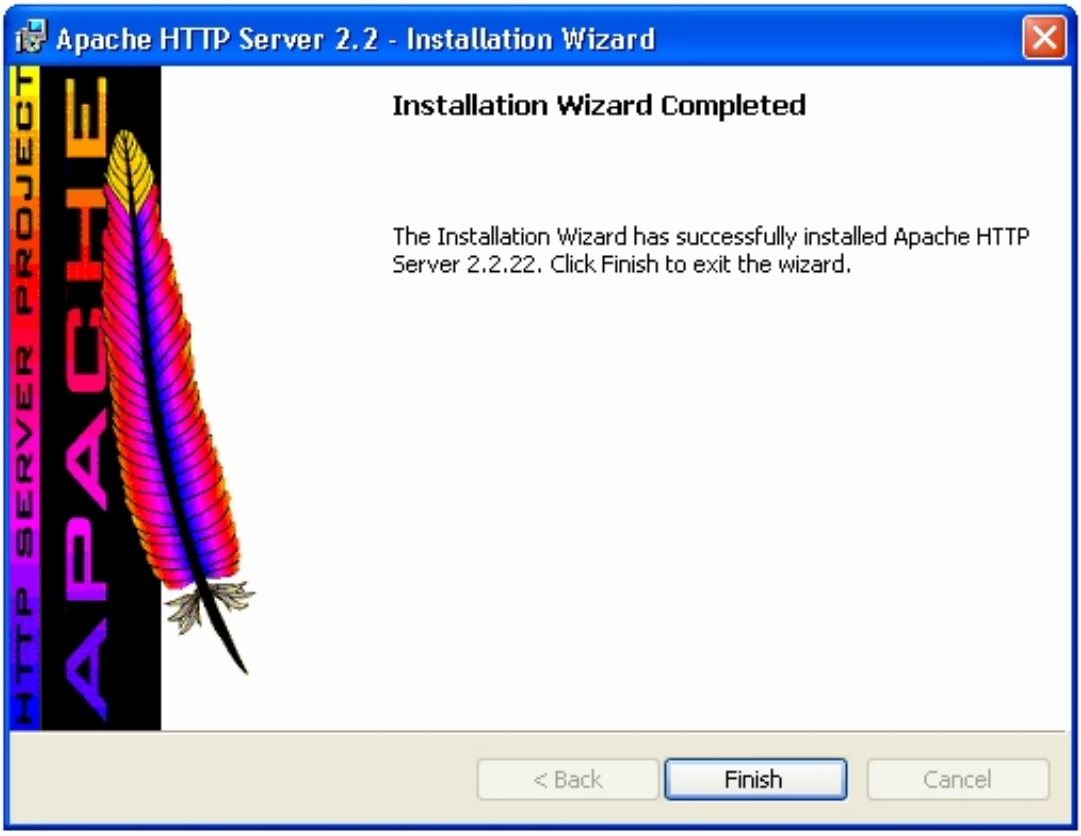


Рис. 3.7. Закінчення установлення

3.2. Установлення програми PHP

3.2.1. Для ОС Windows XP

Необхідно скопіювати дистрибутив програми PHP 5.3.10 з офіційного сайту – http://www.php.net. Далі вибрати автоматичне установлення, як і у випадку з установленням веб-сервера. Спочатку з'явиться вітання – рис. 3.9.

Рис. 3.9. Програма автоматичного установлення PHP

Далі на екрані буде попередження про існування авторських прав на цей продукт і текст ліцензії, яку необхідно прочитати і погодитися (рис. 3.10).

На наступному кроці програма запропонує вибрати директорію для установлення програми PHP, куди будуть скопійовані файли бібліотек, розширень, модулі тощо (рис. 3.11).

Далі слід вибрати із списку сервер, з яким працюватиме PHP. Оскільки раніше був вибраний і встановлений веб-сервер Apache, то необхідно вказати із списку саме його (рис. 3.12).

Потім програма запропонує вибрати каталог для конфігураційних файлів (рис. 3.13) і компоненти програми PHP (рис. 3.14).

Після натиснення кнопки Install (рис. 3.15) почнеться установлення програми PHP (рис. 3.16).

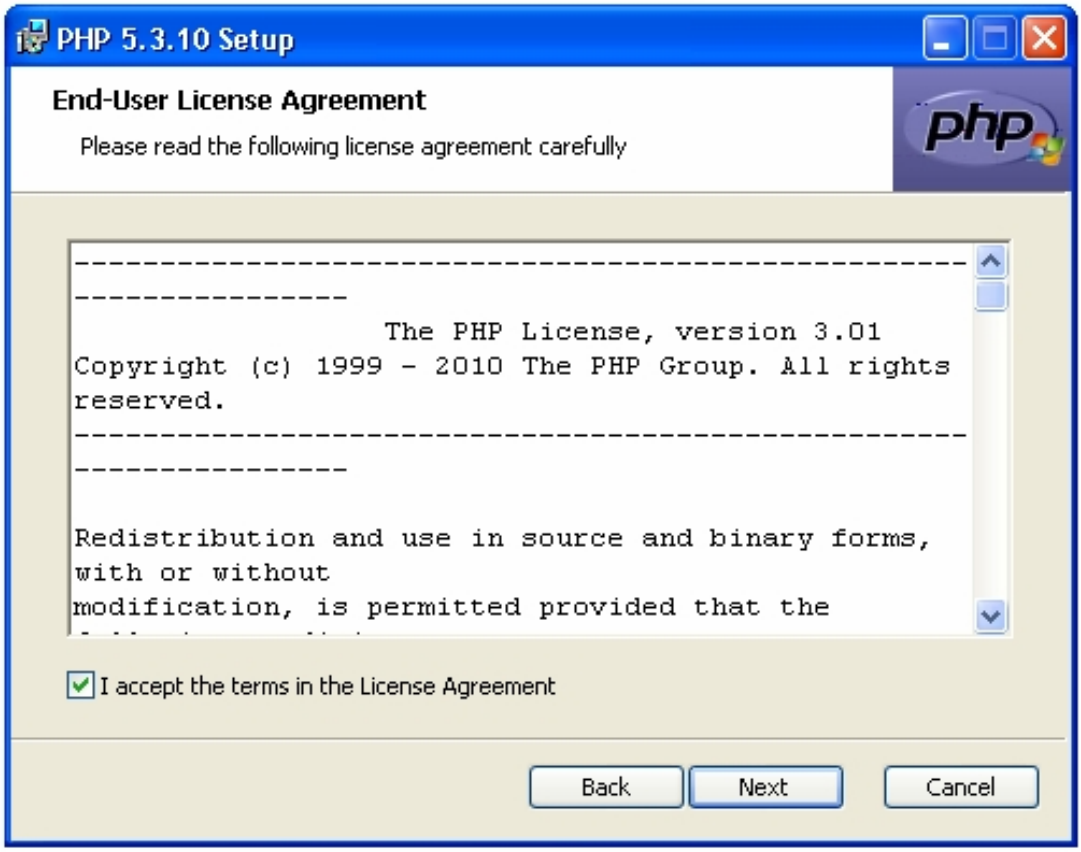


Рис. 3.10. Ліцензійна угода

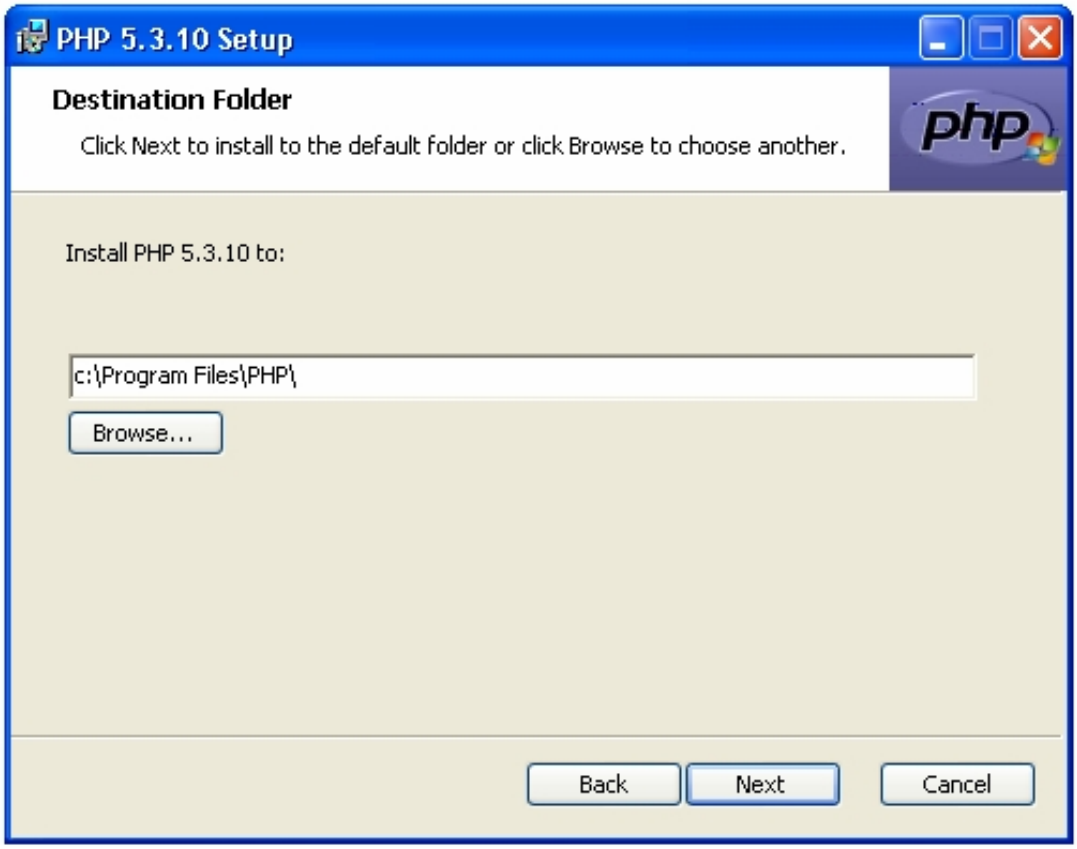


Рис. 3.11. Вибір каталогу, в який буде встановлений PHP

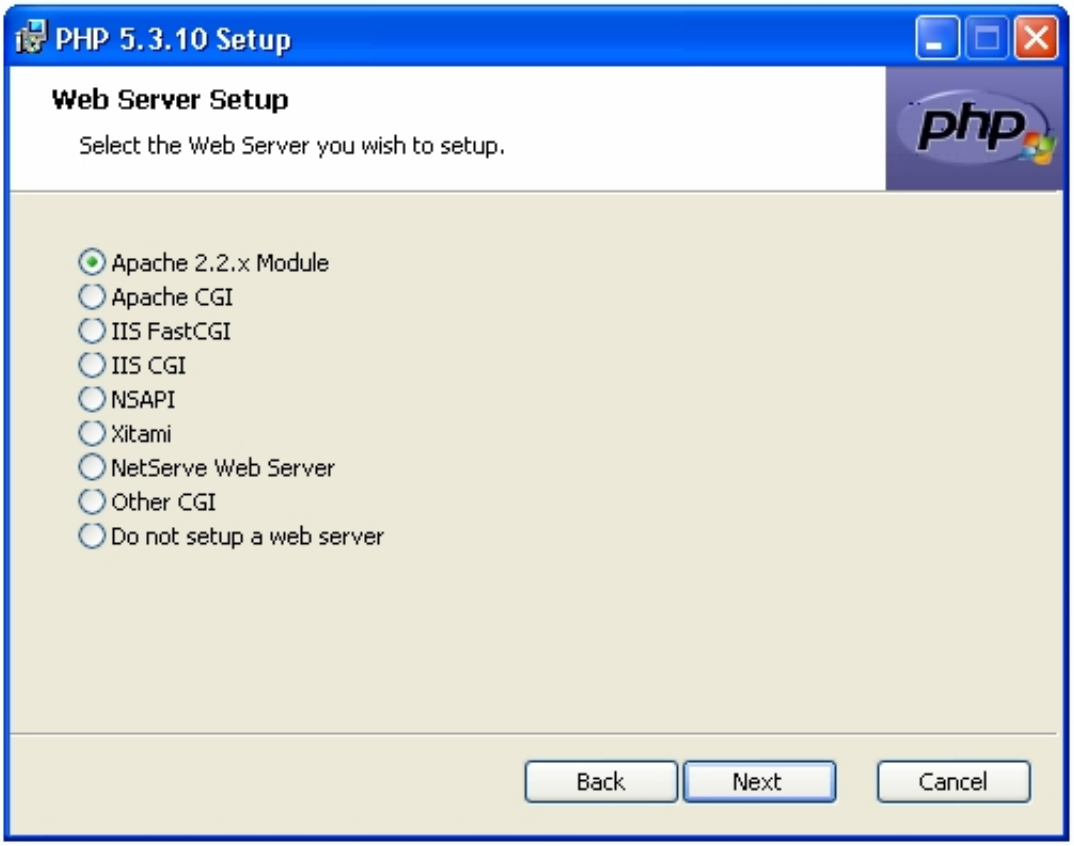


Рис. 3.12. Вибір сервера, на якому працюватиме PHP

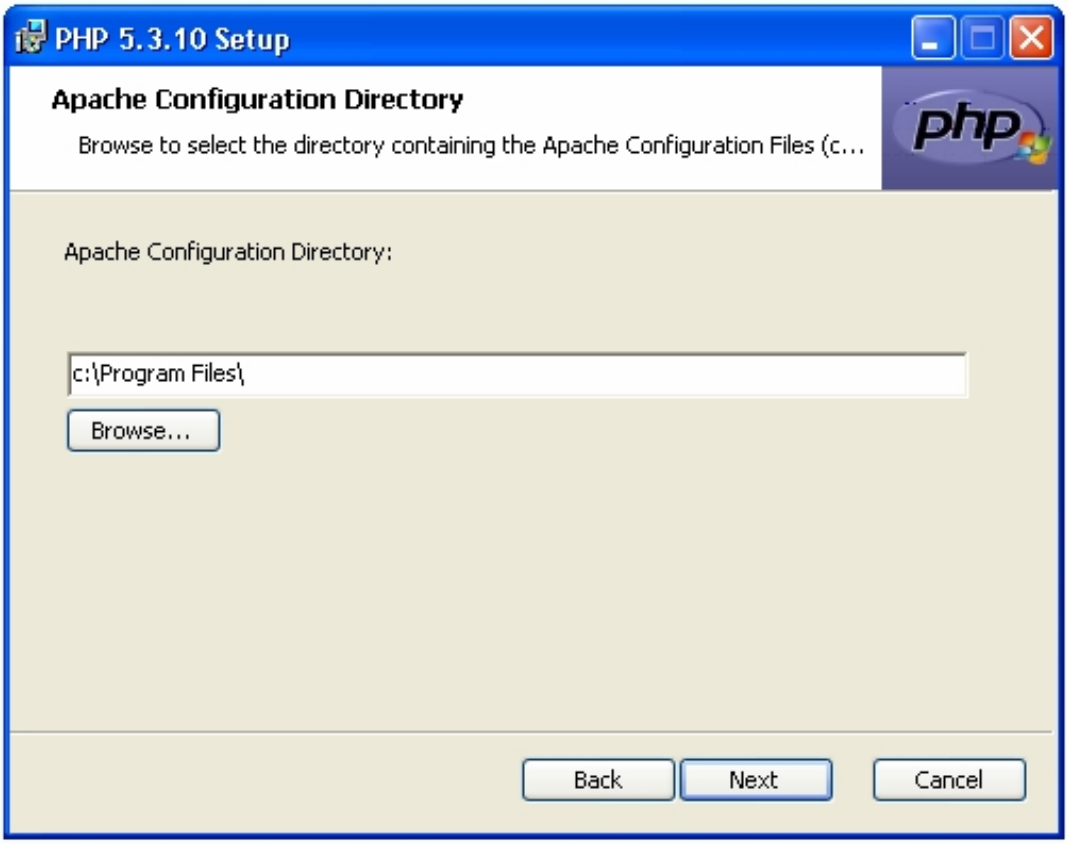


Рис. 3.13. Вибір каталогу для конфігураційних файлів PHP

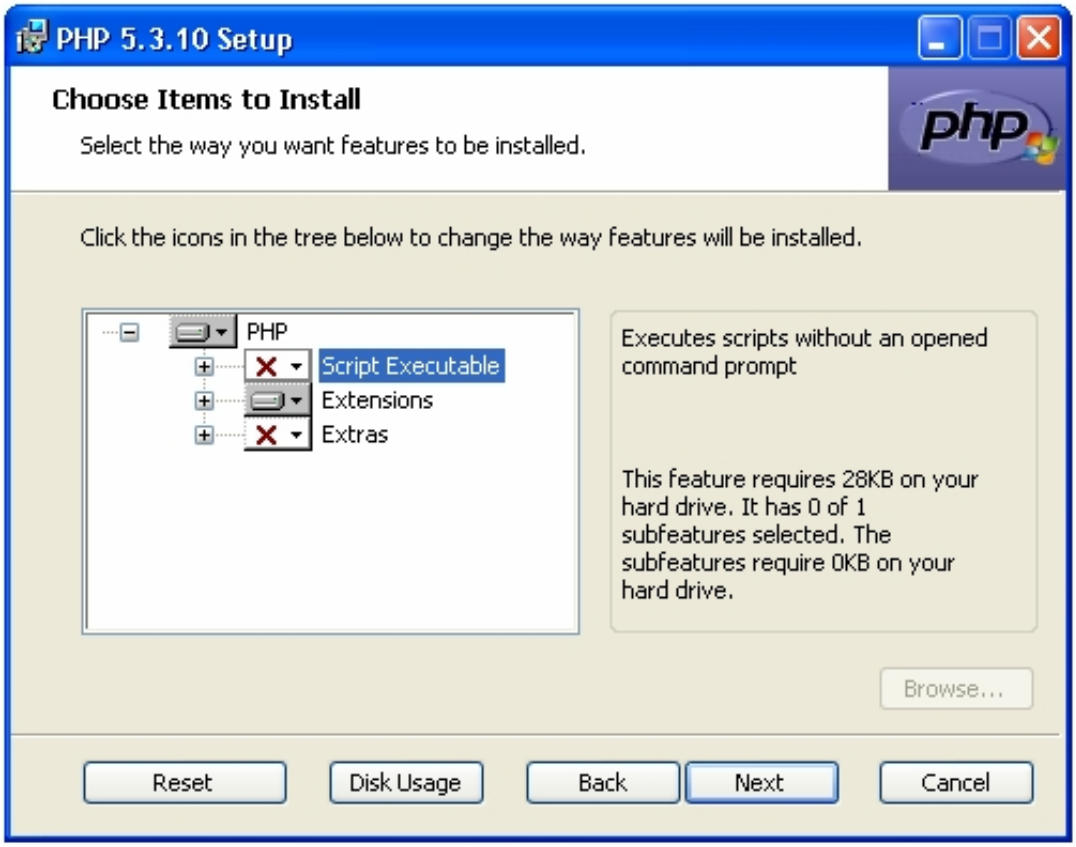


Рис. 3.14. Вибір компонент програми PHP

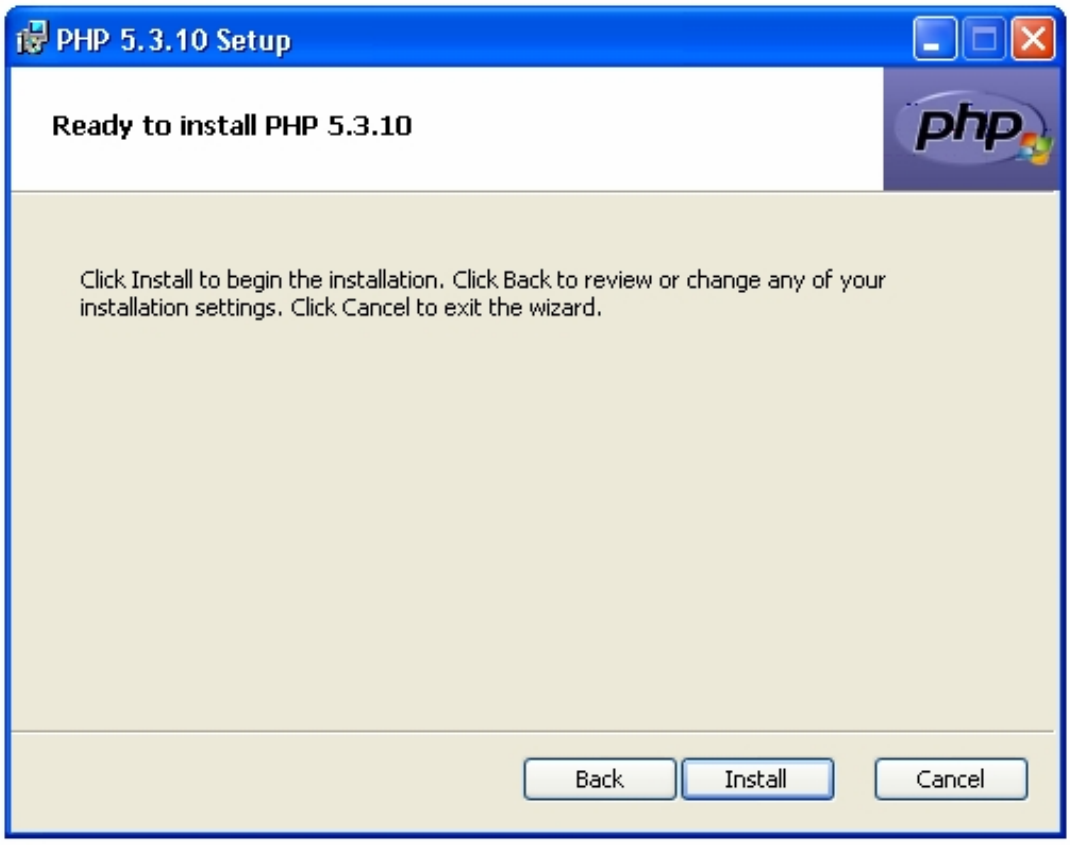


Рис. 3.15. Початок установлення PHP

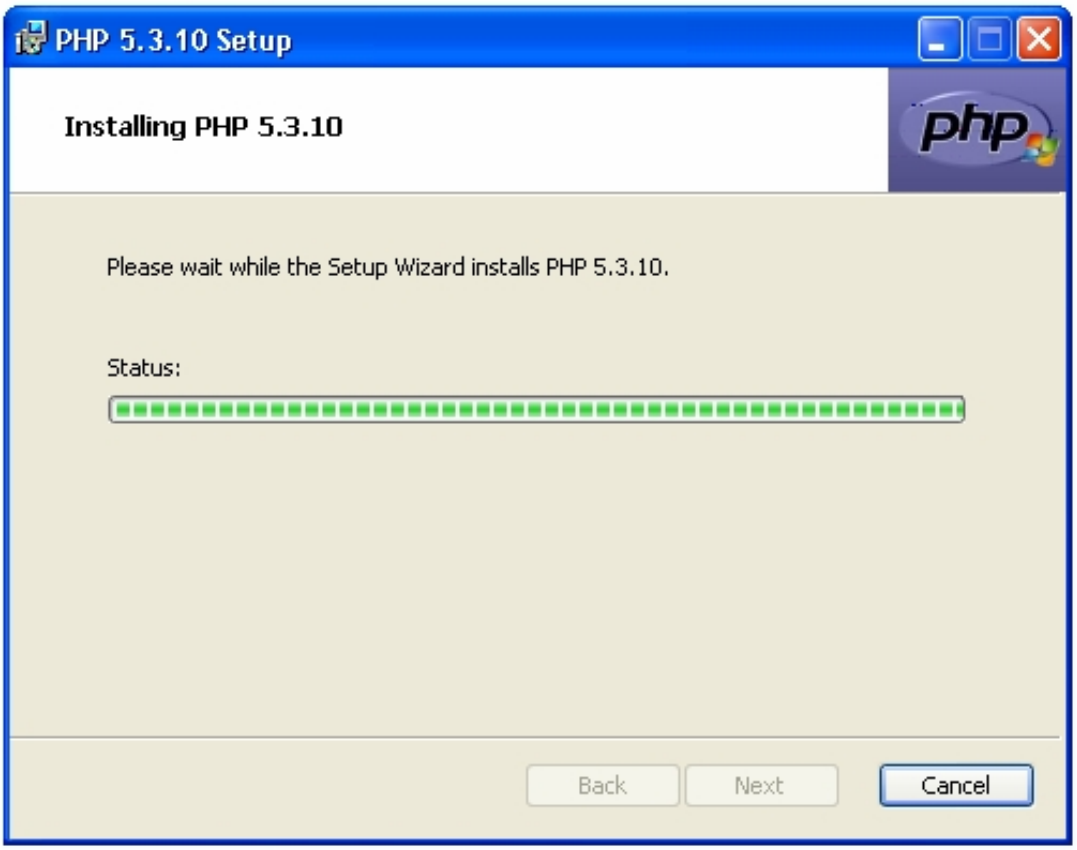


Рис. 3.16. Установлення PHP

Під час установлення програми PHP на екран буде виведене повідомлення, наведене на рис. 3.17, про неможливість автоматичного налаштування, а далі, після натисненя на кнопку Ok, з`явиться повідомлення про закінчення установлення програми PHP (рис. 3.18).

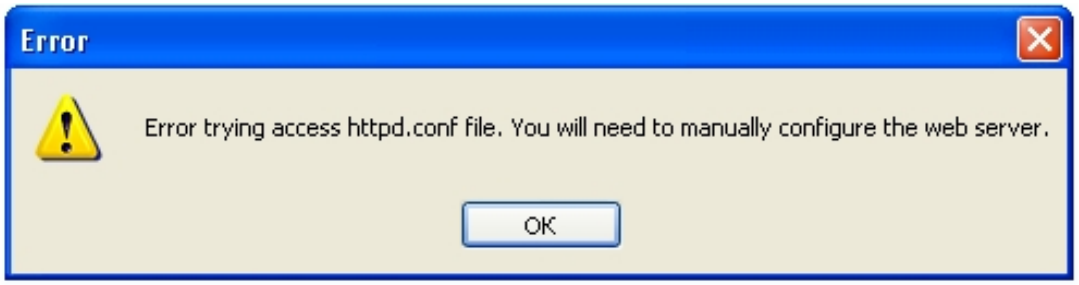


Рис. 3.17. Повідомлення про неможливість автоматичного налаштування

Apache для роботи з PHP

Далі необхідно виконати вручну налаштування сервера Apache для роботи з PHP.

Спочатку слід вибрати тип установки PHP, оскільки він з`являється у двох варіантах: CGI-інтерпретатор (php.exe) та набір SAPI-модулів, які використовуються сервером.

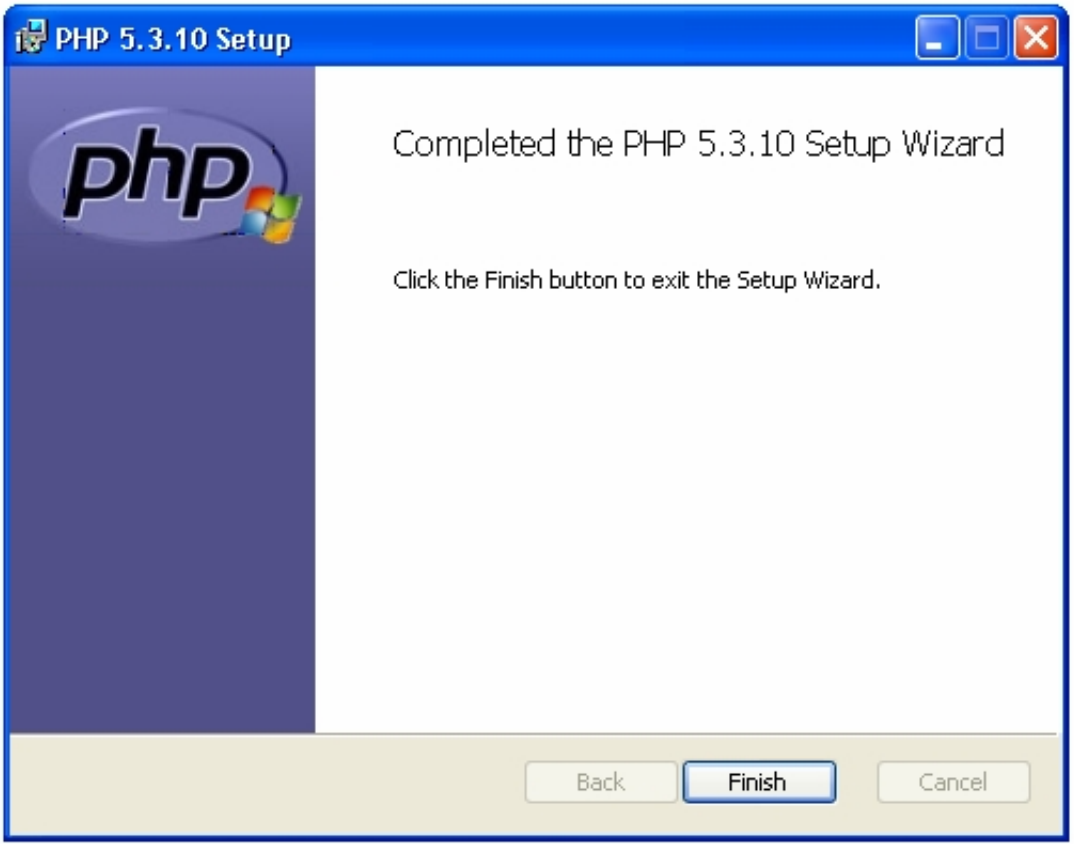


Рис. 3.17. Закінчення установлення PHP

Для установлення PHP як серверного модуля у файлі конфігурації сервера (httpd.conf) потрібно написати:

```
LoadModule php4_module c:/php/sapi/php4apache.dll
AddType application/x-httpd-php .php .phtml
AddModule mod_php4.c
```

Якщо установлення програми PHP виконується як cgi-програма, то в httpd.conf потрібно зазначити:

```
ScriptAlias /php/ "c:/php/"
AddType application/x-httpd-php .php .phtml
Action application/x-httpd-php "/php/php.exe"
```

Після установлення PHP як серверного модуля слід відредагувати файл php.ini (у каталозі C:\Windows), щоб програма PHP визначила, де знаходиться коренева директорія сервера.

У файлі php.ini змінні doc_root, user_dir і extension_dir визначають, де будуть знаходитися призначені для користувача директорії, а де власні бібліотеки розширень програми PHP:

```
doc_root = "C:\Program Files\Apache Group Software Foundation
\Apache2.2\htdocs"
user_dir = "C:\users"
extension_dir = "C:\php\extensions"
```

Крім того, можна вибрати розширення, які завантажуватимуться при запуску програми PHP. У реалізацію PHP для ОС Windows XP спочатку входить дуже мало розширень. Щоб завантажити розширення, потрібно розкоментувати в php.ini (тобто видалити знак #) відповідний йому рядок 'extension=php_*.dll'.

Наприклад, щоб завантажити розширення для роботи з MySQL, слід розкоментувати рядок:

```
extension=php_mysql.dll
```

Деякі розширення вимагають додаткових бібліотек. Тому рекомендується скопіювати їх в системну директорію (з каталогу C:\php\dlls).

При першій установці слід налаштувати і протестувати PHP без розширень.

Для того, щоб налаштування, які вказані в конфігураційних файлах веб-сервера і PHP почали діяти, потрібно перезавантажити сервер.

Для перевірки працездатності PHP необхідно створити тестовий файл (test.php) в директорії користувача (C:\users\student) з таким змістом:

```
<?php
  echo"<h1>Привіт, Україно!</h1>";
?>
```

Для запуску цього файлу через браузер треба набрати http://localhost/~student/test.php.

Якщо все виконано правильно, то програма буде оброблена сервером і на екрані монітору великими літерами буде виведене «Привіт, Україно!», якщо ні – на екрані з`явиться текст цього файлу.

3.2.2. Для ОС Linux

Як і у випадку з ОС Windows, для ОС Linux існує два способи установлення PHP.

Установлення програм PHP і Apache може бути виконане з використанням дистрибутиву AltMaster2.2 (http://altlinux.ru).

Для того, щоб установити Apache, необхідно вибрати відповідний пакет. Це можна зробити за допомогою менеджера пакетів Synaptic.

Скориставшись меню «Пакет -> Встановити», виберемо потрібні пакети.

Після цього необхідно перейти до установлення вибраних пакетів за допомогою меню «Дії → Виконати». Далі треба перезавантажити комп'ютер або виконати таку команду:

```
/etc/init.d/httpd start
# /etc/init.d/httpd start
Starting httpd:
    [ OK ]
```

Це означає, що зроблено запуск веб-сервера. За адресою http://localhost можна буде побачити майже таку саму веб-сторінку, як і при установці сервера Apache для Windows XP. Можливою відмінністю буде присутність логотипу AltLinux.

Після ініціалізації веб-сервера Apache можна приступити до установлення безпосередньо програми PHP.

Знову ж таки існує можливість використовувати програму PHP за допомогою CGI і через модуль Apache. У першому випадку досить пакета php і необхідних для його установлення програм. У другому – треба додатково встановити пакет mod_php.

Менеджер пакетів Synaptic – це оболонка до програми apt-get, детальніше про неї можна дізнатися за допомогою команди man apt-get.

Командою apt-get build-dep mod_php можна встановити пакети, які необхідні для інсталяції mod_php, а командою apt-get install mod_php закінчити установлення:

```
[root@greydragon apt]# apt-get build-dep mod_php
Reading Package Lists... Done
Collecting File Provides... Done
Building Dependency Tree... Done
Note, selecting libgdbm-devel instead of gdbm-devel
The following NEW packages will be installed:
    apache-devel bison byacc flex libexpat-devel
libgdbm-devel libltdl libmm-devel libpam-devel libtool
0 packages upgraded, 10 newly installed
0 removed and 73 not upgraded.
Need to get 937kB of archives.
After unpacking 2357kB of additional disk space will be used.
Do you want to continue? [Y/n]
Get:1 ftp://ftp.altlinux.ru ALTLinux/main byacc
    1.9-ipl9mdk [31.8kB]
.....
Get:10 ftp://ftp.altlinux.ru ALTLinux/main
libtool 2:1.4.2-alt0.2 [302kB]
Fetched 937kB in 9m19s (1675B/s)
Executing RPM (/bin/rpm -Uvh --fancypercent
--oldpackage)...
Preparing...
################################################## [100%]
....
[root@greydragon apt]# apt-get install mod_php
Reading Package Lists... Done
Collecting File Provides... Done
Building Dependency Tree... Done
The following NEW packages will be installed:
    mod_php
0 packages upgraded, 1 newly installed
0 removed and 73 not upgraded.
Need to get 46.6kB of archives.
After unpacking 69.2kB of additional disk space will be used.
```

```
Get:1 ftp://ftp.altlinux.ru ALTLinux/main
mod_php 1:4.3.0-alt0.cvs20030207 [46.6kB]
Fetched 46.6kB in 24s (1918B/s)
Executing RPM (/bin/rpm -Uvh --fancypercent --oldpackage) ...
Preparing...
################################################## [100%]
1: mod_php
################################################## [100%]
Shutting down httpd: [ OK ]
Checking configuration sanity for httpd: [ OK ]
Starting httpd: [ OK ]
```

Як видно вище, програма установлення самостійно перезавантажила веб-сервер Apache, а також самостійно внесла зміни у конфігураційний файл цього веб-сервера. У файл httpd.conf доданий рядок:

```
Include conf/addon-modules/mod_php4.conf
```

Зміст файлу mod_php4.conf:

```
LoadModule php4_module usr/lib/apache/libphp4.so
AddModule mod_php4.c
AddType application/x-httpd-php .php .php4 .php3 .phtml
AddType application/x-httpd-php-source .phps
AddHandler application/x-httpd-php .php .php4 .php3 .phtml
```

Як видно, зміни у файлі httpd.conf подібні до змін, які треба було вносити при установці для ОС Windows.

Для перевірки працездатності програм PHP і Apache необхідно створити файл test.php з таким змістом:

```
<?php
  echo"<h1>Привіт, Україно!</h1>";
?>
```

Кореневий каталог веб-сервера знаходиться у директорії /var/www/html/. Для файлу test.php можна створити директорію test. При запуску цього файлу з браузера за його адресою http://localhost/test/test.php на екран буде виведений напис «Привіт, Україно!».

Процес установлення PHP за допомогою пакетів в інших ОС Linux принципово не відрізняється від розглянутого вище. Слід відзначити, що «ручне» компонування дозволяє налаштувати програму PHP для конкретного користувача, а також забезпечує можливий приріст у продуктивності роботи програми.

3.3. Спрощене установлення програми PHP

У попередньому розділі достатньо детально було розглянуто установлення і налаштування програми PHP для ОС Linux і Windows. Для користувачів, які не бажають вникати в принципи роботи програми PHP, існують готові інсталяційні програми.

Одна з найвідоміших – програма Денвер (http://dklab.ru/chicken/web/). Інструкції щодо її установлення знаходяться на сайті розробників. Варто відзначити, що установлення цієї програми не вимагає особливих навиків. Програма Денвер рекомендується програмістам, які починають працювати з PHP. Для серйозніших завдань краще все ж таки встановити і налаштувати програму PHP самостійно.

3.4. Перша PHP-програма

Розглянемо приклад простого html-файлу з вбудованим кодом мовою PHP.

Приклад 3.1

```
<html>
  <head>
  <title>Example</title>
  </head>
  <body>
  <?php
    echo "<p>PHP Script data</p>";
  ?>
</body>
</html>
```

Це простий HTML-файл, у який вбудований за допомогою спеціальних тегів код, написаний мовою PHP.

Як вже було зазначено вище, мова PHP схожа на Сі і Perl. Проте наведена програма помітно відрізняється від аналогічних програм мовою Сі та Perl. Тут не потрібно писати велику кількість спеціальних команд для виведення HTML. Пишеться безпосередньо HTML-код, у який можна вбудовувати PHP-код, що здійснює які-небудь дії.

Недоліком мови PHP на відміну від Сі і Perl, все ж таки є невисока швидкість виконання складних програм.

PHP-програми виконуються та обробляються сервером. Отже, порівнювати їх з програмами мовами типу JavaScript неможливо. Написані на них програми виконуються на комп'ютері клієнта.

У чому відмінність програм, що виконуються на сервері та на комп’ютерах клієнтів?

Якщо програма обробляється сервером, клієнту посилаються тільки результати її роботи. Наприклад, якщо на сервері виконувалася програма, подібна до наведеної вище (приклад 3.1), клієнт одержить HTML-сторінку в такому вигляді:

```
<html>
    <head>
    <title>Example</title>
    </head>
    <body>
```

```
    <p>PHP Script data </p>
    </body>
</html>
```

У цьому випадку клієнт не знає, який код виконується.

Було відмічено вище, що PHP-програми вбудовуються в HTML-код. Виникає питання, яким чином?

Є декілька способів. Один з них наведений в прикладі 3.1 – за допомогою тегів, що відкривають <?php і закривають ?>. Спеціальні теги такого виду дозволяють перемикатися між режимами HTML і PHP. Цей синтаксис дає можливість використати мову PHP в XML-сумісних програмах.

Коли програма PHP обробляє файл, вона просто передає його текст, поки не зустріне один із спеціальних тегів, які повідомляють про необхідність почати інтерпретацію тексту як коду PHP. Потім вона виконує весь знайдений код до тегу, що закриває, який, усвою чергу, повідомляє інтерпретатор, що далі знову йде просто текст. Цей механізм дозволяє розмістити PHP-код в HTML – все за межами тегів PHP залишається незмінним, тоді як всередині інтерпретується як код. Відмітимо також, що php-файл не схожий на CGI-програму. PHP файл не повинен бути виконуваний або ще яким-небудь чином помічений.

Для того, щоб відправити php-файл для обробки сервера, потрібно в рядку браузера вказати шлях до цього файлу. Якщо php-файл знаходиться на локальному комп'ютері, то його можна обробити за допомогою інтерпретатора командного рядка.

Висновки

У даному розділі розглянуто такі основні питання:

- визначення веб-сервера та інтерпретатора PHP;
- встановлення веб-сервера і програми PHP;
- налаштування веб-сервера і програми PHP для роботи з ними;
- структура PHP-програм.

Контрольні питання

1. Дати визначення веб-сервера Apache.
2. Які є способи установки веб-сервера Apache?
3. Для чого потрібно встановлювати PHP-інтерпретатор?
4. Які ОС можна використовувати для виконання PHP-програм.
5. Які файли необхідно налаштувати для того, щоб забезпечити роботу веб-сервера та PHP-інтерпретатора?
6. Яку структуру мають PHP-програми?
7. Яким програмним забезпеченням можна користуватися для написання програм мовою PHP?
8. Як виконати програму, написану мовою PHP?
9. Чому в браузер не передається код PHP-програми?
10. Що таке інсталяційна програма Денвер?

4. ОСНОВИ СИНТАКСИСУ МОВИ PHP

Навчальною метою розділу є ознайомлення студентів із синтаксисом мови PHP.

У результаті вивчення даного розділу студенти повинні знати:

– способи розділення інструкцій PHP;
– створення коментарів;
– змінні, константи;
– типи даних;
– види операторів.

4.1. Структура програми

4.1.1. Розміщення PHP-програм

Код PHP-програм вбудовується в HTML-код. Щоб інтерпретатор зрозумів, що це код, на мові PHP застосовуються спеціальні позначення.

У попередньому розділі згадувалося про те, що в програмах найчастіше використовують такий варіант: `<?php` – початок коду, а `?>` – кінець коду.

4.1.2. Розділення інструкцій

Програма мовою PHP – це набір команд (інструкцій). Обробнику програми (парсеру) необхідно якось відрізняти одну команду від іншої. Для цього використовуються спеціальні символи – роздільники.

У PHP-програмах інструкції розділяються так само, як і в Сі або Perl, – кожен вираз закінчується крапкою з комою.

Тег «`?>`», що закриває також, означає кінець інструкції, тому перед ним крапку з комою можна не ставити.

Наприклад, два еквівалентні фрагменти коду:

```
<?php
// Крапка з комою в кінці команди обов'язкова
echo "Привіт, Україно!";
?>

<?php
echo " Привіт, Україно!"
?>

<!-- крапка з комою опускається із-за "?>" -->
```

4.1.3. Коментарі

Часто виникає необхідність робити у програмах які-небудь пояснення до коду, що не повинні ніяк впливати на сам код. Для цього застосовують коментарі.

Це також важливо для створення великих програм у разі, коли декілька програмістів працюють над єдиним проектом.

При наявності коментарів у коді програми розібратися набагато легше.

Крім того, якщо це робити частинами, недопрацьовані модулі також треба якось визначати, тобто прокоментувати.

У всіх мовах програмування передбачена можливість включати коментарі в код програми.

Мова PHP підтримує декілька видів коментарів: у стилі Сі, C++ та Unix. Символи // і # позначають початок однорядкових коментарів, /* і */ – відповідно початок і кінець багаторядкових коментарів.

Далі наведено приклади використання різних типів коментарів.

Приклад 4.1

```
<?php

// Це однорядковий коментар у стилі C++

/*
Це багаторядковий коментар. Тут можна написати
декілька рядків. При виконанні програми все,
що знаходиться між символами початку і кінця багаторядкового
коментаря, сприймається програмою інтерпретатора PHP як коментар
і не виконується.
*/

# Це коментар у стилі
# оболонки Unix

?>
```

4.2. Змінні, константи та оператори

Важливим елементом кожної мови є змінні, константи та оператори.

4.2.1. Змінні

Змінна в PHP позначається знаком долара, після якого пишеться її ім'я. Наприклад:

```
$exampleVar
```

Існують деякі правила іменування змінних:

– ім'я змінної чутливе до регістра (тобто змінні $exampleVar і $ExampleVar різні);

– ім'я змінної повинне починатися з букви або символу підкреслення, за ним можуть бути букви, цифри або символи підкреслення.

У мові PHP 3.0 змінні завжди присвоювалися відповідно до значення, тобто в змінну поміщався сам результат, а не посилання на нього.

Це показано у прикладі 4.2.

Приклад 4.2

```
<?php

// Змінній $first присвоюється значення 'Text'
$first = 'Text';

// Змінній $second присвоюється значення змінної $first
$second = $first;

// Змінюємо значення $first на 'New text'
$first = 'New text';

// Виводимо значення $first
echo "Змінна з ім'ям first подібна до $first <br>";

// Виводимо значення $second
echo "Змінна з ім'ям second подібна до $second";

?>
```

Результат роботи програми, вказаної у прикладі 4.2, буде таким:

```
Змінна з ім'ям first подібна до New text
Змінна з ім'ям second подібна до Text
```

Мова PHP 4.0 пропонує ще один спосіб присвоювання значень змінним – за посиланням. Щоб вказати, що значення однієї змінної присвоюється іншій за посиланням, потрібно перед ім'ям першої змінної поставити знак амперсанд &.

У прикладі 4.3 наведена програма аналогічна прикладу 2.2, тільки присвоювання здійснюється за посиланням.

Приклад 4.3

```
<?php

// Присвоюємо $first значення 'Text'
$first = 'Text';

// Робимо посилання на $first через $second.
// тепер значення цих змінних завжди будуть збігатися
$second = &$first;

// Змінюємо значення $first на 'New text'
$first = 'New text';
```

```
// Виведемо значення обох змінних
echo "Змінна з ім'ям first подібна до $first <br>";
echo "Змінна з ім'ям second подібна до $second";

?>
```

Результат роботи програми:

```
Змінна з ім'ям first подібна до New text.
Змінна з ім'ям second подібна до New text.
```

Таким чином разом із змінною $first змінилася і змінна $second.

4.2.2. Константи

Для зберігання постійних величин, значення яких не змінюється в ході виконання програми, використовуються константи. Такими величинами можуть бути математичні константи, паролі, а також будь-які інші значення.

Основна відмінність константи від змінної полягає в тому, що їй не можна присвоїти значення більше ніж один раз, а також її значення не можливо анулювати після її визначення. Крім того, у константи немає позначки у вигляді знаку долара і її не можна визначити простим присвоюванням значення.

Для визначення константи існує спеціальна функція define(). Ось її синтаксис:

```
define("Ім’я_константи", "Значення_константи",
Нечутливість_до_регістра])
```

За умовчанням імена констант чутливі до регістра. Для кожної константи це можна змінити, вказавши як значення аргументу `Нечутливість_до_регістра` значення True.

Існує угода, згідно з якою імена констант завжди пишуться у верхньому регістрі.

Отримати значення константи можна, вказавши її ім'я. На відміну від змінних, не потрібно перед ім'ям константи ставити символ $. Крім того, можна також використовувати функцію constant() з ім'ям константи замість параметра.

У прикладі 4.4 наведена програма визначення констант.

Приклад 4.4

```
<?php
// Визначаємо константу PASSWORD
define("PASSWORD","qwerty");

// Визначаємо регістронезалежну константу PI, яка дорівнює 3.14
define("PI","3.14", True);

// Виводимо значення константи PASSWORD, тобто qwerty
echo (PASSWORD);
```

```
// Аналогічно буде виведене qwerty
echo constant("PASSWORD");

// Інтерпретатор PHP згенерує попередження
// оскільки константа PASSWORD є регістрозалежною
echo (password);

// На екран буде виведене 3.14, оскільки константа PI
// регістронезалежна за визначенням
echo pi;
?>
```

Окрім змінних, що визначаються користувачем, в PHP існує ряд констант, які визначаються самим інтерпретатором. Ось деякі з них:

– __FILE__ зберігає ім'я файлу програми (і шлях до нього), який виконується у даний момент;

– __FUNCTION__ містить ім'я функції;

– __CLASS__ – ім'я класу;

– PHP_VERSION – версія інтерпретатора мови PHP.

4.2.3. Оператори

Оператори дозволяють виконувати різні дії зі змінними, константами і виразами.

Вираз можна визначити як все, що має значення.

Змінні й константи – це основні та найбільш прості форми виразів. Існує безліч операцій (і відповідних їм операторів), які можна проводити з виразами.

Розглянемо деякі з них докладніше (табл. 4.1 – 4.5).

Таблиця 4.1

Арифметичні оператори

Позначення	Назва	Приклад
+	Складання	$a + $b
-	Віднімання	$a - $b
*	Множення	$a * $b
/	Ділення	$a / $b
%	Залишок від ділення	$a % $b
.	Конкатенація (складання рядків)	$c = $a . $b (рядок, що складається з $a і $b)

Таблиця 4.2

Оператори присвоювання

Позначення	Назва	Опис	Приклад
=	Присвоювання	Змінною зліва від оператора буде присвоєне значення, одержане в результаті виконання яких-небудь операцій або у випадку змінної / константи з правого боку	$a = ($b = 4) +5; ($a буде дорівнювати 9, $b буде дорівнювати 4)
+=		Скорочено позначає комбінацію операцій складання і присвоєння (спочатку додає до змінної число, потім присвоює їй набуте значення)	$a += 5; (еквівалентно $a = $a + 5;)
.=		Скорочено позначає комбінацію операцій конкатенації і присвоєння (спочатку додається рядок, потім одержаний рядок записується в змінну)	$b = "Привіт "; $b .= "всім"; (еквівалентно $b = $b . "всім";) В результаті: $b="Привіт всім"

Таблиця 4.3

Логічні оператори

Позначення	Назва	Опис	Приклад
and	І	Результат операції дорівнює True, якщо $a і $b істинні (True)	$a and $b
&&			
or	Або	Результат операції дорівнює True, якщо хоч би одна із змінних $a або $b дорівнює True (можливо, що й обидві)	$a or $b
||			$a || $b
xor	Виключне або	Результат операції дорівнює True, якщо одна із змінних істинна. Випадок, коли вони обидві істинні, виключається	$a xor $b
!	Інверсія (NOT)	Якщо $a=True, то !$a=False, і навпаки	! $a

Таблиця 4.4

Оператори порівняння

Позначення	Назва	Опис	Приклад
==	Рівність	Значення змінних рівні між собою	$a == $b
===	Еквівалентність	Рівні між собою значення і типи змінних	$a === $b
!=	Нерівність	Значення змінних не рівні між собою	$a != $b
<>	Нерівність		$a <> $b
!==	Нееквівалентність	Значення і типи змінних не рівні між собою	$a !== $b
<	Менше	Значення змінної $a менше значення змінної $b	$a < $b
>	Більше	Значення змінної $a більше значення змінної $b	$a > $b
<=	Менше або дорівнює	Значення змінної $a менше або дорівнює значенню змінної $b	$a <= $b
>=	Більше або дорівнює	Значення змінної $a більше або дорівнює значенню змінної $b	$a >= $b

Таблиця 4.5

Оператори інкремента-декременту

Позначення	Назва	Опис
++$a	Пре-інкремент	Збільшує $a на одиницю і повертає $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

4.3. Типи даних

Мова PHP підтримує вісім простих типів даних, з них
чотири скалярні:

- boolean (логічний);
- integer (цілочисельний);
- float (з плаваючою точкою);
- string (строковий);

два змішані:

- array (масив);
- object (об'єкт);

два спеціальні:

- resource (ресурс);
- NULL.

У мові PHP не прийняте явне визначення типів змінних. Краще, щоб це робив сам інтерпретатор під час виконання програми залежно від контексту, в якому використовується змінна. Розглянемо за порядком всі перераховані типи даних.

4.3.1 Тип boolean (булевий або логічний тип)

Цей простий тип визначає істинність значення, тобто змінна цього типу може мати тільки два значення – правда TRUE або неправда FALSE.

Щоб визначити булевий тип, використовують ключове слово TRUE або FALSE. Вони обидва є регістронезалежними.

Нижче наведено визначення змінної логічного типу.

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

Логічні змінні використовуються в різних керуючих конструкціях. Логічний тип можуть мати також і деякі оператори (наприклад оператор рівності). Вони також використовуються в керуючих конструкціях для перевірки яких-небудь умов.

Наприклад, в умовній конструкції перевіряється істинність значення оператора або змінної. Залежно від результату перевірки виконуються ті або інші дії. В даному випадку умова може бути істинною або помилковою.

У прикладі 4.5 поданий код, що ілюструє використання оператора ‘= =’.

Приклад 4.5

```
<?php
// Оператор '==' перевіряє рівність
// і повертає логічне значення
if ($know == false)
{
  // якщо $know має значення false
  echo "Вивчайте мову PHP";
```

```
}
if (!$know)
{
  // Те саме, що і вище, тобто перевірка,
  // чи має $know значення false
  echo "Вивчайте мову PHP";
}

// Оператор == перевіряє, чи сбігається значення
// змінної $action з рядком "Вивчаю мову PHP".
// Якщо сбігається, то повертає true, інакше – false.
// Якщо повернене true, то виконується те, що всередині
// фігурних дужок

if ($action == "Вивчаю мову PHP")
{
  echo "Почав вивчати";
}

?>
```

4.3.2. Тип integer (цілочисельний)

Цей тип задає число з безлічі цілих чисел Z = {..., -2, -1, 0, 1, 2, ...}. Цілі числа можуть бути вказані в десятковій, шістнадцятковій або вісімковій системах числення. За бажанням перед числом може бути знак «+» або «–», вказуючий на те, що число більше або менше від нуля.

Якщо використовувати вісімкову систему числення, необхідно перед числом вказати 0 (нуль), а якщо шістнадцяткову – 0x.

```
<?php

// Десяткове число
$a = 1234;
// Від'ємне число
$a = -123;
// Вісімкове число
// (еквівалентно 83 в десятковій системі)
$a = 0123;
// Шістнадцяткове число
// (еквівалентно 26 в десятковій системі)
$a = 0x1A;

?>
```

Розмір цілого числа залежить від ОС, хоча, як правило, максимальне значення складає близько двох мільярдів (32-бітове знакове число). Беззнакові цілі числа мова PHP не підтримує.

Якщо буде визначене число, що перевищує межі цілого типу, його буде інтерпретовано як число з плаваючою точкою. Також при використанні оператора, результатом роботи якого буде число, що перевищує межі цілого, замість нього буде повернене число з плаваючою точкою.

У мові PHP не існує оператора ділення цілих чисел. Результатом 1/2 буде число з плаваючою точкою 0.5. Можна звести це значення до цілого, але така дія округлить його в меншу сторону. Існує також можливість використовувати функцію round(), що округляє значення за стандартними правилами.

Щоб надати змінній конкретного типу, потрібно цей тип вказати перед нею в дужках. Наприклад, для перетворення змінної $a = 0.5 до цілого типу необхідно написати:

```
(integer)(0.5);
```

або

```
(integer) $a;
```

Можливість явного надання типів за таким принципом існує для всіх типів даних. Проте варто не забувати, що не завжди значення одного типу можна звести до іншого.

4.3.3. Тип float (числа з плаваючою точкою)

Числа з плаваючою точкою (вони ж дійсні числа) можуть бути визначені за допомогою будь-якого з таких синтаксисів:

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

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

4.3.4. Тип string

Цей тип являє собою набір символів. У мові PHP символ – те саме, що і байт. Це означає, що існує 256 різних символів. Мова PHP не має вбудованої підтримки Unicode, а також у ній не існує обмежень на довжину рядка.

Рядок у мові PHP може бути визначений трьома різними способами:

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

Спосіб визначення рядка за допомогою одинарних лапок «'» – простий. Щоб використовувати одинарну лапку всередині рядка, перед нею необхідно поставити символ зворотної косої межі (слеш) «\'». Цей процес називається екрануванням.

Якщо зворотна коса межа повинна йти перед одинарною лапкою або бути в кінці рядка, необхідно продублювати її «\\'».

Якщо всередині рядка, розміщеного в одинарних лапках, зворотна коса межа «\» зустрічається перед будь-яким іншим символом (відмінним від «\» і «'»), то він розглядається як звичайний символ і виводиться, як і всі інші. Тому зворотну косу межу необхідно екранувати, але тільки тоді, коли вона знаходиться в кінці рядка, перед закриваючою лапкою.

Мова PHP має ряд комбінацій символів, що починаються із символу зворотної косої межі. Їх називають керуючими послідовностями. Вони мають спеціальні значення, які будуть розглянуті пізніше. Варто розуміти, що змінні й такі керуючі послідовності, розміщені в одинарних лапках, не обробляються.

Нижче у прикладі 4.6 подана програма, що ілюструє роботу керуючих послідовностей.

Приклад 4.6

```
<?php

echo 'Існує можливість розміщувати
    символ переходу на новий рядок таким чином.
    При цьому помилки не виникне';

// Буде виведено:
// Щоб вивести символ одинарної лапки ', необхідно перед ним
// поставити символ зворотної косої межі \

echo 'Щоб вивести символ одинарної лапки \', необхідно перед ним';
echo 'поставити символ зворотної косої межі \';

// Буде виведено:
// Ви хочете видалити C:\*.*?
echo 'Ви хочете видалити C:\*.*?';

// Буде виведено:
// Перехід на наступний рядок \n не буде проведено
echo 'Перехід на наступний рядок \n не буде проведено';

// Буде виведено:
// Значення $expand і $either не будуть виведені
echo 'Значення $expand і $either не будуть виведені

?>
```

Якщо рядок поміщений в подвійні лапки «"», мова PHP розпізнає більшу кількість керуючих послідовностей для спеціальних символів. Деякі з них наведені в табл. 4.6.

Найважливішою властивістю рядків у подвійних лапках є обробка змінних.

Таблиця 4.6

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)
\	Зворотна коса межа
\$	Знак долара
\"	Подвійна лапка

Третій спосіб визначення рядків – це використання heredoc-синтаксису.

Символи <<< «ідентифікатор» позначають початок рядка, «ідентифікатор» – його кінець. Ідентифікатор, що закриває рядок, повинен розміщатися в першому стовпці.

Крім того, ідентифікатор повинен відповідати тим самим правилам іменування, що і вся решта міток мови PHP:

– містити тільки літеро-цифрові символи і знак підкреслення;

– починатися не з цифри або знаку підкреслення.

Heredoc-текст поводиться так само, як і рядок у подвійних лапках, при цьому не маючи їх. Це означає, що немає необхідності екранувати лапки, але, як і раніше, можна використовувати розглянуті керуючі послідовності. Змінні всередині heredoc теж обробляються.

Нижче у прикладі 4.7 подана програма, що ілюструє роботу heredoc-синтактиса.

Приклад 4.7

```
<?php

$str = <<<EOD
Приклад рядка, що включає декілька
рядків, з використанням
heredoc-синтаксису
EOD;
// Тут вказаний ідентифікатор – EOD. Нижче
// ідентифікатор EOT
$name = 'Сергій';
echo <<<EOT
Мене звуть "$name".
EOT;
// На екран буде виведено:
// "Мене звуть "Сергій"."

?>
```

Підтримка heredoc була додана у мові PHP 4.0.

4.3.5. Тип array (масив)

Масив мови PHP є набір упорядкованих елементів. Визначити масив можна за допомогою конструкції array() або безпосередньо задаючи значення його елементам.

```
array ([key] => value [key1] => value1, ... )
```

Мовна конструкція array() приймає пари `ключ => значення`, які розділені комами. Символ => встановлює відповідність між значенням і його ключем. Ключ може бути як цілим числом, так і рядком, а значення – будь-якого PHP-типу. Числовий ключ масиву часто називають індексом. Індексування масиву мови PHP починається з нуля. Значення елемента масиву можна отримати, вказавши після імені масиву в квадратних дужках ключ потрібного елемента. Якщо ключ масиву є стандартним записом цілого числа, то він розглядається як число, інакше – як рядок. Тому запис $a["1"] означає те саме, що і запис $a[1].

Нижче наведена програма, що ілюструє присвоювання значень елементів масиву.

```
<?php

$books = array ("php" => "PHP users guide", 12 => true);

//Буде виведено: "PHP users guide"
echo $books["php"];

//Буде виведено: 1
echo $books[12];

?>
```

Якщо для елементу ключ не заданий, то як ключ береться максимальний числовий ключ, збільшений на одиницю. Якщо вказати ключ, для елементу якого вже було присвоєне значення – його буде перезаписано. Починаючи з PHP 4.3.0, якщо максимальний ключ – від'ємне число, то наступним ключем масиву буде нуль (0).

```
<?php

// масиви $arrayOne і $arrayTwo еквівалентні
$arrayOne = array(5 => 43, 32, 56, "b" => 12);
$arrayTwo = array(5 => 43, 6 => 32, 7 => 56, "b" => 12);

?>
```

Якщо використовувати як ключ TRUE або FALSE, то його значення переводиться відповідно в одиницю і нуль типу integer.

Якщо використовувати NULL, то замість ключа буде одержаний порожній рядок.

Можна використовувати і порожній рядок як ключ, але тоді його необхідно брати в лапки.

Отже, це не те саме, що застосування порожніх квадратних дужок. Не можна використовувати як ключ масиви та об'єкти.

Створити масив можна просто – присвоїти його елементам значення. Як було раніше сказано, значення елемента масиву можна набути за допомогою квадратних дужок, усередині яких потрібно 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 не вказати ключ, а тільки присвоїти значення $book[]="new_value", то новий елемент масиву матиме числовий ключ, на одиницю більший від максимального. Якщо масив, у який додається значення, ще не існує, то він буде створений:

```
<?php

// Додавання в масив $books значення value з ключем key
$books["key"]= "value";

// Додавання в масив значення nextValue з ключем 13
// оскільки максимальний ключ був 12
$books[] = "nextValue";

?>
```

Для того, щоб змінити конкретний елемент масиву, потрібно просто присвоїти йому нове значення. Не можливо змінити ключ елемента, можна тільки видалити елемент (пару ключ/значення) і додати новий.

Щоб видалити елемент масиву, потрібно використовувати функцію unset().

Приклад 4.8

```
<?php
// Створення масиву
$books = array ("php" => "PHP users guide", 12 => true);

// Додавання елемента з ключем 13
$books[] = "Book about Perl";

// Додавання нового елемента з ключем "lisp" і значенням 123456
$books["lisp"] = 123456;

// Видалення елемента з ключем 12 з масиву
unset($books[12]);

// Видалення масиву повністю
unset ($books);
?>
```

Необхідно знати, що, коли використовуються порожні квадратні дужки, максимальний числовий ключ шукається серед ключів, що існують у масиві з моменту останнього переіндексування. Переіндексувати масив можна за допомогою функції array_values(), як це показано в поданій у прикладі 4.9 програмі.

Приклад 4.9

```
<?php

// Створення  масиву із значеннями "a", "b" і "c".
// Оскільки ключі не вказані, вони будуть 0, 1, 2 відповідно
$arr = array ("a", "b", "c");

// Виведення масиву ключів і значень масиву на екран
print_r($arr);

// Видалення з масиву всіх значень
unset($arr[0]);
unset($arr[1]);
unset($arr[2]);

// Виведення на екран ключів і значень масиву
print_r($arr);

// Додавання нового елемента в масив
// при цьому його індексом буде 3, а не 0
$arr[] = "newValue";
print_r($arr);

// Перєіндексування масиву
$arr = array_values($arr);

// Ключем цього елемента буде 1
$arr[] = "newValue2";

print_r($arr);
?>
```

Результатом роботи розглянутої програми буде таке виведення на екран:

```
Array ( [0] => a [1] => b [2] => c )
Array ( )
Array ( [3] => newValue )
Array ( [0] => newValue [1] => newValue2 )
```

4.3.6. Тип object (об'єкт)

Об'єкт – тип даних, що прийшов з об'єктно-орієнтованого програмування (ООП). Згідно з принципами ООП клас – це набір об'єктів, що володіють певними властивостями і методами роботи з ним. Об'єкт відповідно є екземпляром 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 взяти одного конкретного програміста, то можна сказати:

- він є об'єктом класу програмістів;
- володіє тими ж властивостями, що й інші програмісти, також має ім'я;
- пише програми і т. д.

У мові PHP для доступу до методів об'єкта використовується оператор ->.

Для ініціалізації об'єкта застосовується вираз new, що створює в змінній екземпляр об'єкта (див. приклад програми 4.10).

Приклад 4.10

```
<?php

// Створення класу людей
class Person
{
     // Метод, який навчає людину мові PHP
  function know_php()
     {
    echo "Тепер я знаю мову PHP";
  }
}

// Створення об'єкта класу Person
$nikolay = new Person();

// Навчання людини мові PHP
$nikolay -> know_php();

?>
```

Детальніше реалізація принципів ООП в мові PHP буде розглянута в наступних розділах.

4.3.7. Тип resource (ресурс)

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 т. п.).

4.3.8. 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 присвоєне яке-небудь значення;

– вона була видалена за допомогою unset().

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

4.4. Вирішення завдання

Як приклад вирішимо завдання створення шаблону листа для електронної пошти.

Припустімо, що є якесь оголошення і декілька різних людей, яким потрібне це оголошення відправити.

Для цього необхідно зробити шаблон із змістом оголошення, в тексті якого є ряд параметрів, що змінюються (залежно від потенційного одержувача).

Спробуємо використовувати 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.

Більш того, оскільки подій і людей багато, зручно використовувати масиви. Підпис в листі залишається постійним завжди, тому логічно задати його як константу. Щоб не писати дуже довгі й громіздкі рядки, варто використовувати оператора конкатенації.

На основі одержаних знань можна написати програму – приклад 4.11.

Приклад 4.11

```
<?php

// Хай підпис буде константою
define("SIGN","З повагою, Леонід Іванович");

// Створення масиву, для зберігання адресатів
$names = array("Гнат", "Петро", "Сергій");
$events = array(
    "f" => "день відкритих дверей"
    "o" => "відкриття виставки"
    "p" => "бал випускників");

// Складання тексту запрошення
$str = "Шановний (а) " . $names[0];
$str .= "<br>Запрошуємо Вас на " . $events["f"];
$str .= "<br>" . SIGN;
// Виведення на екран
echo $str;

?>
```

Висновки

У даному розділі розглянуто такі основні питання:

- основи синтаксису мови PHP;
- змінні;
- константи;
- оператори;
- типи даних.

Контрольні питання

1. Яким чином визначається код PHP-програм?
2. Для чого потрібні коментарі, які бувають типи коментарів?
3. Що таке змінні мови PHP?
4. Які відомі правила іменування змінних?
5. Яку роль у змінних відіграє знак амперсанд &.
6. Яка різниця між змінними і константами?
7. Які відомі оператори мови PHP?
8. Чим відрізняються оператори «==» та «===»мови PHP?
9. Які відомі типи змінних мови PHP?
10. Опишіть роботу зі змінними типу object мови PHP?

5. КЕРУЮЧІ КОНСТРУКЦІЇ МОВИ PHP

Навчальною метою розділу є ознайомлення студентів з керуючими конструкціями мови PHP.

У результаті вивчення даного розділу студенти повинні знати:

- умовні оператори мови PHP;
- циклічні конструкції мови PHP;
- оператори включення мови PHP;
- альтернативний синтаксис керуючих конструкцій мови PHP.

5.1. Умовні оператори

5.1.1. Оператор if

Оператор if – це один з найважливіших операторів багатьох мов, включаючи PHP. Він дозволяє виконувати фрагменти коду залежно від умови. Структуру оператора if можна подати в такий спосіб:

```
if (вираз) блок_виконання
```

У процесі обробки програми виразу буде надано логічний тип. Якщо значення виразу істинно (true), то виконується `блок_виконання`. У противному разі `блок_виконання` ігнорується. Якщо `блок_виконання` містить кілька команд, то він повинен бути поміщений у фігурні дужки { }.

У false будуть переведені такі значення:

- логічне false;
- цілий нуль (0);
- дійсний нуль (0.0);
- порожній рядок і рядок "0";
- масив, що не містить елементів;
- об'єкт, що не містить змінних;
- спеціальний тип NULL.

Всі інші значення будуть зведені до true.

У прикладі 5.1 наведена програма, що ілюструє роботу умовного оператора if.

Приклад 5.1

```
<?php

$names = array("Гнат", "Петро", "Сергій");
if ($names[0] == "Гнат")
{
 echo "Привіт, Гнате!";
 $num = 1;
 $account = 2000;
}
if ($num) echo "Гнат перший у списку";
$bax = 30;
```

```
if ($account > 100 * $bax + 3)
{
  echo "Цей рядок не з'явиться на екрані";
  echo "тому що умова не виконана";
}

?>
```

5.1.2. Оператор else

Вище була розглянута тільки одна, основна частина оператора if. Існує кілька розширень цього оператора. Оператор else розширює if на випадок, якщо вираз if є неправильним, і дозволяє виконати які-небудь дії за умови, що структуру оператора if, розширеного за допомогою оператора else, можна подати в такий спосіб:

```
if (вираз) блок_виконання
else блок_виконання_2
```

Цю конструкцію if...else можна інтерпретувати приблизно так: якщо виконано умову (тобто вираз дорівнює true), то виконуються дії з `блоку_виконання`, у противному разі – дії з `блоку_виконання_2`. Використовувати оператор else не обов'язково.

Нижче наведена програма, що демонструє використання оператора else.

Приклад 5.2

```
<?php

$names = array("Гнат", "Петро", "Семен");
if ($names[0] == "Гнат")
{
  echo "Привіт, Гнате!";
  $num = 1;
  $account = 2000;
}
else
{
  echo "Привіт,” . $names[0];
  // А ми чекали Гната
}
if ($num) echo "Гнат перший у списку!";
else echo "Гнат не перший у списку!";
$bax = 30;
if ($account > 100*$bax+3)
    echo "Цей рядок не з'явиться на екрані,
    тому що умова не виконана";
    else echo "Зате з'явиться цей рядок!";

?>
```

5.1.3. Оператор elseif

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, в if не виконується. Але на відміну від else, альтернативні дії будуть виконані, тільки якщо elseif-умова є правильною. Структуру оператора if, розширеного за допомогою операторів else і elseif, можна подати в такий спосіб:

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

Операторів elseif може бути відразу декілька в одному if-блоці. Elseif-умову буде виконано, тільки якщо попередня if-умова є false, всі попередні elseif-умови є false, а дана elseif-умова – true.

Далі наведений приклад використання оператора elseif.

Приклад 5.3

```
<?php

$names = array("Гнат", "Петро", "Семен");
if ($names[0] == "Іван")
{
  // Якщо перше ім'я в масиві Гнат
  echo "Привіт, Гнате!";
}
elseif ($names[0] == "Петро")
{
  // Якщо перше ім'я
  // не Гнат, а Петро
echo "Привіт, Петро!";
}
elseif ($names[0] == "Семен")
{
  // Якщо перше ім'я не
  // Гнат, не Петро, а Семен
  echo "Привіт, Семене!";
}
else
{
  // Якщо перше ім'я не Гнат,
  // не Петро і не Семен
  echo "Привіт, $names[0]. А ти хто такий?";
}

?>
```

5.1.4. Оператор switch

Конструкція switch дозволяє перевіряти умову і виконувати залежно від цього різні дії. У перекладі назва даного оператора означає «перемикач». Залежно від того, яке значення має змінна, він перемикається між різними блоками. Оператор switch дуже схожий на

`if...elseif...else або набір операторів if.`

Структуру switch можна записати в такий спосіб:

```
switch (вираз або змінна)
{
  case значення1: блок_дій1 break;
  case значення2: блок_дій2 break;
  ...
  default: блок_дій_за_умовчанням
}
```

На відміну від if, тут значенню виразу не надається логічний тип, а просто порівнюється зі значеннями, перерахованими після ключових слів case (`значення1, значення2` і т.д.). Якщо значення виразу збігається з якимось варіантом, то виконується відповідний `блок_дій` – від двокрапки після значення, що збіглося, до кінця switch або до першого оператора break, якщо такий буде знайдено.

Якщо значення виразу не збіглося з жодним з варіантів, то виконуються дії за умовчанням (`блок_дій_за_умовчанням`), що знаходяться після ключового слова default.

Вираз у switch обчислюється тільки один раз, а в операторі elseif – щоразу, тому, якщо вираз досить складний, switch працює швидше.

Програму з приклада 5.3 можна переписати з використанням switch (див. приклад 5.4).

Приклад 5.4

```
<?php
$names = array("Гнат", "Петро", "Семен");
switch ($names[0])
{
  case "Гнат":
    echo "Привіт, Гнате!";
  break;
  case "Петро":
    echo "Привіт, Петре!";
  break;
  case "Семен":
    echo "Привіт, Семене!";
  break;
  default:
    echo "Привіт, $names[0].
    А як Вас звати?";
} ?>
```

Якщо в цьому прикладі опустити оператор break, наприклад в case "Петро", то, якщо змінна дорівнюватиме рядку "Петро", після виводу на екран повідомлення "Привіт, Петре!" програма піде далі й виведе також повідомлення "Привіт, Семене!", 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 заміняється двокрапкою, а закриваюча – endswitch відповідно.

5.2. Оператори передачі керування

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

5.2.1. Оператор break

Оператор break закінчує виконання поточного циклу, будь то for, foreach, while, do..while або switch. Оператор break може використовуватися з числовим аргументом, що вказує кількість керуючих структур, які потрібно завершити.

Приклад 5.5

```
<?php
$i=1;
while ($i) {
    // Генеруємо довільне число від 1 до 10
    $n = rand(1,10);
    // Виводимо номер ітерації та отримане число
    echo "$i:$n ";

    /* Якщо було отримане число 5,
    то припиняємо роботу циклу. У цьому випадку
    все, що перебуває після цього рядка
    всередині циклу, не буде виконане */
    if ($n==5) break;

    echo "Цикл працює <br>";
    $i++;
}
echo "<br>Число ітерацій циклу $i ";
?>
```

Результат роботи цієї програми приблизно такий:

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

Якщо після оператора break указати число, то перерветься саме така кількість утримуючих цей оператор циклів.

У наведеному вище прикладі це неактуально, оскільки вкладених циклів немає. Трохи змінимо програму прикладу 5.5.

Приклад 5.6

```
<?php
$i=1;

while ($i) {
  $n = rand(1,10);
  // Генеруємо довільне число від 1 до 10
  switch ($n){
    case 5:
      // Припиняємо роботу switch
      // (першого утримуючого break циклу)
      echo "Вихід з switch (n=$n)";
      break 1;
    case 10:
      // Припиняємо роботу switch і while
      // (двох утримуючих break циклів)
      echo "Вихід з switch і while (n=$n)";
      break 2;
    default:
      echo "switch працює (n=$n), ";
  }
  echo " while працює - крок $i <br>";
  $i++;
}
echo "<br>Число ітерацій циклу $i ";

?>
```

5.2.2. Оператор continue

Іноді потрібно не повністю припинити роботу циклу, а тільки продовжити, почавши його з нової ітерації. Оператор continue дозволяє пропустити подальші інструкції з `блоку_виконання` будь-якого циклу і продовжити виконання з ітерації. Оператор continue можна використати з числовим аргументом, що вказує, яке число утримуючих його керуючих конструкцій повинні завершити роботу.

Замінимо в попередньому прикладі оператор break на continue. Крім того, обмежимо число кроків циклу чотирма.

Приклад 5.7

```
<?php
$i=1;
while ($i<4) {
  // Генеруємо довільне число від 1 до 10
  $n = rand(1,10);
  // Виводимо номер ітерації та отримане число
  echo "$i:$n ";
```

```
  if ($n==5) {
    echo "Нова ітерація ";
    continue;

    /* Якщо було отримано число 5,
    то починаємо нову ітерацію циклу,
    $i не збільшується */
  }
  echo "Цикл працює <br>";
  $i++;
}

echo "<br>Число ітерацій циклу $i ";
?>
```

Результатом роботи цієї програми буде:

```
1:10 Цикл працює
2:5 Нова ітерація
2:1 Цикл працює
3:1 Цикл працює
Число ітерацій циклу 4
```

Треба зазначити, що після виконання оператора continue, робота циклу не закінчується. У прикладі лічильник циклу не змінюється у випадку одержання 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 виразу.

Приклад 5.8

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

Мова PHP має ще одну особливість використання оператора continue – у конструкціях switch він працює так само, як і break. Якщо switch знаходиться у середині циклу, потрібно почати нову ітерацію циклу.

5.3. Оператори включення

5.3.1. Оператор include

Оператор include дозволяє включати код, який знаходиться у файлі, і виконувати його стільки разів, скільки програма зустрічає цей оператор.

Така операція може виконуватися кожним з перерахованих способів:

```
include 'ім'я_файлу';
include $file_name;
include ("ім'я_файлу");
```

Нехай у файлі params.inc зберігається набір якихось параметрів і функцій. Щоразу, коли нам потрібно буде використати ці параметри (функції), ми будемо вставляти в текст нашої основної програми команду include 'params.inc'.

Приклад 5.9

```
params.inc
<?php
$user = "Миколо";

// Функція date() повертає дату і час
//(тут - дату у форматі день.місяць.рік)
$today = date("d.m.y");
?>
include.php
<?php
include ("params.inc");
// Змінні $user і $today задані у файлі params.inc
// Тут теж можна скористатися завдяки команді
include("params.inc") */
// Виведе "Привіт, Миколо!"
echo "Привіт, $user!<br>";
// Виведе, наприклад, "Сьогодні 01.09.12"
echo "Сьогодні $today";
?>
```

Відзначимо, що використання оператора include еквівалентно простій вставці змісту файлу params.inc у код програми include.php.

Може тоді треба було в params.inc записати простий текст без тегів, що вказують на те, що це php-код?

Не можна! Справа в тому, що в момент вставки файлу відбувається перемикання з режиму обробки PHP у режим HTML. Тому код усередині файлу повинен бути поміщений у відповідні теги.

Пошук файлу для вставки відбувається за такими правилами:

– спочатку здійснюється пошук файлу в include_path відповідно до поточної робочої директорії;

– якщо файл не знайдений, то пошук виконується в include_path відносно директорії поточної програми.

Параметр include_path, вказується у файлі конфігурації PHP.

5.3.2. Оператор require

Цей оператор діє приблизно так само, як і #include в C++. Усе, що ми говорили про include, лише за деякими виключеннями, справедливо і для require. Оператор require також дозволяє включати в програму і виконувати який-небудь файл.

Основна відмінність require і include полягає у тому, як вони реагують на виникнення помилки. Робота програми після видачі попередження оператором include триває, але помилка в require викликає фатальну помилку роботи програми і припиняє її виконання.

5.4. Циклічні конструкції

Мова PHP має кілька конструкцій, які дозволяють виконувати дії, що повторюються залежно від умови. Це цикли while, do..while, foreach і for. Розглянемо їх.

5.4.1. Цикл while

Цей цикл має таку структуру:

```
while (вираз) { блок_виконання }
```

або

```
while (вираз): блок_виконання endwhile;
```

While – це простий цикл. Він дозволяє PHP-інтерпретатору виконувати команди `блоку_виконання` доти, доки вираз обчислюється як true (тут, як і в if, `виразу` надається логічний тип).

Значення `виразу` перевіряється кожного разу на початку циклу, так що, навіть якщо його значення змінилося в процесі виконання всіх команд `блоку_виконання`, цикл не буде зупинений до кінця ітерації (тобто поки всі команди `блоку_виконання` не будуть виконані).

Приклад 5.10

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

5.4.2. Цикл do ... while

Цикли do ... while дуже схожі на цикли while, з тією лише різницею, що істинність виразу перевіряється наприкінці циклу, а не на початку. Завдяки цьому `блок_виконання` циклу do...while гарантовано виконується хоча б один раз.

Цей цикл має таку структуру:

```
do {блок_виконання} while (вираз);
```

Приклад 5.11

```
<?
// Ця програма надрукує число 12, незважаючи на те,
// що умова циклу не виконана
$i = 12;
    do {
       // Якщо число парне, то друкуємо його
       if ($i % 2 == 0) print $i;
       // Збільшуємо число на одиницю
       $i++;
    } while ($i<10)
?>
```

5.4.3. Цикл for

Цикли for найбільш складні у мові PHP. Вони нагадують відповідні цикли мови Сі і мають таку структуру:

```
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 `вираз1` обчислюється на початку циклу і лише один раз. На початку кожної ітерації обчислюється `вираз2`. Якщо `вираз2` є істинним, то цикл продовжує своє виконання, поки всі команди `блоку_виконання` не будуть пройдені. Якщо `вираз2` дорівнює false, то виконання циклу зупиняється. Наприкінці кожної ітерації (тобто після виконання всіх команд `блоку_виконання`) обчислюється `вираз3`.

Кожен з виразів 1, 2, 3 може бути порожнім. Якщо `вираз2` – порожній, то це значить, що цикл повинен виконуватися невизначений час (у цьому випадку PHP-інтерпретатор вважає цей вираз завжди істинним). Це зручно коли цикл потрібно зупиняти самостійно, використовуючи оператор break.

Наприклад, всі парні цифри можна вивести з використанням циклу for у такий спосіб:

```
<?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 саме завдання можна вирішити, зупиняючи цикл оператором break.

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

Можна опустити всі три вирази. У цьому випадку просто не буде задане початкове значення лічильника $i і воно не буде змінюватися щоразу наприкінці циклу. Всі ці дії можна записати у вигляді окремих команд або в блоці_виконання, або перед циклом.

Приклад 5.12

```
<?php
// Задаємо початкове значення лічильника
$i=2;

for ( ; ; ) {
  // Якщо $i більше або дорівнює 10,
  // то припиняємо роботу циклу
  if ($i>=10) break;

  // Якщо число парне,
  // те друкуємо його
  if ($i % 2 == 0) print $i;

  // збільшуємо лічильник на одиницю
  $i++;
}
```

?>

У третій вираз конструкції for можна записувати через кому кілька найпростіших команд. Наприклад, якщо ми хочемо просто вивести всі цифри, то програму можна записати зовсім просто:

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

5.4.4. Цикл foreach

Це ще одна корисна конструкція. Вона з'явилася тільки у мові PHP 4.0 і призначена винятково для роботи з масивами.

Її синтаксис:

```
foreach ($array as $value) { блок_виконання }
```

або

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

У першому випадку формується цикл за всіма елементами масиву, заданого змінною $array. На кожному кроці циклу значення поточного елемента масиву записується у змінну $value, а внутрішній лічильник масиву пересувається на одиницю (так що на наступному кроці буде вилучено наступний елемент масиву). Усередині `блоку_виконання` значення поточного елемента масиву може бути отримане за допомогою змінної $value. Застосування `блоку_виконання` відбувається стільки разів, скільки елементів у масиві $array.

Друга форма запису на кожному кроці циклу записує ключ поточного елемента масиву в змінну $key, що також можна використати в `блоці_виконання`.

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

Приклад 5.13

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

5.5. Альтернативний синтаксис керуючих конструкцій

Мова PHP пропонує альтернативний синтаксис для деяких керуючих структур: if, while, for, foreach та switch. У кожному випадку відкриваючу дужку потрібно замінити на двокрапку (:), а закриваючу – на endif;, endwhile; і т. д.

Наприклад, синтаксис оператора if можна записати в такий спосіб:

```
if(вираз): блок_виконання endif;
```

Якщо умова, записана в круглих дужках оператора if, виявилася істиною, буде виконуватися весь код, від двокрапки «:» до команди endif;. Використання такого синтаксису корисно для вбудовування php в html-код.

Далі наведена програма, що ілюструє використання альтернативного синтаксису.

```
<?php

$names = array("Гнат", "Петро", "Семен");

if ($names[0]=="Гнат"):
print "Привіт, Гнате!";
endif;

?>
```

Якщо використовувати конструкції else і elseif, то можна також задіяти альтернативний синтаксис.

Приклад 5.14

```
<?php

$a=1;
if ($a == 5):
    print "a дорівнює 5";
    print "...";
elseif ($a == 6):
    print "a дорівнює 6";
    print "!!!";
else:
    print "a не дорівнює ні 5, ні 6";
endif;

?>
```

5.6. Вирішення завдання

Повернемося до завдання, сформульованого у попередньому розділі.

Створимо програму, яку можна було б використати для відправлення листів (або просто для їхньої генерації) із запрошеннями безлічі користувачів. Але зараз винесемо всю інформацію про людей і події в окремий файл data.php і напишемо програму, що не залежить від цієї інформації та її структури. У цьому випадку для того, щоб, наприклад, розширити список адресатів, не потрібно буде змінювати програму, що генерує запрошення.

Крім того, можна буде використовувати інформацію про людей і події в інших програмах. У самій програмі, що генерує запрошення letters.php, використовуються умовні оператори, цикли, require та інші вивчені раніше конструкції.

Нижче наведено записи програм data.hlp (приклад 5.15) і letters.php (приклад 5.16).

Приклад 5.15

```
<?php

define("SIGN","З повагою адміністрація");
// Нехай наш підпис буде константою інформації про події

$events = array(
    "f" => "день відкритих дверей",
    "o" => "відкриття виставки",
    "p" => "бал випускників");
// Наявна інформація про людей
// (ім'я та електронна адреса)
$people = array(
    "ivan" => array(
       "name" => "Іван Іванович",
        "email"=>"user_ivan@intuit.ru"),
    "pit" => array(
       "name" => "Петро Петрович",
        "email" => "user_petr@intuit.ru"),
    "semen" => array(
       "name" => "Семен Семенович"));
// Хто куди запрошується
// Іван - на виставку
$who_where["ivan"] = "o" ;
// Петро - на бал
$who_where["pit"] = "p";
?>
```

Приклад 5.16

```
<?php
require("data.php");
// Включаємо файл з даними про події
foreach($people as $key => $man_info) {
// для кожної людини робимо таке:
$event_key = $who_where[$key];
// Одержуємо подію, на яку вона запрошується
if ($event_key <> "") {
    foreach($man_info as $key1 => $info){
    // Одержуємо ім'я і email конкретної людини
    if ($key1=="name")
      $str = "Шановний (а), $info";
    if ($key1=="email") $email = $info;
     }
    // Складаємо запрошення
    $str .= "<br>Запрошуємо Вас на ".
       $events[$event_key];
    switch ($event_key){
```

```
     // Залежно від події додаємо який-небудь рядок
        case "f":
        $str .= "<br>Підтвердіть Вашу участь по телефону!";
        break;
        case "o":
        $str .= "<br>Завітайте за 15 хвилин до відкриття!";
        break;
        case "p":
        $str .= "<br>Не забудьте подарунок :-)";
        break;
    }
    $str .= "<br>" . SIGN . "<hr>";
   // додаємо підпис
       echo $str; // Виводимо запрошення на екран
/* якщо у Вас є настрій відповісти, то лист можна
відправити за командою mail($email,"Letter",$str); */
  }
}
?>
```

Висновки

У даному розділі розглянуто такі основні питання:

- умовні оператори мови PHP;
- оператори передачі керування мови PHP;
- оператори включення мови PHP;
- циклічні конструкції мови PHP;
- альтернативний синтаксис керуючих конструкцій мови PHP.

Контрольні питання

1. Назвіть керуючі конструкції мови PHP.
2. Який синтаксис оператора if?
3. Яким чином можна використовувати оператор else без оператора if?
5. Що таке альтернативний синтаксис керуючих конструкцій?
5. У яких випадках використовують оператор switch?
6. Що таке цикли і яким чином їх використовують у програмах?
7. Вкажіть типи циклів, які ви знаєте?
8. З якими даними зручно використовувати цикл foreach?
9. Яким чином можна зупинити роботу циклу?
10. За допомогою яких операторів можна включити код з іншого файлу в програму мови PHP?

6. СТВОРЕННЯ І ВИКОРИСТАННЯ ФУНКЦІЙ

Навчальною метою розділу є ознайомлення студентів з функціями.

У результаті вивчення даного розділу студенти повинні знати:

- визначення функції;
- методи створення функцій;

уміти:

- використовувати змінні у функціях.

У цьому розділі будуть розглядатися питання створення і застосування функцій у мові PHP. Використовуючи термін «функції», не мають на увазі всі існуючі в PHP функції, а лише ті, що створені користувачем.

Розглянемо способи завдання таких функцій, методи передачі аргументів, використання аргументів зі значенням за умовчанням.

Розповідаючи про функції, обумовлені користувачем, все-таки не можна не сказати про вбудовані функції. Про деякі з убудованих функцій, таких як echo(), print(), date(), include(), вже розповідалося.

Насправді всі перераховані функції, крім date(), – мовні конструкції. Вони є складовою ядра мови PHP і не вимагають ніяких додаткових налаштувань і модулів. Функція date() теж входить до складу ядра PHP і не вимагає налаштувань. Але є такі функції, для роботи з якими потрібно встановити різні бібліотеки і включення відповідних модулів.

Наприклад, для використання функцій роботи з базою даних MySql треба скомпілювати мову PHP з підтримкою цього розширення. Останнім часом найбільш розповсюджені розширення і відповідно їхні функції включають до складу мови PHP так, щоб з ними можна було працювати без будь-яких додаткових налаштувань інтерпретатора.

6.1. Визначення і виклик функцій

Для чого потрібні функції?

Щоб відповісти на це запитання, потрібно зрозуміти, що взагалі являють собою функції. У програмуванні, як і у математиці, функція є відображенням безлічі її аргументів на безліч її значень. Іншими словами, функція для кожного набору значень аргументу повертає якісь значення, що є результатом її роботи.

Для чого потрібні функції, спробуємо пояснити на прикладі. Класичний приклад функції в програмуванні – це функція, що обчислює значення факторіала числа. Отже, задаємо їй число, а вона повертає нам його факторіал. При цьому не потрібно для кожного числа, факторіал якого ми хочемо одержати, повторювати той самий код – досить просто викликати функцію з аргументом, який дорівнює цьому числу.

Наведемо приклад функції обчислення факторіала натурального числа.

Приклад 6.1

```
<?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,
   // ніж писати echo (50*49*48*...*3*2);
?>
```

Таким чином, коли виконуємо дії, у яких простежується залежність від якихось даних, і при цьому, можливо, необхідно робити те саме, але з іншими вихідними даними, зручно використати механізм функцій – оформити блок дій у вигляді тіла функції, а мінливі дані – як її параметри.

Подивимося, як взагалі виглядає завдання (оголошення) функції.

Функція може бути визначена за допомогою такого синтаксису:

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

Якщо так написати в php-програмі, то працювати нічого не буде.

По-перше, `Ім'я_функції` та імена параметрів функції (`параметр1`, `параметр2` і т.д.) повинні відповідати правилам найменування в мові PHP (символа кирилицею у них краще не використовувати). Імена функцій не чутливі до регістра.

По-друге, параметри функції – це змінні мови, тому перед назвою кожного з них повинен стояти знак $.

По-третє, замість слів `блок_дій` у тілі функції повинен перебувати будь-який правильний PHP-код (не обов'язково залежний від параметрів).

І нарешті, після ключового слова return повинне йти коректний php-вираз (що-небудь, що має значення). Крім того, у функції може і не бути параметрів, так само як і значення, що повертає.

Приклад функції обчислення факторіала (приклад 6.1) – це правильне завдання функції.

Для виклику функції спочатку вказується ім'я функції, а потім у круглих дужках список значень її параметрів, якщо такі є.

Приклад 6.2.

```
<?php
Ім'я_функції ("значення_для_параметра1",
```

```
  "значення_для_параметра2",...);
// приклад виклику функції - виклик функції
// обчислення факторіала, наведений вище,
// там для обчислення факторіала числа 3
// ми писали: fact(3);
// де fact - ім'я викликаної функції,
// а 3 - значення її параметра з ім'ям $n
?>
```

Функцію можна викликати після її визначення, тобто в будь-якому рядку програми нижче блока function f_name(){...}. У мові PHP 3.0 це було дійсно так. Але вже у мові PHP 4.0 такої вимоги немає. Вся справа в тому, як інтерпретатор обробляє одержуваний код.

Єдиний виняток становлять функції, визначені умовно (усередині умовних операторів або інших функцій), тоді її визначення повинне передувати її виклику (приклад 6.3).

Приклад 6.3

```
<?
$make = true;
/* тут не можна викликати Make_event();
тому що вона ще не існує, але можна
викликати Save_info() */

Save_info("Семен","Іванов",
   "Я вибрав курс мови PHP");

if ($make){
// визначення функції Make_event()
  function Make_event(){
    echo "<p>Хочу вивчати Python<br>";
  }
}
// тепер можна викликати Make_event()
Make_event();
// визначення функції Save_info
function Save_info($first, $last, $message){
  echo "<br>$message<br>";
  echo "Ім'я: ". $first . " ". $last . "<br>";
}
Save_info("Сергій","Федоров", "А я вибрав курс мови Lisp");
// Save_info можна викликати і тут
?>
```

Якщо функція лише раз визначена в програмі, то перевизначити або видалити її пізніше не можна. Незважаючи на те, що імена функцій нечутливі до регістра, краще викликати функцію за допомогою того самого імені, під яким вона була задана у визначенні.

Наведемо приклад визначення функції усередині функції.

Приклад 6.4

```
<?php
/* не можна зберегти дані, тобто викликати
функцію DataSave() до того, як виконана
перевірка її правильності, тобто викликана
функція DataCheck() */

DataCheck();
DataSave();

function DataCheck(){
// перевірка правильності даних
  function DataSave(){
  // зберігаємо дані
  }
}
?>
```

Розглянемо докладніше аргументи функцій, їхнє призначення і використання.

6.2. Аргументи функцій

Кожна функція може мати список аргументів. За допомогою цих аргументів у функцію передається різна інформація (наприклад, значення числа, факторіал якого треба підрахувати). Кожен аргумент являє собою змінну або константу.

Дані у функцію можна передавати трьома різними способами: передача аргументів за значенням (використовується за умовчанням), посиланням і завдання значення аргументів за умовчанням.

Розглянемо ці способи докладніше.

Коли аргумент передається у функцію за значенням, зміна значення аргументу всередині функції не впливає на його значення поза функцією. Щоб дозволити функції змінювати її аргументи, їх потрібно передавати за посиланням. Для цього у визначенні функції перед ім'ям аргументу слід написати знак «&».

Наведемо приклад передачі аргументів за посиланням.

Приклад 6.5

```
<?php
// напишемо функцію, яка б додавала
// до рядка слово checked
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 відзначену кнопку
add_label($str);
    // викличемо функцію
echo $str ."><br>";
    // це виведе вже відзначену
    // кнопку
?>
```

У функції можна встановити значення аргументів, використовувані за умовчанням. Це значення повинно бути константним виразом, а не змінною і не представником класу або викликом іншої функції.

Наприклад, є функція, що створює інформаційне повідомлення, підпис до якого міняється залежно від значення переданого їй параметра. Якщо значення параметра не задано, то використається напис "Оргкомітет".

Наведемо приклад завдання значення аргументів за умовчанням.

Приклад 6.6

```
<?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.
    // У цьому випадку підпис - це Оргкомітет
Message("З повагою Семен");
    // У цьому випадку підпис
    // буде "З повагою Семен."
?>
```

Результатом роботи цієї програми буде:

```
Наступні збори відбудуться завтра.
Оргкомітет.
Наступні збори відбудуться завтра.
З повагою Семен.
```

Якщо у функції кілька параметрів, то ті аргументи, для яких задаються значення за умовчанням, повинні бути записані після всіх інших аргументів у визначенні функції. У противному разі з'явиться помилка, якщо ці аргументи будуть опущені за викликом функції.

Наприклад, необхідно внести опис статті в каталог. Користувач повинен увести такі дані: назву, прізвище автора і короткий опис статті. Якщо користувач не вводить прізвище автора статті, вважаємо, що це Петренко Петро.

Приклад 6.7

```
<?php
function Add_article($title, $description,
    $author=" Петренко Петро "){
  echo "Заносимо в каталог статтю: $title,";
  echo "автор $author";
  echo "<br>Короткий опис: ";
  echo "$description <hr>";
}
Add_article("Особливості використання теорії графів для
оптимізації топології інформаційної мережі вугільної шахти",
   "Це стаття про використання теорії графів ...",
   "Цвіркун Леонід");
Add_article("Сучасні проблеми розвитку Інтернет", "Це стаття про
Інтернет ...");
?>
```

У результаті роботи програми одержимо таке:

```
Заносимо в каталог статтю: Особливості використання теорії графів
для  оптимізації топології інформаційної мережі вугільної шахти,
   автор Цвіркун Леонід.
Короткий опис:
   Це стаття про використання теорії графів...

Заносимо в каталог статтю: Сучасні проблеми розвитку Інтернету,
   автор Петренко Петро.
Короткий опис:
   Це стаття про Інтернет...
```

Можна також написати інакше.

Приклад 6.8

```
<?php
function Add_article($author=" Петренко Петро ",
    $title, $description){
// ...дії, як у попередньому прикладі
}
Add_article("Сучасні проблеми розвитку Інтернету ",
    "Це стаття про Інтернет...");
?>
```

Результатом роботи цієї програми буде:

```
Warning: Missing argument 3 for
    add_article() in
c:\users\nina\tasks\func\def_bad.php
    on line 2
```

Мовою PHP 4.0 можна створювати функції зі змінним числом аргументів. Іншими словами, створювати функцію, не знаючи заздалегідь, зі 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().

Результат роботи функції func_num_args() – це визначення числа аргументів, які передані у поточну функцію. Ця функція може використовуватися тільки всередині визначення користувацької функції. Якщо вона з'явиться поза функцією, то інтерпретатор видасть попередження.

Наведемо приклад використання функції func_num_args().

Приклад 6.9

```
<?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 (номер_аргументу) повертає аргумент зі списку переданих у функцію аргументів, порядковий номер якого заданий параметром номер_аргументу. Аргументи функції нумеруються, починаючи з нуля. Як і func_num_args(), ця функція може використовуватися тільки всередині визначення якої-небудь функції.

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

Створимо функцію для перевірки типу даних її аргументів. Вважатимемо, що перевірка пройшла успішно, якщо перший аргумент функції – ціле число, другий – рядок.

Наведемо приклад функції для перевірки типу даних її аргументів.

Приклад 6.10

```
<?
function DataCheck(){
  $check =true;
  $n = func_num_args();
     // число аргументів,
     // переданих у функцію
  /* перевіряємо, чи є перший
```

```
  переданий аргумент цілим числом */
  if ($n>=1) if (!is_int(func_get_arg(0)))
      $check = false;
  /* перевіряємо, чи є другий
  переданий аргумент рядком */
  if ($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
   умови<br>";
if (DataCheck(324))
   echo "Перевірка пройшла успішно<br>";
else echo "Дані не задовольняють умови<br>";
?>
```

Результатом роботи буде такий текст:

```
Перевірка пройшла успішно
Перевірка пройшла успішно
```

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 користувацької функції, то генерується попередження.

Перепишемо попередній приклад, використовуючи цю функцію. Будемо перевіряти, чи є цілим числом кожен парний аргумент, переданий функції.

Приклад 6.11

```
<?
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(array("text", 324)))
   echo "Перевірка пройшла успішно<br>";
else echo "Дані не задовольняють
   умови<br>";
?>
```

Як бачимо, комбінації функцій func_num_args(), func_get_arg() і func_get_args() використовуються для того, щоб функції могли мати змінний список аргументів. Ці функції були додані тільки в мову PHP 4.0. У мові PHP 3.0 для того, щоб домогтися подібного ефекту, можна використати як аргумент функції масив.

Наведемо приклад програми, яка перевіряє, чи є кожен непарний параметр функції цілим числом.

Приклад 6.12

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

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

6.3. Використання змінних всередині функції

6.3.1. Глобальні змінні

Щоб використати всередині функції змінні, задані поза нею, ці змінні потрібно заявити як глобальні. Для цього в тілі функції варто перелічити їхні імена після ключового слова global.

Приклад 6.13

```
global $var1, $var2;
<?
```

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

У результаті роботи цієї програми одержимо:

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

Коли змінна заявляється як глобальна, фактично створюється посилання на глобальну змінну. Тому буде такий запис (масив GLOBALS містить всі змінні, глобальні відносно поточної області видимості):

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

Це значить, наприклад, що видалення змінної $var1 не видаляє глобальну змінну $_GLOBALS["var1"].

6.3.2. Статичні змінні

Щоб використати змінні тільки всередині функції, при цьому зберігаючи їхнє значення і після виходу з функції, потрібно заявити ці змінні як статичні. Статичні змінні помітні тільки всередині функції, вони не доступні, якщо виконання програми виходить за межі функції. Заявлення таких змінних проводиться за допомогою ключового слова static:

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

Статичній змінній може бути присвоєне будь-яке значення, але не посилання.

Наведемо приклад використання статичної змінної.

Приклад 6.14

```
<?
function Test_s(){
static $a = 1;
// не можна присвоювати вираз або посилання
  $a = $a*2;
  echo $a;
```

```
}
Test_s(); // виведе 2
echo $a;  // нічого не виведе, тому що $a доступна
          // тільки всередині функції
Test_s(); // усередині функції $a=2, тому результатом
          // роботи функції буде число 4
?>
```

6.4. Результати роботи функції

6.4.1. Значення

Всі функції, наведені вище як приклади, виконували які-небудь дії.

Крім подібних дій, будь-яка функція може повертати як результат своєї роботи яке-небудь значення. Це робиться за допомогою команди return. Цей результат може бути будь-якого типу, включаючи списки й об'єкти. Коли інтерпретатор зустрічає команду return у тілі функції, він негайно припиняє її виконання і переходить на той рядок, з якого була викликана функція.

Наприклад, складемо функцію, результатом роботи якої буде визначення віку людини. Якщо людина не вмерла, то вік визначається згідно з поточним роком.

Приклад 6.15

```
<?php
/* якщо другий параметр обчислюється
як true, те 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); // виведе 33
echo Age(1971, 2001); // виведе 30
?>
```

У цьому прикладі можна було і не використовувати функцію return, а просто замінити її функцією echo. Однак якщо все-таки робити так, що функція повертає якесь значення (у цьому випадку вік людини), то в програмі можемо присвоїти будь-якій змінній значення цієї функції:

```
$my_age = Age(1981, 2004);
```

У результаті роботи функції може бути повернуте тільки одне значення. Кілька значень можна одержати, якщо повертати список значень (одномірний масив). Наприклад, необхідно одержати повний вік людини з точністю до дня.

Приклад 6.16

```
<?php
function Full_age($b_day, $b_month, $b_year)
```

```
{
        $y = date("Y");
        $m = intval(date("m"));
        $d = intval(date("d"));
        $b_month = intval($b_month);
        $b_day = intval($b_day);
        $b_year = intval($b_year);

        $day = ($b_day > $d ? 30 - $b_day + $d : $d - $b_day);
        $tmpMonth = ($b_day > $d ?  -1:0);
        $month = ($b_month > $m + $tmpMonth ? 12 - $b_month +
          $tmpMonth + $m : $m+$tmpMonth - $b_month);
        $tmpYear  = ($b_month >  $m + $tmpMonth ?  -1:0);
        if ($b_year > $y + $tmpYear)
        {
                $year = 0; $month = 0; $day = 0;
        }
        else
        {
                $year = $y + $tmpYear - $b_year;
        }
        return array ($day,$month,$year);
}
$age = Full_age("29","06","1986");
echo "Вам $age[2] роки, $age[1] місяців і $age[0] днів";
     ?>
```

Коли функція повертає кілька значень для їхньої обробки в програмі, зручно використати мовну конструкцію list(), що дозволяє однією дією присвоїти значення відразу декільком змінним.

Так, у попередньому прикладі 6.16 можна було, залишивши без зміни функцію, обробити значення, що вона повертає, як показано далі у прикладі.

Приклад 6.17

```
<?
// 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.

Приклад 6.18

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

```
echo $a," ",$b;
   // виведе рядок «first second»
?>
```

6.4.2. Посилання

У результаті своєї роботи функція також може повертати посилання на яку-небудь змінну. Це може стати в пригоді, якщо потрібно використати функцію для того, щоб визначити, якій змінній повинно бути присвоєне посилання. Щоб одержати з функції посилання, потрібно при заявленні перед її ім'ям написати знак (&) і щоразу за викликом функції перед її ім'ям також писати (&). Звичайно функція повертає посилання на яку-небудь глобальну змінну (або її частину – посилання на елемент глобального масиву), на статичну змінну (або її частину) або на один з аргументів.

Приклад 6.19

```
<?
$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 у змінну $var цього прикладу не копіюється значення змінної $b повернутою функцією $ref, а створюється посилання на цю змінну. Таким чином, змінні $var і $b ідентичні й будуть змінюватися одночасно.

6.5. Функції як змінні

Мова PHP підтримує концепцію функцій як змінних. Це значить, що коли ім'я змінної закінчується круглими дужками, то інтерпретатор PHP шукає функцію з таким самим ім'ям і намагається її виконати.

Приклад 6.20

```
<?
/* створимо дві прості функції:
Add_sign - додає підпис до рядка й
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,
   // що дорівнює імені функції Show_text
$func();
   // це викличе функцію Show_text
$func = "Add_sign";
   // створюємо змінну зі значенням,
   // що дорівнює імені функції Add_sign
$func("Привіт усім <br>");
   // це викличе функцію
   // Add_sign з параметром "Привіт усім"
?>
```

У цьому прикладі функція Show_text виводить рядок тексту. Здавалося б, навіщо для цього створювати окрему функцію, якщо існує спеціальна функція echo(). Справа в тому, що такі функції, як echo(), print(), unset(), include() і т.п., не можна використовувати як змінні, тобто якщо написати,

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

то інтерпретатор PHP виведе помилку:

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

Отже, для того щоб використовувати кожну з наведених вище функцій як змінну, потрібно створити власну функцію, що було зроблено у попередньому прикладі.

6.6. Вирішення завдання

Як приклад завдання створимо веб-інтерфейс для генерації html-форми. Користувач зможе вибрати, не застосовуючи програмування, елементи форми і необхідну кількість характеристик цих елементів, а також їх назви, при цьому програма буде генерувати потрібну форму.

Розіб'ємо завдання на декілька підзадач: вибір типів елементів введення і їхньої кількості, створення назв елементів введення та обробка отриманих даних, тобто безпосередня генерація форми.

Перша підзадача досить проста: потрібно написати програму для завдання відповідної форми.

Наведемо приклад програми task_form.html.

Приклад 6.21

```
<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 в імені елемента форми пишеться, наприклад types[], це значить, що його ім'я – наступний елемент масиву types. Іншими словами перший елемент форми ("рядок введення тексту") буде мати ім'я types[0], а другий (текстова область) – types[1].

Форма task_form.html буде виглядати так:

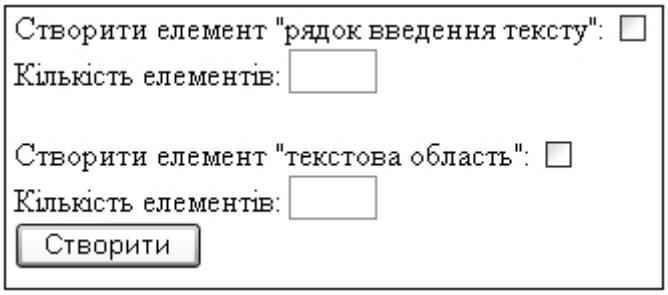


Рис. 6.1. Форма для вибору створюваних елементів і їх кількості

Після відправлення даних цієї форми одержимо інформацію про те, які елементи і скільки елементів кожного типу потрібно створити. Наступна програма запитує назви для цих елементів. Наведемо приклад програми ask_names.php

Приклад 6.22

```
<?
/* Файл, що буде обробляти форму */
$file = "task.php";

// Функція генерує форму для введення назв елементів введення
function Ask_names(){
```

```
// Повідомляємо, що хочемо використати цю
// змінну, задану поза функцією
global $file;
if (isset($_GET["types"])){
  $st = '<form action="'.$file.'">';
  foreach ($_GET["types"] as $k => $type){
/* перебираємо всі типи елементів,
   які потрібно створити */

  $num = $_GET["numbers"][$type];
     // скільки елементів кожного типу потрібно
  for ($i=1;$i<=$num;$i++){
     // створюємо $num рядків для введення
    $st.= "Уведіть ім'я $i-го елемента 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 змінної $st утримується код форми
   // для запиту імен
} else echo "Select type";
}
echo Ask_names();
   // викликаємо функцію і виводимо
   // результати її роботи
?>
```

Наприклад, потрібно створити два елементи типу «текстовий рядок» і один елемент типу «текстова область», як і відзначено у формі вище. Тоді програма ask_names.php обробить її таким чином, що буде одержана форма, як на рис. 6.2.

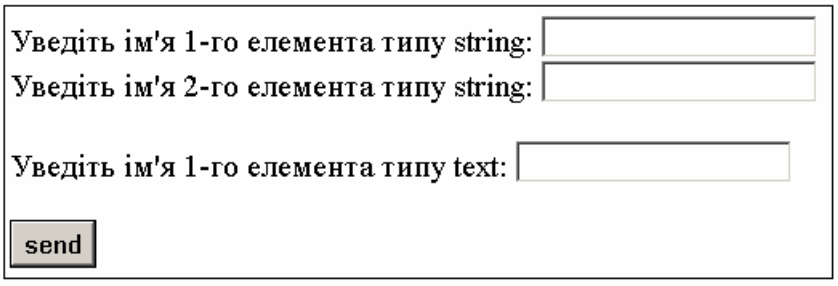


Рис. 6.2. Форма для введення назв створюваних елементів

Введемо в цю форму, наприклад, рядки «Назва», «Автор» і «Короткий зміст» (рис. 6.3).

Ці дані буде обробляти програма task.php.

Приклад 6.23

```
<?
// Файл, що буде обробляти дані створеної цим файлом форми
$show_file = "task_show.php";

// Функція створює елемент введення за типом і назвою
function Create_element($type,$name){
  $str="";
  switch($type){
  case "string":
    $str .= "$name: <input type=text name=string[]><br>";
  break;
  case "text":
    $str .= "$name: <textarea name=text[]></textarea><br>";
  break;
  }
return $str;
}
// Функція створює форму з потрібними елементами
function Create_form(){
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++){
     // Ім'я створюваного елемента
      $arr = $_GET["names"][$type][$i-1];

     // Викликаємо функцію для створення елемента
     $str .= Create_element($type,$arr);
    }
  }
  $str .= "<input type=submit value=send></form>";
echo $str;
}
$crt = "Create_form";
// Викликаємо функцію створення форми Create_form
$crt();
?>
```

Результатом роботи цієї програми з вхідними даними, наведеними вище, буде форма, подана далі (рис. 6.4).

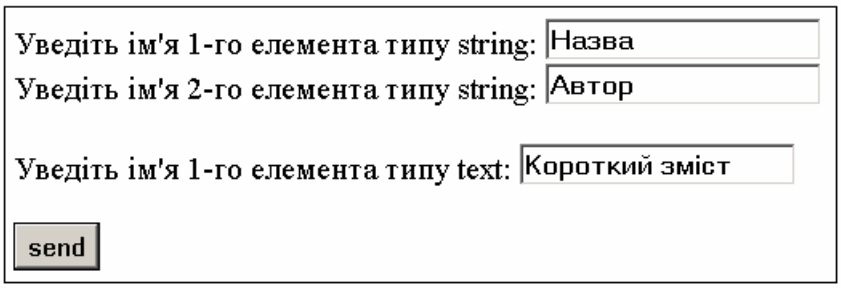


Рис. 6.3. Введення у форму назв створюваних елементів

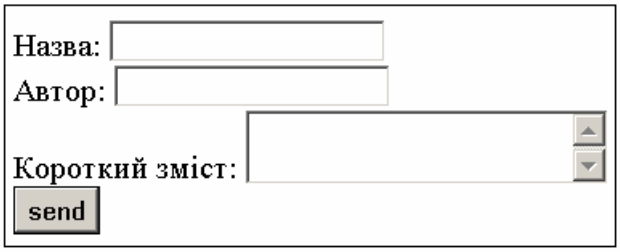


Рис. 6.4. Приклад отриманої форми

Висновки

У даному розділі розглянуто такі основні питання:

- визначення функції, їх синтаксис і семантика;
- способи передачі аргументів у функції;
- принципи роботи зі змінним числом аргументів у функціях.

Контрольні питання

1. Для чого потрібні функції?
2. Що таке аргументи функцій?
3. Що таке передача аргумента за посиланням?
4. Як створити функцію змінної довжини?
5. Як використати глобальні змінні у функції?
6. Що таке статичні змінні? Як їх використовують?
7. Яку роль у функціях грає команда return?
8. Що таке функції як змінні?
9. Що таке результат роботи функціїї?
10. Які є внутрішні функції мови PHP?

7. КЛАСИ ТА ОБ'ЄКТИ

Навчальною метою розділу є ознайомлення студентів з класами та об’єктами у мові PHP.

У результаті вивчення даного розділу студенти повинні знати:

– поняття класу;
– механізми успадковування;
– конструктори та оператори класів.

7.1. Поняття класу та об’єкта

Почнемо з основних понять об’єктно-орієнтованого програмування – класу та об'єкта. Існує безліч визначень цих понять.

Дамо таке визначення: об'єкт – це структурована змінна, яка утримує всю інформацію про деякий фізичний предмет або реалізовані в програмі поняття, клас – це опис таких об'єктів і дій, які можна з ними виконувати.

У мові PHP клас визначається за допомогою такого синтаксису:

```
class Ім'я_класу{
  var $ім'я_властивості;
    /* список властивостей */
  function ім'я_методу( ){
   /* визначення методу */
  }
  /* список методів */
}
```

Імена властивостей об'єктів класу визначаються за допомогою ключового слова var, методи, застосовні до об'єктів даного класу, описуються функціями. Усередині визначення класу можна використовувати ключове слово this для звертання до поточного представника класу.

Наприклад, потрібно створити клас, що описує категорію статей. У кожної статті є такі властивості, як назва, автор і короткий зміст.

Які дії необхідно робити зі статтями?

Можливо, знадобиться задавати значення перерахованим властивостям статті, відображати статтю в браузері. Тоді визначення цього класу може виглядати, як показано далі у прикладі.

Приклад 7.1

```
<?
// Створюємо клас статей
class Articles {
  var $title;
  var $author;
  var $description;

  // Метод, що присвоює значення атрибутам класу
```

```
  function make_article($t, $a, $d){
    $this->title = $t;
    $this->author = $a;
    $this->description = $d;
  }

  // Метод для відображення екземплярів класу
  function show_article(){
    $art = $this->title . "<br>" .
           $this->description .
      "<br>Автор: " . $this->author;
    echo $art;
  }
}
?>
```

Отже, для того щоб описати фізичні об'єкти типу «стаття», необхідно створити клас з ім'ям Articles, що складається із трьох змінних, отримуючих характеристики статті, і двох функцій для створення конкретної статті та її відображення.

Як відомо, використовуючи мову PHP, можна періодично перемикатися в режим HTML. У цьому випадку програма складається з декількох блоків коду. Визначення класу не можна розносити по різних блоках php-коду і тим більше по різних файлах, тобто якщо написати так:

```
<?php

// Початок класу
class Articles {
   var $title;
?>

<?php
// Продовження опису класу
   function show_article(){
   // зміст методу
   }
}
// Кінець опису класу

?> ,
```

то ця програма не буде працювати коректно.

Кілька зауважень з приводу імен класів.

Ім'я класу повинно задовольняти правила іменування об'єктів мовою PHP, але є ряд імен, які зарезервовані розробниками для своїх цілей. У першу чергу це імена, що починаються із символу підкреслення «_». Для створення класів і функцій не можна застосовувати такі імена. Крім того, зарезервоване ім'я stdClass, оскільки воно використається безпосередньо мовою PHP.

7.2. Ініціалізація змінних

Часто деяким атрибутам класу буває необхідно присвоювати значення відразу після створення представника класу. Коли було створено клас статей, для присвоювання значень атрибутам (властивостям) класу використовували спеціальну функцію make_article().

Загалом кажучи, це не зовсім правильно. Спеціально для завдання початкових значень атрибутам класу існує два стандартних методи.

Мова PHP 4.0 дозволяє проводити ініціалізацію за допомогою оператора var або функції конструктора, при чому var допоможе проініціалізувати тільки константні значення. Для завдання не константних значень використовують функцію конструктор, що викликається автоматично, коли об'єкт конструюється із класу. Функція-конструктор повинна мати таке саме ім'я, що й ім'я класу, у якому вона визначена.

Наведемо приклад.

Припустімо, при створенні об'єкта «стаття» необхідно встановити її властивості в такий спосіб: змінну «автор» зрівняти з рядком «Іванов», назву і короткий зміст – з відповідними елементами глобального масиву $_POST, а дату публікації статті – з поточною датою. Тоді такий опис класу не є коректним мовою PHP 4.0:

```
<?
class Articles { // Створюємо клас статей
  var $title= $_POST["title"];
  var $author = "Петренко";
  var $description = $_POST["description"];
  var $published = date("Y-m-d");
// Метод, що присвоює значення атрибутам класу
  }
?>
```

А от такий опис класу в PHP 4.0, що наведений у прикладі 7.2, буде працювати так, як треба.

Приклад 7.2

```
<?
class Articles { // Створюємо клас статей
  var $title;
  var $author = "Петренко";
  var $description;
  var $published;
// Метод, що присвоює значення атрибутам класу
function Articles(){
  $this->title = $_POST["title"];
  $this->description = $_POST["description"];
  $this ->published = date("Y-m-d");
 }
}
?>
```

Відзначимо, що у версіях мови PHP 3.0 і 4.0 конструктори працюють по-різному. У мові PHP 3.0 функція ставала конструктором, якщо вона мала те саме ім'я, що і клас, а у мові PHP 4.0 – якщо вона мала те саме ім'я, що і клас, у якому вона визначена.

Різниця в підходах видна, коли один клас розширює інший і відбувається спадкування властивостей і методів базового класу.

У мові PHP 5.0 конструктор класу називається _construct. Крім того, у мові PHP 5.0 з'явилися і деструктори – функції, які викликаються автоматично перед знищенням об'єкта. У мові PHP 5.0 функція-деструктор повинна бути названа _destruct.

7.3. Об'єкти

Раніше описувалося існування у мові PHP такого типу даних, як об'єкт. Клас – це опис даних одного типу, даних типу об'єкт. Класи є як би шаблонами для реальних змінних. Змінна потрібного типу створюється із класу за допомогою оператора new. Створивши об'єкт, ми можемо застосовувати до нього всі методи й одержувати всі властивості в описі класу. Для цього використовують такий синтаксис:

```
$ім'я_об'єкта->назва_властивості
або
$ім'я_об'єкта->назва_методу(список аргументів).
```

Помітимо, що перед назвою властивості або методу знак $ не ставлять.

Приклад 7.3

```
<?php
// Створюємо об'єкт $art
$art = new Articles;

// Виводимо назву об'єкта $art
echo ($art->title);

// Створюємо об'єкт $another_art
$another_art = new Articles;

// Викликаємо метод для відображення об'єкта в браузер
$another_art->show_article();
?>
```

Кожний з об'єктів класу має ті самі властивості й методи.

Так, в об'єктів $art і $another_art є властивості title, description, author і методи Articles(), show_article(). Але це два різних об'єкти.

Уявимо собі об'єкт як директорію у файловій системі, а його характеристики – як файли в цій директорії. Очевидно, що в кожній директорії можуть лежати однакові файли, але проте вони вважаються різними, оскільки зберігаються в різних директоріях.

Так само властивості й методи вважаються різними, якщо вони застосовуються до різних об'єктів. Щоб одержати потрібний файл з директорії верхнього рівня, ми пишемо повний шлях до цього файлу. Для роботи з класами необхідно вказувати повне ім'я функції, яку потрібно викликати.

Директорією верхнього рівня для мови PHP буде простір глобальних змінних, а шлях указується за допомогою роздільника ->.

Таким чином, імена $art->title і $another_art->title позначають дві різні змінні. Змінна мови PHP має тільки один знак долара перед ім'ям, тому не можна писати $art->$title. Ця конструкція буде розглянута не як звертання до властивості title об'єкта $art, а як звертання до властивості, ім'я якої задано змінною $title (наприклад $art->"").

Приклад 7.4

```
<?php

// Так можна встановити значення властивості об'єкта
$art->title = "Вступ до Інтернету";

// Так не можна встановити значення властивості об'єкта
$art->$title = "Вступ до Інтернету";

$property = "title";
$art->$property = "Вступ до Інтернету";
    // так можна встановити значення
    // властивості об'єкта
?>
```

Коли створюють клас, можна не знати, яке ім'я буде мати об'єкт цього класу. Тим більше, що об'єктів може бути багато і всі можуть мати різні імена. Відповідно не будемо знати, як звертатися до об'єкта всередині визначення класу. Для того щоб мати доступ до функцій і змінних всередині визначення класу, потрібно використовувати змінну $this.

Наприклад, $this->title дозволяє отримати значення властивості title у поточного об'єкта даного класу. Іноді цю змінну пропонують читати як «моє власне» (стосовно властивості).

7.4. Успадкування класів

7.4.1. Механізм успадкування

Механізм успадкування – дуже важлива частина всього об’єктно-орієнтованого підходу. Спробуємо пояснити його суть на прикладі.

Припустімо, що створюємо опис людини. Очевидно, що зробити це можна по-різному, залежно від того, для чого потрібен цей опис.

Наприклад, описати людину як програміста: вона знає мови програмування, операційні системи, брала участь у проектах. Однак якщо людина за фахом програміст, то вона не перестає бути людиною взагалі, тобто вона має ім'я, прізвище, місце проживання і т.д.

Якщо перевести ці міркування в терміни об’єктно-орієнтованого програмування, то можна сказати, що описано два класи – клас людей і клас програмістів, кожний зі своїми властивостями і методами. Причому клас програмістів, мабуть, має всі властивості класу людей і при цьому має свої специфічні характеристики, тобто клас програмістів є підкласом класу людей.

Отже, якщо у людини є ім'я, то в програміста воно теж повинне бути, але не навпаки. Крім програмістів, можна виділити ще безліч класів відповідно до професійної приналежності людей. І всі вони будуть підкласами класу людей.

Часто на практиці зручно визначати загальний клас, що може використовуватися відразу в декількох проектах (наприклад, клас людей або особистостей), і адаптувати його для специфічних потреб кожного проекту (наприклад, як клас програмістів).

Це можна реалізувати за допомогою механізму розширень. Будь-який клас може бути розширенням іншого класу. Розширювальний (або похідний) клас, крім тих властивостей і методів, які описані в його визначенні, має всі функції і властивості основного (базового класу).

У нашому прикладі клас програмістів – розширений, а клас усіх людей – базовий. Із класу не можна видалити ніякі існуючі властивості та функції, клас можна тільки розширити. Розширювальний клас мови PHP 4.0 завжди залежить тільки від одного базового класу, оскільки множинне успадкування мова PHP не підтримує.

У мові PHP класи розширюються за допомогою ключового слова extends.

Приклад 7.5

```
<?php

// Визначаємо клас особи
class Person {
  // Ім'я особи
  var $first_name;

  // Прізвище особи
  var $last_name;

  // Метод установлює значення імені й прізвища особи
  function make_person($t,$a){
    $this->first_name = $t;
    $this->last_name = $a;
  }

  // Метод відображає інформацію особи
  function show_person(){
    echo ("<h2>" . $this->first_name . " " .
      $this->last_name . "</h2>");
  }
}
```

```
// Визначаємо клас Programmer, що розширює Person
class Programmer extends Person {
  // Константний масив буде зберігати мови програмування
  var $langs = array ("Lisp");

  // Метод додає ще одну мову до списку відомих
  function set_lang($new_lang) {
    $this->langs[] = $new_lang;
  }
}
?>
```

Клас Programmer має такі самі змінні й функції, що і клас Person, плюс змінну $langs, у якій зберігається список вивчених програмістом мов, і функцію set_lang для додавання ще однієї мови до списку вивчених.

Створюють представника класу програмістів звичайним способом за допомогою конструкції new. Після цього можна встановлювати й одержувати список мов, які знає програміст, використовувати функції, які задані для класу Person, тобто встановлювати й одержувати ім'я і прізвище програміста та відображати відомості про нього в браузері, наприклад:

```
<?php

$progr = new Programmer;
$progr->set_lang("PHP");
print_r ($progr->langs);
$progr->make_person("Bill", "Gates");
$progr->show_person();

?>
```

Відношення між створеними класами Person і Programmer, називають також відношеннями батько-нащадок. Клас Person – батько, а його нащадки – такі як клас Programmer, створюються за допомогою розширень. Будь-який клас може стати батьківським і відповідно породити нащадків.

Порядок визначення класів – дуже важливий. Не можна спочатку визначити клас Programmer, що розширює клас Person, а вже потім сам клас Person. Клас повинен бути визначений перед тим, як він буде використовуватися (розширюватися).

7.4.2. Відмінності успадкування у версіях мови PHP 3.0 та 4.0

Тепер, після знайомства з механізмом успадкування мови PHP, можемо розглянути розходження між конструкторами версій мов PHP 3.0 та 4.0 і більш докладно розповісти про конструктори взагалі.

Нагадаємо, що для мови PHP 3.0 конструктор – це функція, ім'я якої збігається з ім'ям класу. А для мови PHP 4.0 – функція, ім'я якої збігається з ім'ям класу, у якому вона визначена.

Приклад 7.6

```
<?php
// Визначаємо клас Programmer, що розширює Person
class Programmer extends Person {
  var $langs = array ("Lisp");
  function Programmer(){
    // Цей конструктор буде працювати і в PHP 3.0, і в PHP 4.0
    $this->make_person("Семен","Петренко");
  }
}
?>
```

Тут функція Programmer() є конструктором, тобто виконується відразу після створення будь-якого представника класу Programmer, задаючи йому ім'я «Тарас» і прізвище «Кузьменко».

Конструктори, як і будь-які інші функції, можуть мати аргументи. У цьому випадку, створюючи представника класу, потрібно вказати значення цих параметрів. Аргументи конструктора можуть мати і наперед задані значення. Якщо всі аргументи мають такі значення, тоді можна створювати екземпляр класу без параметрів.

Приклад 7.7

```
<?php
// Визначаємо клас Programmer, що розширює Person
class Programmer extends Person{
  var $langs = array ("Lisp");

  // Це конструктор
  function Programmer($n = "Семен", $f = "Петренко") {
    $this->make_person($n,$f);
  }
}

// Створити програміста Семена Петренка
$default_progr = new Programmer();

// Створити програміста Тараса Кузьменка
$new_progr = new Programmer("Тарас", "Кузьменко");

// Виведе інформацію про змінну $new_progr
// тобто властивості об'єкта і їх значення

print_r($new_progr);

?>
```

Наведене у прикладі буде працювати для версій мов PHP 3.0 і 4.0, звичайно якщо дописати в них визначення базового класу Person.

Розглянемо у прикладі трохи іншу ситуацію: конструктор є тільки в базовому класі Person.

Приклад 7.8

```
<?php

// Визначаємо клас особи
class Person {
  var $first_name;
  var $last_name;

  // Конструктор
  function Person($t,$a){
    $this->first_name = $t;
    $this->last_name = $a;
  }
  /*  ...  */
}

// Визначаємо клас Programmer, що розширює Person
class Programmer extends Person {
  var $langs = array ("Lisp");
  function set_lang($new_lang) {
    $this->langs[] = $new_lang;
  }
}
$new_progr = new Programmer("Тарас", "Кузьменко");
?>
```

Що відбудеться в цьому випадку при створенні об'єкта класу Programmer? Чи буде автоматично викликатися яка-небудь функція?

Для версії мови PHP 3.0 нічого не відбудеться, оскільки в цьому класі немає функції з ім'ям Programmer() (тут конструктор – це функція, ім'я якої збігається з ім'ям класу).

У PHP 4.0 буде викликаний конструктор базового класу, якщо він існує, тобто викличеться функція Person() із класу Person (тут конструктор – функція, ім'я якої збігається з ім'ям класу, у якому вона визначена).

Ще одна ситуація – у базовому класі є функція, ім'я якої збігається з ім'ям розширювального класу, а в розширювальному класі немає конструктора.

Приклад 7.9

```
<?php
// Визначаємо клас особи
class Person {
  var $first_name;
  var $last_name;

  // Конструктор
```

```
  function Person($t,$a){
    $this->first_name = $t;
    $this->last_name = $a;
  }
  function Programmer($new_lang){
    echo "Я - програміст";
  }
}

// Визначаємо клас Programmer, що розширює Person
class Programmer extends Person{
  var $langs = array ("Lisp");
  function set_lang($new_lang){
    $this->langs[] = $new_lang;
  }
}
$new_progr = new Programmer("Тарас", "Кузьменко");
?>
```

У цьому випадку мова PHP 3.0 викличе як конструктор функцію Programmer() з опису класу Person, оскільки конструктор – це функція, у якої таке саме ім'я, що і в класу. І неважливо, чи визначена ця функція в самому класі або вона успадковується з базового класу. У мові PHP 4.0 клас Programmer не буде мати свого конструктора, тому викличеться конструктор базового класу.

У жодній з цих мов (PHP 3.0, PHP 4.0) конструктор базового класу не викликається автоматично з конструктора розширювального класу.

7.4.3. Відмінності успадкування мови PHP 5.0

Крім нової назви для конструкторів і появи деструкторів, у мові PHP 5.0 відбулося ще досить багато змін.

Не будемо обговорювати їх докладно, тільки опишемо загалом.

Основна зміна – це передача значень параметрів класу за посиланням і присвоєння об'єктів за посиланням, а не за значенням, як це було у мові PHP 4.0.

Якщо у мові PHP 5.0 створюються дві однакові змінні типу об'єкта, то вони вказують на одне значення і змінюються одночасно (раніше був наведений схожий приклад зі змінними рядкового типу). У зв'язку з цим з'явився новий механізм для створення копій об'єктів – клонування.

У мові PHP 4.0 всі методи і змінні класу доступні ззовні, тобто вони завжди є відкритими. Для мови PHP 5.0 змінні й методи можна робити відкритими (доступними з будь-кого місця), закритими (доступними тільки всередині класу) і захищеними (доступними всередині класу і в його похідних класах).

Крім того, з'явилася можливість створювати інтерфейси й абстрактні класи і багато чого іншого.

В цілому об'єктна модель мови PHP 5.0 значно вдосконалена для більш точної відповідності об’єктно-орієнтованій парадигмі програмування.

7.5. Спеціальний оператор «::»

Іноді всередині опису класу виникає необхідність послатися на функції або змінні з базового класу. Буває, що потрібно посилатися на функції в класі, жоден представник якого ще не створений. Як бути в такому випадку?

В PHP 4.0 для цього існує спеціальний оператор «::».

У прикладі 7.10 показано, як можна викликати в описі класу Programmer функцію show_name() з базового класу Person і функцію say_hello(), задану в описі класу Programmer, коли жоден об'єкт цього класу ще не був створений.

Приклад 7.10

```
<?php
// Визначаємо клас особи
class Person {
  var $first_name;
  var $last_name;

  // Конструктор
  function Person($t,$a){
    $this->first_name = $t;
    $this->last_name = $a;
  }

  // Метод відображає інформацію про особу
  function show_name(){
    echo ("Мене звуть, " .
          $this->first_name . " " .
          $this->last_name . "!<br>");
  }
}

// Визначаємо клас Programmer, що розширює Person
class Programmer extends Person{
  function set_lang($new_lang){
    // метод додає ще одну мову до списку відомих
    $this->langs[] = $new_lang;

    // Викликаємо функцію з базового класу
    Person::show_name();
    echo "Я знаю тепер ще і мову" . $new_lang;
  }

  function show_name() {
  echo ("Я програміст, " .
        $this->first_name . " " .
        $this->last_name . "!<br>");
  }

  function say_hello() {
    echo "Привіт!<br>";
  }
```

```
}
// Викликаємо функцію, коли жодного об’єкта класу ще не створено
Programmer::say_hello();

$new_progr = new Programmer("Семен","Петренко");
$new_progr->set_lang("PHP");
?>
```

У результаті роботи цієї програми одержимо таке:

```
Привіт!
Мене звуть Семен Петренко!
Я знаю тепер ще і мову PHP
```

За допомогою команди `Programmer::say_hello();` викликано функцію `say_hello` класу Programmer не як метод, застосовуваний до об'єкта даного класу. Цей метод не має змінних класу. Тому функції, які викликані до створення об'єкта, не можуть користуватися змінними класу і конструкцією `this`, але можуть користуватися локальними і глобальними змінними.

У визначенні класу `Programmer` перевизначено функцію `show_name()`, тому викликати функцію `show_name()` з базового класу `Person` можна тільки за допомогою оператора «::».

Загалом кажучи, усередині визначення класу можна викликати будь-які методи і властивості, що задані в його базовому класі за допомогою звичайного $this, якщо тільки породжений клас не перевизначає ці властивості й методи, як у прикладі 7.10.

У наведеному раніше прикладі, коли зверталися до базового класу, було використано його ім'я (писали `Person::show_name()`).

Це не зовсім зручно, тому що ім'я класу або ієрархія класів може змінитися, і тоді потрібно буде переписувати код усіх класів, тобто зробити використовувані в них імена відповідно до нової ієрархії.

Щоб уникнути подібної ситуації, замість імені базового класу потрібно використати ключове слово parent (наприклад, `parent::show_name()`). Parent посилається на клас, записаний після extends. Тому, якщо раптом ієрархія класів зміниться, досить буде внести зміни в імена, зазначені після extends в описах класів.

7.6. Вирішення завдання

Для прикладу вирішимо завдання автоматичної генерації (на вибір користувача) форм для введення опису статті або людини, а також відображення цих даних на сторінці браузера.

Спробуємо це зробити, використовуючи об’єктно-орієнтований підхід.

Для початку створимо форму, де користувач вибирає, що він хоче створити: опис статті або людини (точніше, це будуть дві форми):

```
<form action="task1.php">
Створити опис статті:
  <input type=submit name=art_create value="Create Article">
```

```
</form>
<form action="task1.php">
Створити опис особи:
  <input type=submit name=pers_create value="Create Person">
</form>
```

Тепер напишемо файл для обробки цих форм. У ньому створимо два класи – статті й особи. У кожного класу є два методи для ініціалізації його змінних і для відображення об'єктів даного класу.

Для вирішенні завдання будуть використані дві функції, вбудовані в мову PHP для роботи з класами та об'єктами.

Це функція get_class (об'єкт), що повертає ім'я класу, екземпляром якого є об'єкт, переданий їй як параметр, і функція get_class_vars (ім'я класу), що повертає масив усіх властивостей класу та їхніх наперед заданих значень.

Аналогічно можна одержати масив імен усіх методів класу: get_class_methods (ім'я класу).

Приклад 7.11

```
<?php
// Створюємо класи статей і осіб.
// Стаття має заголовок, автора і опис.
// Особа має ім'я, прізвище і e-mail

class Article {
  var $title;
  var $author;
  var $description;

  // Метод, що присвоює значення атрибутам класу
  function Article($t="Назва відсутня",
      $a="Автор відсутній",
      $d="Опис відсутній") {
    $this->title = $t;
    $this->author = $a;
    $this->description = $d;
  }

  // Метод для відображення екземплярів класу
  function show() {
    $art = "<h2>$this->title</h2><font
    size=-1>$this->description</font><p>Автор:
    $this->author</p>";
    echo $art;
  }
}

// Визначення класу особи
class Person {
  var $first_name;
  var $last_name;
```

```
  var $email;
// Метод, що присвоює значення атрибутам класу

  function Person($t = "Ім'я не введене",
    $a = "Прізвище не введене", $d = "E-mail не зазначений") {
    $this->first_name = $t;
    $this->last_name = $a;
    $this->email = $d;
  }

  // Метод для відображення екземплярів класу
  function show(){
    $art = "<h2>$this->first_name</h2><font
    size=-1>$this->last_name</font><p>Автор:$this->email</p>";
    echo $art;
  }
}

// Далі треба створити і відобразити екземпляр обраного класу
// Якщо була обрана стаття
if (isset($_GET["art_create"])){
  // Створюємо представника класу статей
  $art = new Article;

  // Які аргументи цього класу потрібно задати
  $art_vars = get_class_vars(get_class($art));

  // Виклик функції для створення форми
  Make_form($art,$art_vars,"art_create");

  // Якщо дані цієї форми відправлені, то викликаємо
  // функцію відображення форми
  if (isset($_GET["create_real"])){ Show_($art_vars); }
}

// Те саме, якщо була обрана особа
if (isset($_GET["pers_create"])){
  $art = new Person;
  $art_vars = get_class_vars(get_class($art));
  Make_form($art,$art_vars,"pers_create");
  if (isset($_GET["create_real"])){ Show_($art_vars); }
}

// Функція для створення форми
function Make_form($art,$art_vars,$glob){
  // html код форми записується в рядок $str
  $str = "<form>";

  // Перебираємо список змінних класу об'єкта $art
  foreach ($art_vars as $var_name => $var_value){
    // Створюємо елемент форми з ім'ям властивості класу
    $str .="$var_name<input type=text name=$var_name><br>";
  }
```

```
  // Щоб не забути, що ми створюємо
  $str .= "<input type=hidden name=$glob>";
  $str .= "<input type=submit name=create_real
  value='Create and Show'></form>";

// Виводимо форму
  echo "$str";
}

// Функція відображення об'єкта
function Show_($art_vars){

  // Використовується глобальне ім'я об'єкта
  global $art;

  // Число властивостей класу (змінних у формі)
  $k = count($art_vars);

  //Допоміжна змінна
  $p=0;

  foreach ($art_vars as $name => $value){
    $p++;
      if ($_GET["$name"]=="") $val= $art->$name;
      else $val = $_GET["$name"];
      if ($p<>$k) $par .='"'. $val.'",';
      else $par .='"'. $val.'"';
  }
  $par = '$art->'.$const ."(" .$par.");";

  // Тепер $par являє собою php-код для виклику
  // методу класу $art, записаного в $par
  // наприклад,
  // $art->Person('Vasia','Petrov','vas@intuit.ru');

  // Функція eval виконує код, який утримується в $par
  eval($par);
  $art->show();
}
?>
```

Висновки

У даному розділі розглянуто такі питання:

- основні поняття об’єктної моделі;
- правила створення класів;
- способи завдання початкових значень змінним класу;
- способи одержання значень властивостей і виклик методів класів.

Контрольні питання

1. Що таке клас?
2. Розкажіть про визначення класу?
3. З яких частин складається клас?
4. Яким чином можна виконати ініціалізацію змінних класу?
5. Що являє собою об’єкт?
6. Чи потрібно ставити знак $ перед назвою властивості або методу класу?
7. Яким чином виконується механізм успадкування?
8. Що таке конструктор класу?
9. Яким чином виконується посилання на функції або змінні з базового класу?
10. Коли використовується оператор parent?
11. Опишіть об’єктну модель мови PHP 5.0.

8. МАСИВИ

Навчальною метою розділу є ознайомлення студентів з масивами.

У результаті вивчення даного розділу студенти повинні знати:

– визначення масиву;
– базові операції з масивами;
– сортування елементів масиву.

Мова PHP надає безліч функцій для роботи з масивами даних. Розглянемо деякі з таких функцій і з їхньою допомогою вирішимо кілька прикладних завдань. Зокрема, буде розглянуто функції для пошуку елементів у масиві та для їх сортування, застосування створених користувачем функцій до всіх елементів масиву і розбиття масиву на частини.

Раніше розповідалося про те, як можна створити масив даних. Нагадаємо, що масив можна створити двома способами:

– за допомогою конструкції array

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

– безпосередньо задаючи значення елементам масиву

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

Наприклад, нам потрібно зберігати список документів, які будуть вилучені з бази даних. Правильно зберігати його у вигляді масиву, ключем при цьому буде ідентифікатор документа (його унікальний номер), а значенням – назва документа. Цей масив можна створити таким чином:

```
<?php
$del_items = array("10" => "Наука і життя", "12" =>
"Інформатика");

// Додаємо елемент у масив
$del_items["13"] = "Програмування мовою PHP";
?>
```

8.1. Складання та порівняння масивів

Масив – це тип даних, відповідно до яких повинні бути визначені операції.

Масиви можна складати і порівнювати.

Складають масиви за допомогою стандартного оператора «+». Загалом кажучи, цю операцію точніше назвати об'єднанням масивів. Якщо у нас є два масиви, $a і $b, то результатом їх об'єднання буде масив $c, що складається з елементів $a, до яких праворуч дописані елементи масиву $b.

Причому, якщо зустрічаються збіжні ключі, то в результуючий масив включається елемент з першого масиву, тобто із $a. Таким чином, якщо складаються масиви у мові PHP, від зміни місць масивів сума міняється.

Приклад 8.1

```
<?
$a = array("i"=>"Інформатика", "м"=>"Математика");
$b = array("i"=>"Історія","б"=>"Біологія", "ф"=>"Фізика");
$c = $a + $b;
$d = $b +$a;
print_r($c);

/*
  Одержимо:
  Array([i]=>Інформатика [м]=>Математика [б]=> Біологія
*/
print_r($d);

/*
  Одержимо:
  Array([й]=>Історія [м]=>Біологія [ф]=>Фізика)
*/

?>
```

Порівнювати масиви можна, перевіряючи їхню рівність, нерівність, еквівалентність або нееквівалентність.

Рівність масивів – це збіг усіх пар ключ/значення елементів масивів.

Еквівалентність являє собою рівність значень і ключів елементів, а також запис елементів в обох масивах у тому самому порядку.

Рівність значень у мові PHP позначається символом «==», а еквівалентність – символом «===».

Приклад 8.2

```
<?php
$a = array("i" => "Інформатика", "м" => "Математика");
$b = array("м" => "Математика", "i" => "Інформатика");

if ($a == $b) echo "Масиви рівні між собою й";
else echo "Масиви не рівні і ";

if ($a === $b) echo "еквівалентні";
else echo "не еквівалентні";

// Одержимо echo "Масиви рівні між собою й не еквівалентні"

?>
```

8.2. Операції з масивами

8.2.1. Функція count

Для підрахунку кількості елементів масиву мова PHP має спеціальну функцію count().

Насправді ця функція обчислює число елементів у змінній взагалі.

Якщо застосувати її до будь-якої іншої змінної, вона поверне 1. Виняток становить змінна типу NULL – count(NULL) повертає 0.

Для застосування цієї функції до багатомірного масиву потрібно використати додатковий параметр COUNT_RECURSIVE.

Приклад 8.3

```
<?php
$del_items = array("langs" => array("10"=>"Python", "12"=>"Lisp"),
"other"=>"Інформатика");

// Виведе 2
echo count($del_items) . "<br>";

// Виведе 4
echo count($del_items, COUNT_RECURSIVE);
?>
```

8.2.2. Функція in_array

Вбудована функція in_array дозволяє встановити, чи міститься в заданому масиві шукане значення.

```
in_array("шукане значення","масив", ["обмеження на тип"]);
```

Якщо третій аргумент задати як true, то в масиві буде знайдений елемент, що збігається із шуканим не тільки за значенням, але й за типом.

Якщо шукане значення – рядок, то порівняння чуттєве до регістру.

Наприклад, є масив не вивчених нами мов програмування. Необхідно визначити, чи міститься в цьому масиві мова PHP.

Напишемо програму.

Приклад 8.4

```
<?php
$langs = array("Lisp", "Python", "Java", "PHP", "Perl");
// Виведе повідомлення "Треба вивчити мову PHP!"
if (in_array("PHP",$langs,true))
    echo "Треба б вивчити мову PHP!";
// Нічого не виведе, оскільки в масиві є рядок "PHP"
// а не "php"
if (in_array("php",$langs))
    echo "Треба вивчити php!";

?>
```

Шуканим значенням цієї функції може виступати і масив. Правда, ця властивість була додана тільки починаючи з версії мови PHP 4.2.0.

Наприклад:

```
<?php

$langs = array("Lisp", "Python", array("PHP","Java"), "Perl");
if (in_array(array("PHP","Java"),$langs))
  echo "Треба вивчити мови PHP і Java";

?>
```

8.2.3. Функція array_search

Це ще одна функція для пошуку значення в масиві. У результаті роботи array_search, на відміну від in_array, повертає значення ключа, якщо елемент знайдений, і false – у противному разі. А от синтаксис у цих функцій однаковий:

array_search ("шукане значення","масив", ["обмеження на тип"]);

Порівняння рядків чутливо до регістру, а якщо вказати відповідний аргумент, то порівнюються ще і типи значень. До версії мови PHP 4.2. 0, якщо шукане значення не було знайдене, результатом роботи функції було значення False або порожнє значення NULL.

Тепер, навпаки, нехай є масив вивчених мов програмування. Причому ключ кожного елемента – це номер, що вказує, якою за рахунком була вивчена ця мова.

Приклад 8.5

```
<?php

$langs = array("Lisp", "Python", "Java", "PHP", "Perl");
if (!array_search("PHP",$langs))
echo "Треба вивчити мову PHP!<br>";
else {
  $k = array_search("PHP", $langs);
  echo "Мову PHP я вивчив $k – ю!";
}
?>
```

У результаті одержимо рядок:

```
Мову PHP я вивчив 3-ю!
```

Очевидно, що ця функція більш функціональна, ніж in_array, оскільки можна одержати не тільки інформацію про те, що шуканий елемент у масиві є, але і визначити, де саме в масиві він знаходиться.

8.2.4. Функція array_keys

Розглянемо випадок, коли шуканих елементів у масиві декілька. Тоді функція array_search() поверне ключ першого зі знайдених елементів. Щоб одержати ключі всіх елементів, потрібно скористатися функцією array_keys().

Функція array_keys() вибирає всі ключі масиву. Але вона має додатковий аргумент, за допомогою якого можна одержати список ключів елементів з конкретним значенням. Синтаксис цієї функції такий:

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

Функція array_keys() повертає як строкові, так і числові ключі масиву, організовує всі значення у вигляді нового масиву із числовими індексами.

Наприклад, розблено масив мов, які вивчили.

Список буде довгим, і деякі мови будуть записані кілька разів. Перевіримо, чи стосується це мови Lisp.

Приклад 8.6

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

У результаті одержимо:

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

Функція array_keys() з'явилася тільки в PHP 4.0. У мові PHP 3.0 для реалізації її функціональності потрібно вигадувати свою функцію. Як і дві попередні, array_keys() залежить від регістра, отже, елементів LISP у масиві вона не знайде.

Якщо є функція для одержання всіх ключів масиву, то можна припустити, що існує і функція для одержання всіх значень масиву. Дійсно, вона є. Це функція array_values (масив).

Всі значення переданого їй масиву записуються в новий масив, проіндексований цілими числами, тобто всі ключі масиву втрачаються, залишаються тільки значення. Але повернемося до прикладу.

Отже, було з'ясовано, що мова Lisp випадково зазначена у масиві двічі. Оскільки вивчити одну мову двічі не можливо, то потрібно якось позбутися від дублікатів мов. Зробити це досить просто за допомогою функції array_unique().

8.2.5. Функція array_unique

Функція array_unique (масив) повертає новий масив, у якому повторювані елементи фігурують в одному екземплярі. Таким чином, замість декількох однакових значень і їхніх ключів ми маємо одне значення.

Який у нього буде ключ? Як з декількох ключів однакових елементів вибирається той, який буде збережений у новому масиві?

Функція виконує такі дії: всі елементи масиву перетворює в рядки і сортує, потім запам'ятовує перший ключ для кожного значення, а інші – ігнорує.

Спробуємо позбутися від повторюваних назв мов у списку вивчених.

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

На екрані з`явиться таке:

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

8.3. Сортування масивів

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

8.3.1. Функція sort

Функція sort має такий синтаксис:

sort (масив [, додатковий аргумент])

Функція сортує масив, тобто впорядковує його значення за зростанням. Вона видаляє всі існуючі в масиві ключі, заміняючи їх числовими індексами, що відповідають новому порядку елементів. У випадку успішного завершення роботи вона повертає true, у противному разі – false.

Нехай є два масиви: ціни товарів – їх назви й, навпаки, назви товарів – їх ціни. Упорядкуємо ці масиви за зростанням.

Приклад 8.7

```
<?php

$items = array(10 => "хліб", 20 => "молоко", 30 => "бутерброд");

// Рядки сортуються в алфавітному порядку, ключі губляться
sort($items);
print_r($items);
```

```
$rev_items = array("хліб" => 10, "бутерброд" => 30, "молоко" =>
20);

// Числа сортуються за зростанням, ключі губляться
sort($rev_items);
print_r($rev_items);

?>
```

Одержимо:

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

Як додатковий аргумент може бути використана одна з таких констант:
SORT_REGULAR – порівнювати елементи масиву звичайно;
SORT_NUMERIC – порівнювати елементи масиву як числа;
SORT_STRING – порівнювати елементи масиву як рядка.

8.3.2. Функції asort, rsort, arsort

Якщо потрібно зберігати індекси елементів масиву після сортування, то треба використати функцію asort (масив [, додатковий аргумент]).

Якщо необхідно відсортувати масив у зворотному порядку, тобто від найбільшого значення до найменшого, то можна задіяти функцію rsort (масив [, додатковий аргумент]).

А якщо при цьому потрібно ще і зберегти значення ключів, то слід використати функцію arsort(масив [, додатковий аргумент]).

Синтаксис у цих функцій абсолютно такий самий, як у функції sort. Відповідно і значення додаткового аргументу можуть бути такими ж, як в sort: SORT_REGULAR, SORT_NUMERIC, SORT_STRING.

До речі, функція SORT_NUMERIC з'явилася тільки у версії мови PHP 4.0.

Приклад 8.8

```
<?php
$books = array("Куліш Пантелеймон"=>"Січові гості Чуприна і
Чортоус","Гуцало Євген"=>"Мертва зона", "Нечуй-Левицький Іван"=>"
Запорожці", "Коцюбинський Михайло"=>"Коні не винні");
// Сортуємо масив, зберігаючи значення ключів
asort($books);

print_r($books);
echo "<br>";

// Сортуємо масив у зворотному порядку, ключі будуть замінені
rsort($books);
print_r($books);
?>
```

У результаті роботи цієї програми одержимо:

```
Array ( [Нечуй-Левицький Іван] => Запорожці
        [Коцюбинський Михайло] => Коні не винні
        [Гуцало Євген] => Мертва зона
        [Куліш Пантелеймон] => Січові гості Чуприна і Чортоус )
Array ( [0] => Січові гості Чуприна і Чортоус
        [1] => Мертва зона
        [2] => Коні не винні
        [3] => Запорожці )
```

Припустімо, необхідно створити каталог описів документів. У кожного документа є автор, назва, дата публікації і короткий зміст.

У попередніх розділах уже не раз розроблялися програми, які відображали описи, складені з цих характеристик. Щоразу порядок відображення цих елементів залежав від створеної програми. Тепер розробимо програму, яка дає можливість змінювати порядок відображення елементів за бажанням користувача.

Складемо для цього таку 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>
```

Будемо впорядковувати дані, передані цією формою, за спаданням їхніх значень, зберігаючи при цьому значення ключів. Для цього зручно скористатися функцією arsort(). Оскільки нам важливий тільки новий порядок елементів, збережемо в новому масиві ключі вихідного масиву в потрібному порядку, проте вони є іменами елементів, з яких конструюється опис документа, і це пам'ятати важливо.

Отже, одержуємо таку програму, наведену у прикладі 8.9.

Приклад 8.9

```
<?php
print_r($_GET);
echo "<br>";

// Сортуємо масив у зворотному порядку, зберігаючи ключі
arsort ($_GET);

print_r($_GET); echo "<br>";

// Складаємо новий масив
$ordered_names = array_keys($_GET);

// Виводимо елементи нового масиву
foreach($ordered_names as $key => $val) echo "$key :$val <br>";
?>
```

8.3.3. Сортування масиву за значенням ключів

Очевидно, що може виникнути необхідність у сортуванні масиву за значеннями ключів.

Наприклад, якщо є масив даних про книги, як у наведеному вище прикладі, то цілком імовірно, що потрібно буде відсортувати книги за іменами авторів.

Для цього у мові PHP також не потрібно писати багато рядків коду – можна просто скористатися функцією ksort() для сортування за зростанням (прямий порядок сортування) або krsort() – для сортування за спаданням (зворотний порядок сортування).

Синтаксис цих функцій знову таки аналогічний синтаксису функції sort().

Приклад 8.10

```
<?php
$books = array("Куліш Пантелеймон"=>"Січові гості Чуприна і
Чортоус","Гуцало Євген"=>" Мертва зона", "Нечуй-Левицький Іван"=>"
Запорожці", "Коцюбинський Михайло"=>"Коні не винні");

// Сортуємо масив, зберігаючи значення ключів
ksort($books);
print_r($books);
?>
```

Одержимо:

```
Array ( [Гуцало Євген] => Мертва зона
        [Коцюбинський Михайло] => Коні не винні
        [Куліш Пантелеймон] => Січові гості Чуприна і Чортоус
        [Нечуй-Левицький Іван] => Запорожці )
```

8.3.4. Сортування за допомогою функції, яка задана користувачем

Крім двох простих способів сортування значень масиву (за убуванням або за зростанням), мова PHP пропонує користувачеві можливість самому задавати критерії для сортування даних.

Критерії задаються за допомогою функції, ім'я якої вказується як аргумент для спеціальних функцій сортування usort() або uksort(). За назвами цих функцій можна здогадатися, що usort() сортує значення елементів масиву, а uksort() – значення ключів масиву за допомогою вказаної користувачем функції. Обидві функції повертають true, якщо сортування пройшло успішно, і false – у противному разі.

Їх синтаксис виглядає таким чином:

```
usort (масив, що сортує функція)
uksort (масив, що сортує функція)
```

Звичайно ж, не можна сортувати масив за допомогою будь-якої користувацької функції. Ця функція повинна задовольняти певні критерії, що дозволяють порівнювати елементи масиву.

Як повинна бути влаштована сортувальна функція?

По-перше, вона повинна мати два аргументи. У них інтерпретатор буде передавати пари значень елементів для функції usort() або ключів масиву для функції uksort().

По-друге, сортувальна функція повинна повертати:

– ціле число менше від нуля, якщо перший аргумент менше ніж другий;

– число, що дорівнює нулю, якщо два аргументи рівні між собою;

– число більше від нуля, якщо перший аргумент більше ніж другий.

Як і для інших функцій сортування, для функції usort() існує аналог, що не змінює значення ключів, – функція uasort().

Припустімо, є масив, що містить відомості про літературні твори, такі як назва, автор і рік видання. Необхідно впорядкувати книги за датою видання.

Приклад 8.11

```
<?php

// Масив виглядає так:
$books = array("Січові гості Чуприна і Чортоус" => array ("Куліш
Пантелеймон",1862), "Мертва зона" => array("Гуцало Євген",1967),
    "Запорожці" => array ("Нечуй-Левицький Іван",1874),
    "Коні не винні" => array("Коцюбинський Михайло",1912));

/*
  Можна, звичайно, переписати цей масив по-іншому,
  зробивши рік видання, наприклад, індексом,
  але набагато зручніше написати свою функцію для 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;
}
?>
```

У результаті одержимо:

```
Куліш Пантелеймон: "Січові гості Чуприна і Чортоус"
Нечуй-Левицький Іван: "Запорожці"
Коцюбинський Михайло: "Коні не винні"
Гуцало Євген: "Мертва зона"
```

У цьому прикладі використано власну функцію сортування до всіх елементів масиву. Далі розглянемо, як застосувати до елементів масиву будь-яку іншу користувацьку функцію.

8.4. Додаткові функції роботи з масивами

8.4.1. Функція array_walk

Функція array_walk застосовує створену користувачем функцію до всіх елементів масиву і повертає true у випадку успішного виконання операції і false – у противному разі.

```
array_walk(масив, функція [, дані])
```

Користувацька функція, як правило, має два аргументи, у які по черзі передаються значення і ключ кожного елемента масиву.

Але якщо при виклику функції array_walk() зазначений третій аргумент, то він буде розглянутий як значення третього аргументу користувацької функції, зміст якого визначає сам користувач.

Якщо функція користувача вимагає більше аргументів, ніж у неї передано, то при кожному виклику функції array_walk() буде видаватися попередження.

Якщо необхідно працювати з реальними значеннями масиву, а не з їхніми копіями, слід передавати аргумент у функцію за посиланням. Однак потрібно мати на увазі, що не можна додавати або видаляти елементи масиву і не можна також видаляти елементи масиву і змінювати сам масив, оскільки в цьому випадку результат роботи функції array_walk() вважається невизначеним.

Приклад 8.12

```
<?php
$books1 = array("Є. Гуцало."=>"Мертва зона",
  "М. Коцюбинський."=>"Коні не винні",
  "П. Куліш."=>"Січові гості Чуприна і Чортоус");

// Створюємо функцію, яку хочемо застосувати до елементів масиву
function try_walk($val,$key,$data){
   echo "$data \"$val\" написав $key<br>";
}

// Застосовуємо до всіх елементів масиву $books1 функцію try_walk
array_walk($books1,"try_walk","Оповідання");
?>
```

У результаті роботи програми одержимо:

```
Оповідання "Мертва зона" написав Є.Гуцало.
Оповідання "Коні не винні" написав М. Коцюбинський.
Оповідання "Січові гості Чуприна і Чортоус" написав П. Куліш.
```

Зауважимо, що значення елементів масиву не змінено. Щоб їх змінити, треба передати значення елементів у змінну $val функції try_walk за посиланням.

Приклад 8.13

```
<?php
$books1 = array
  "Є. Гуцало"=>"Мертва зона",
  "М. Коцюбинський"=>"Коні не винні",
  "П. Куліш"=>"Січові гості Чуприна і Чортоус");

// 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 всіх елементів массиву
// $book1 функцію try_walk
array_walk($books1,"try_walk");
print_r($books1);
?>
```

У результаті роботи програми одержимо:

```
Автор: Є. Гуцало
Назва: "Мертва зона"
```

```
Автор: М. Коцюбинський
Назва: "Коні не винні"
Автор: П. Куліш
Назва: "Січові гості Чуприна і Чортоус"

Array ( [Є. Гуцало] => Назва: "Мертва зона"
        [М. Коцюбинський] => Назва: "Коні не винні"
        [П. Куліш] => Назва: "Січові гості Чуприна і Чортоус")
```

8.4.2. Функція array_slice

Оскільки масив – це набір елементів і, ймовірно, буде потрібно виділити з нього яку-небудь частину. Мова PHP для цього має функцію array_slice.

Її синтаксис такий:

```
array_slice (масив, номер_елемента [, довжина])
```

Ця функція виділяє масив довжини `довжина` в масиві `масив`, починаючи з елемента, номер якого заданий параметром `номер_елемента`. Параметр `номер_елемента`, який більше нуля, вказує на порядковий номер елемента відносно початку масиву, а той, що менше нуля – на номер елемента з кінця масиву.

Приклад 8.14

```
<?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 )
$sub_arr = array_slice($arr,-2);

// Виведе Array ( [0] => 4 [1] => 5 )
print_r($sub_arr);
?>
```

Якщо задати параметр `довжина` при використанні функції array_slice, то буде виділений масив, що має рівно стільки елементів, скільки задано цим параметром.

Приклад 8.15

```
<?php
$arr = array(1,2,3,4,5);

// Містить масив з елементів 3, 4
$sub_arr = array_slice($arr, 2, 2);

// Теж містить масив з елементів 3, 4
$sub = array_slice($arr,-3, 2);
```

```
// Містить масив з елементів 1, 2, 3, 4
$sub1 = array_slice($arr,0, -1);

// Містить масив з елементів 2, 3
$sub2 = array_slice($arr,-4, -2);
?>
```

8.4.3. Функція array_chunk

Є ще одна функція, схожа на array_slice() – це array_chunk(). Вона розбиває масив на декілька масивів заданої довжини.

Синтаксис її такий:

```
array_chunk ( масив, розмір [, зберігати_ключі])
```

У результаті роботи функція array_chunk() повертає багатовимірний масив, його елементи – це отримані масиви.

Якщо задати параметр `зберігати ключі` як true, то при розбивці будуть збережені ключі вихідного масиву. У противному разі ключі елементів заміняються числовими індексами, які починаються з нуля.

Наприклад, є список запрошених, оформлений у вигляді масиву їхніх прізвищ, і є столики на три персони. Необхідно розподілити всіх запрошених по троє.

Приклад 8.16

```
<?php

$persons = array("Чумак", "Головко", "Олійник", "Кожушко",
"Матвієнко");
$triples = array_chunk($persons,3);

// Ділимо масив на частини 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>";
}
?>
```

У результаті одержимо:

```
за столиком номер 0 сидять:

• Чумак
• Головко
• Олійник
```

```
за столиком номер 1 сидять:

• Кожушко
• Матвієнко
```

Висновки

У даному розділі розглянуто такі основні питання:

– визначення масивів;
– функції пошуку елементів у масивах;
– функції сортування.

Контрольні питання

1. Як створити новий масив?
2. Як виконується об’єднання масивів?
3. Цим відрізняються однакові та еквівалентні масиви?
4. Як підраховувати кількість елементів масиву?
5. Наведіть приклад пошуку елементів у масиві.
6. Якою функцією треба скористуватися, щоб одержати всі ключі масиву?
7. За допомогою яких функцій можна виконати сортування елементів масиву?
8. Чим відрізняються функції asort, rsort, arsort?
9. Що таке сортування за допомогою функцій?
10. Як вивільнити частину масиву?

9. РЯДКИ

Навчальною метою розділу є ознайомлення студентів з рядками мови PHP.

У результаті вивчення даного розділу студенти повинні знати:

– визначення рядків;
– методи пошуку у рядках;
– засоби виділення частини рядка;
– механізми заміни підрядків;
– інші операції з рядками.

Раніше було розглянуто декілька способів створення рядків: за допомогою одинарних лапок, подвійних лапок і за допомогою heredoc-синтаксису, а також основні розходження між цими способами.

```
<?php

echo 'У такому рядку не обробляються змінні й більшість
послідовностей';
echo "Тут змінні та послідовності обробляються";
echo <<<EOT
Тут теж обробляються як змінні,
так і керуючі послідовності.
І, крім того, можна вводити символи лапок
без їх екранування зворотним слешем.
EOT;

?>
```

Починаючи з перших частин навчального посібника, використовувалася функція echo. Насправді, echo – не функція, а мовна конструкція, тому застосовувати при її виклику круглі дужки не обов'язково. Echo дозволяє виводити на екран рядки, передані їй як параметри. Параметрів у функції echo може бути скільки завгодно. Їх розділяють комами або поєднують за допомогою оператора конкатенації, однак ніколи не беруть у круглі дужки.

```
<?
// Виведе рядок "Розум – скарб людини"
echo "Розум – ", "скарб ", "людини";

// Багато хто надає перевагу передавати декілька
// параметрів у команду echo за допомогою конкатенації
echo "Розум – " . " скарб " . "людини";

// Спровокує помилку: unexpected ','
echo ("Розум – ", " скарб ", "людини");
?>
```

Існує скорочений синтаксис для команди echo:

```
<?=рядок_для_виводу?>
```

Тут параметр `рядок_для_виводу` містить рядок, який заданий будь-яким з відомих способів.

Наприклад, така програма виведе на екран напис "Мене звати Петро", виділений червоним кольором:

```
<? $name="Петро" ?>
<font color=red>Мене звати <?=$name?></font>
```

Крім мовної конструкції echo, існує ряд функцій для виводу рядків. Це в першу чергу функція print і її різновиди printf, sprintf і т.п.

Функція print дозволяє виводити на екран тільки один рядок. Вона так само, як і echo, не може бути викликана за допомогою змінних функцій, оскільки є мовною конструкцією.

На жаль, функція print_r не належить до рядкових функцій. Вона відображає інформацію про змінну в зрозумілій формі.

Функції sprintf і printf обробляють переданий їм рядок відповідно до заданого формату. Розглянемо, як можна здійснювати пошук у тексті, поданому у вигляді рядка.

9.1. Пошук елемента в рядку

Для того щоб визначити, чи входить даний рядок до складу іншого рядка, використовується функція strpos(). Синтаксис strpos() такий:

```
strpos (вихідний рядок, рядок для пошуку[, символ, з якого шукати])
```

Вона повертає позицію знаходження шуканого рядка у вихідному рядку або повертає логічне false, якщо нічого не знайдено. Додатковий аргумент дозволяє задавати символ, починаючи з якого буде виконуватися пошук. Крім логічного false, ця функція може повертати й інші значення, які зводяться до false (наприклад, 0 або ""). Тому для того, щоб перевірити, чи знайдено шуканий рядок, рекомендують використовувати оператор еквівалентності «===».

Приклад 9.1

```
<?
$str = "Ще не вмерла Україна, і слава, і воля";

$pos = strpos($str, "слава");
if ($pos !== false)
  echo "Шуканий рядок знаходиться в позиції номер $pos";
else
  echo "Шуканий рядок не знайдений";
```

```
/*
  Важливо розуміти, що перевіряється значення $pos
  на еквівалентність із false.
  Інакше рядок, що перебуває в першій позиції,
  не буде знайдено, тому що 0 інтерпретується як false.
*/

?>
```

Якщо значення параметра `рядок_для_пошуку` не є рядком, то воно перетворюється на цілий тип і розглядається як ASCII-код символа. Щоб одержати ASCII-код будь-якого символу у мові PHP, можна скористатися функцією ord("символ").

Наприклад, якщо напишемо $pos = strpos($str,228); то інтерпретатор буде вважати, що виконується пошук символа «д».

Функція, обернена за змістом ord, – це chr (код символа). Вона за ASCII-кодом виводить символ, що відповідає цьому коду.

Використовуючи функцію strpos, можна знайти номер тільки першого застосування рядка у вихідному рядку.

Звичайно, є функції, які дозволяють знайти номер останнього застосування рядка у вихідному рядку. Це функція strrpos(). Її синтаксис такий:

```
strrpos (вихідний рядок, символ для пошуку)
```

На відміну від strpos() ця функція дозволяє знайти позицію останнього застосування в рядку зазначеного символа.

Бувають ситуації, коли знайти позицію, де перебуває рядок, необов'язково, а потрібно тільки одержати всі символи, які розташовані після входження цього рядка. Можна, звичайно, скористатися і наведеними вище функціями strpos() і strrpos(), але можна зробити й інакше – знайти рядок за допомогою призначених саме для цього функцій.

9.2. Виділення підрядка

9.2.1. Функція strstr

Для виділення підрядка із шуканого рядка у мові PHP в першу чергу слід використовувати функцію strstr():

```
strstr (вихідний рядок, рядок для пошуку)
```

Вона знаходить першу появу рядка і повертає підрядок, починаючи з цього шуканого рядка до кінця вихідного рядка.

Якщо «рядок для пошуку» не знайдено, то функція поверне false. Якщо «рядок для пошуку» не належить рядковому типу даних, то вона переводиться в ціле число і розглядається як код символа. Крім того, ця функція чутлива до регістра, тобто якщо ми будемо паралельно шукати входження слів «Ідея» і «ідея», то результати будуть різними. Замість strstr() можна використати абсолютно ідентичну їй функцію strchr().

Наприклад, виділимо з рядка, який містить назву та автора дослідження, підрядок, що починається із слова «Назва».

Приклад 9.2

```
<?php

$str = "
автор: Цвіркун Леонід (<a href=mailto:TsvirkunL@gmail.com>написати
лист</a>),
Назва: 'Розробка програмного забезпечення' ";
echo "<b>Вихідний рядок: </b>",$str;
if (!strstr($str, "назва")) echo "Рядок не знайдений<br>";
else echo "<p><b>Отримано підрядок: </b>", strstr($str, "Назва");

?>
```

У результаті одержимо:

```
Вихідний рядок: автор: Цвіркун Леонід
(написати лист),
Назва: 'Розробка програмного забезпечення'
Отримано підрядок: назва: 'Розробка програмного забезпечення'
```

Для реалізації регістронезалежного пошуку підрядка існує відповідний аналог цієї функції – функція stristr (вихідний рядок, шуканий рядок). Діє і використовується вона так само, як і strstr(), за винятком того, що регістр, у якому записані символи шуканого рядка, не впливає на пошук.

Оскільки на практиці не завжди потрібно одержати підрядок, що починається з певного слова або рядка, то функція strstr() використовується не часто. Але в деяких випадках і вона може знадобитися. Крім того, мова PHP має і більш зручні функції для пошуку.

9.2.2. Функція substr

Іноді не відомо, з яких символів починається шуканий рядок, але знаємо, наприклад, що починається він з п'ятого символа і закінчується за два символи до кінця вихідного рядка.

Як виділити підрядок за таким описом?

Дуже просто, за допомогою функції substr(). Її синтаксис можна записати в такий спосіб:

```
substr (вихідний рядок, позиція початкового символа[, довжина])
```

Ця функція повертає частину рядка довжиною, заданою параметром `довжина`, починаючи із символа, зазначеного параметром `позиція початкового символа`.

Позиція, з якої починається виділений підрядок, може бути як додатним цілим числом, так і від`ємним.

В останньому випадку відлік елементів виконується з кінця рядка. Якщо параметр довжина не вказати, то функція substr() поверне підрядок від зазначеного символа і до кінця вихідного рядка.

Довжина виділеного підрядка теж може бути задана від`ємним числом. Це означає, що зазначене число символів відкидається з кінця рядка.

Припустімо, є фраза, яка виділена жирним шрифтом за допомогою тегу <b> мови HTML. Необхідно одержати цю фразу у звичайному стилі.

Напишемо для цього програму.

Приклад 9.3

```
<?php
$word = "<b>Привіт, Україно!</b>";
echo $word , "<br>";
$pure_str = substr($word, 3, -4);
/*
  виділяємо підрядок, починаючи з 3-го символа і
  не включаючи 4 символи з кінця рядка
*/
echo $pure_str;
?>
```

У результаті роботи програми одержимо:

```
Привіт, Україно!
Привіт, Україно!
```

Насправді вирішити таке завдання можна набагато простіше – за допомогою функції strip_tags:

```
strip_tags (рядок[, припустимі теги])
```

Ця функція повертає рядок, з якого вилучені всі теги (html і php). За допомогою додаткового аргументу можна задати теги, які не будуть вилучені з рядка. Список з декількох тегів уводиться без яких-небудь знаків роздільників.

Функція видає попередження, якщо зустрічають неправильні або неповні теги.

Приклад 9.4

```
<?php
$string = "<b>Bold text</b> <i>Italic text</i>";
$str = strip_tags($string);

// Видаляємо всі теги з рядка
$str1 = strip_tags($string, '<i>');

// Видаляємо всі теги, крім тегу <i>
$str2 = strip_tags($string, '<i><b>');
```

```
// Видаляємо всі теги, крім тегів <i> і <b>
echo $str,"<br>",$str1,"<br>", $str2;
?>
```

У результаті одержимо:

```
Bold text Italic text
Bold text Italic text
Bold text Italic text
```

Наведемо інший приклад використання функції substr(). Припустімо, у нас є якесь повідомлення з вітанням і підписом автора. Ми хочемо видалити спочатку вітання, а потім і підпис, залишивши тільки змістовну частину повідомлення.

Приклад 9.5

```
<?php
$text = "Привіт! Сьогодні ми вивчаємо роботу з рядками. Автор.";

// Вилучаємо вітання
$no_hello = substr($text, 8);

// Те саме, що substr($text, 8, -6). Вилучаємо підпис.
$content = substr($text, 8, 45);
echo $text, "<br>", $no_hello, "<br>", $content;
?>
```

У результаті одержимо:

```
Привіт! Сьогодні ми вивчаємо роботу з рядками. Автор.
Сьогодні ми вивчаємо роботу з рядками. Автор.
Сьогодні ми вивчаємо роботу з рядками.
```

Якщо нам потрібно одержати один конкретний символ з рядка, знаючи його порядковий номер, то не слід використовувати функції типу substr. Можна скористатися більш простим синтаксисом – вказавши номер символа у фігурних дужках після імені рядкової змінної. У контексті попереднього прикладу букву «р», що записана другою за порядком, можна одержати так:

```
echo $text{1}; // виведе символ "р"
```

Помітимо, що номером цього символа є число один, а не два, тому що нумерація символів рядка починається з нуля.

І тоді виникає питання, скільки всього символів у рядку і як це можна дізнатися?

Число символів у рядку – це довжина рядка. Обчислити довжину рядка можна за допомогою функції strlen:

```
strlen (рядок)
```

Наприклад, довжина рядка «Розробка інформаційної моделі» обчислюється за допомогою команди: strlen("Розробка інформаційної моделі"); і дорівнює 29 символам.

Отже, було розглянуто, як виділяти і знаходити підрядки. Тепер навчимося заміняти рядок, що входить до складу вихідного рядка, на інший, що з`являється за вибором.

9.3. Заміна входження підрядків

9.3.1. Функція str_replace

Для заміни входження підрядка можна використати функцію str_replace(). Це проста і зручна функція, яка дозволяє вирішувати безліч завдань, що не вимагають особливих складностей для заміни рядка на інший, що з`являється за вибором. Для того, щоб зробити заміни з більш складними умовами, треба використати 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 розглянутому об'єкті значення і заміняє його таким, що призначене для заміни. Виникає питання: чому розглядаються не рядки для пошуку й заміни і вихідний рядок, а значення та об'єкт, у якому відбувається заміна?

Справа в тому, що, починаючи з версії мови PHP 4.0.5, будь-який аргумент цієї функції може бути масивом.

Якщо об'єкт, у якому проводиться пошук і заміна, є масивом, то ці дії виконуються для кожного елемента масиву і у результаті повертається новий масив.

Приклад 9.6

```
<?php
// Об'єкт
$greeting = array("Привіт", "Привіт усім!", "Привіт, Україно!");

// Робимо заміну
$new_greet = str_replace("Привіт", "Добрий ранок", $greeting);

print_r($new_greet);
/* одержимо: Array (
  [0] => Добрий ранок
  [1] => Добрий ранок усім!
  [2] => Добрий ранок, Україно!
)
*/
?>
```

Якщо шукане значення і значення для заміни – це масиви, то береться по одному значенню з кожного масиву і виконується їхній пошук і заміна в об'єкті.

Якщо значень для заміни менше, ніж значень для пошуку, то як нові значення використається порожній рядок.

Приклад 9.7

```
<?php
// Об'єкт
$greeting = array("Привіт", "Привіт усім!", "Привіт,
кохана!","Здрастуйте","Здрастуйте, товариші", "Ні");

// Значення, які будемо заміняти
$search = array ("Привіт", "Здрастуйте", "Ні");

// Значення, якими будемо заміняти
$replace = array ("Добрий ранок", "День добрий");

// Виконуємо заміну
$new_greet = str_replace($search, $replace, $greeting);

// Виводимо отриманий масив
print_r($new_greet);
?>
```

У результаті одержимо такий масив:

```
Array (
[0] => Добрий ранок
[1] => Добрий ранок усім!
[2] => Добрий ранок, кохана!
[3] => День добрий
[4] => День добрий, товариші
)
```

Якщо значення для пошуку – масив, а значення для заміни – рядок, то цей рядок буде використано для заміни всіх знайдених значень.

Приклад 9.8

```
<?php
// Об'єкт
$greeting = array("Привіт", "Привіт усім!",
   "Привіт, кохана!", "Здрастуйте", "Здрастуйте, товариші");

// Значення, які будемо заміняти
$search = array ("Привіт","Здрастуйте");

// Значення, яким будемо заміняти
$replace = "День добрий";

// Робимо заміну
$new_greet = str_replace($search, $replace, $greeting);
```

```
// Виводимо отриманий масив
print_r($new_greet);
?>
```

Одержимо:

```
Array (
[0] => День добрий
[1] => День добрий всім!
[2] => День добрий, кохана!
[3] => День добрий
[4] => День добрий, товариші
)
```

Функція str_replace() чутлива до регістра, але існує її регістронезалежний аналог – функція str_ireplace(). Однак ця функція підтримується не у всіх версіях мови PHP.

Ще один приклад використання функції str_replace() – обробка шаблонів.

Звернемося до опису якої-небудь книги, наприклад навчального посібника.

Багато разів було вже створено форму для введення подібного опису і навіть відображення даних, які вписані користувачем у такого роду форму.

Тепер зробимо так, щоб спосіб відображення даних задавав сам користувач. Для цього додамо у форму ще один елемент для введення шаблону.

```
<h2>Уведіть опис навчального посібника</h2>
<form action=”sbl.php”>
<table border=”0”>
<tr>
  <td>Назва </td>
  <td><input type=”text” name=”title”></td>
</tr>
<tr>
  <td>Короткий зміст</td>
  <td><textarea name=”text” cols=56 rows=3
name=”description”></textarea></td>
</tr>
<tr>
  <td>Автор або автори</td>
  <td><input type=”text” name=”author”></td>
</tr>
<tr>
  <td>Рік публікації</td>
  <td><input type=”text” name=”published”></td>
</tr>
<tr>
  <td>Шаблон документа</td>
  <td><textarea name=”shablon” cols=42 rows=3></textarea></td>
</tr>
</table>
<input type=submit value="Відправити">
</form>
```

Однак розробити просто поля для введення шаблона недостатньо. Одна людина введе в нього одне, інша – інше. Потрібно домовитися про те, як створювати шаблони, що можна в них використовувати, тобто потрібно придумати мову шаблонів.

Наприклад, домовимося, що при створенні шаблону можна задіяти будь-які html-теги, а набір спецсимволів типу <!ім'я_елемента> встановлює значення елемента з ім'ям <!ім'я_елемента>.

Для обробки таких шаблонів можна використати функцію str_replace().

Програма sbl.php наведена далі у прикладі.

Приклад 9.9

```
<?php
$tmpl = $_GET["shablon"];
/*
  Шаблон, уведений користувачем.
  Наприклад, це може бути такий рядок:
  "<h1><!title></h1><p><font size=-1>
  <!description></font></p><p align=right>
  <!author><br><!published></p>"
*/
// Функція, що робить заміну елемента шаблону на його значення
function show(){
  global $tmpl;
  foreach($_GET as $k => $v) {
  $tmpl = str_replace("<!$k>",$v,$tmpl);
  }
  echo $tmpl;
}
show();
?>
```

Введемо у форму такі дані, як показано на рис. 9.1.

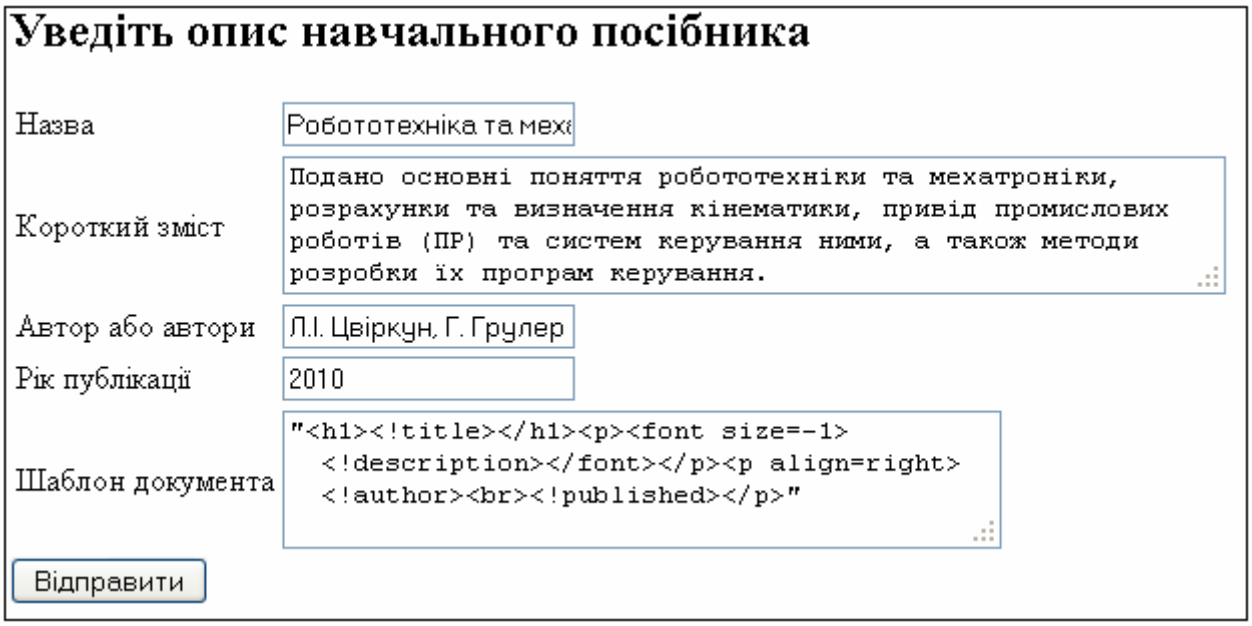


Рис. 9.1. Приклад введення у форму опису навчального посібника

У результаті одержимо:

```
Робототехніка та мехатроніка
Подано основні поняття робототехніки та мехатроніки, розрахунки та
визначення кінематики, привід промислових роботів (ПР) та системи
керування ними, а також методи розробки їх програм керування.
Наведено класифікацію мехатронних об’єктів, концепцію проектування
мехатронних систем, особливості розробки лабораторій для вивчення
мехатронних об’єктів через Інтернет
Л.І. Цвіркун, Г. Грулер
2010
```

9.3.2. Функція substr_replace

Ця функція сполучає в собі властивості двох уже розглянутих функцій – str_replace() і substr(). Її синтаксис такий:

```
substr_replace (вихідний рядок, рядок для заміни,
позиція початкового символа[, довжина])
```

Ця функція заміняє частину рядка рядком, призначеним для заміни, тобто ту, яка починається з позиції, зазначеної параметром `позиція початкового символа`. За допомогою додаткового аргументу `довжина` можна обмежити число замінних символів. Отже, фактично можна не вказувати конкретно рядок, який потрібно замінити, а тільки зазначити, де він міститься і, можливо, яку довжину має.

У цьому відмінність функції substr_replace() від str_replace().

Як і у випадку з функцією substr(), аргументи `позиція початкового символа` і `довжина` можуть бути від`ємними. Якщо `позиція початкового символа` від`ємна, то заміна відбувається, починаючи з цієї позиції відносно кінця рядка. Від`ємна довжина задає, скільки символів від кінця рядка не можна заміняти.

Якщо довжина не вказується, то заміна відбувається до кінця рядка.

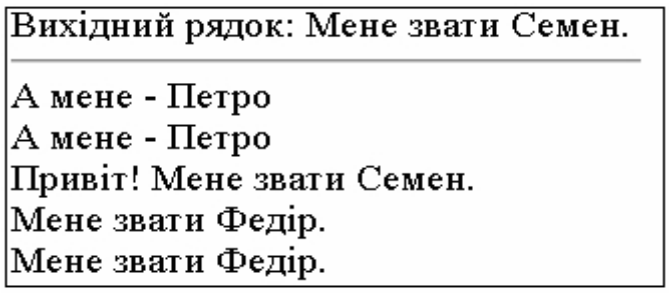
Приклад 9.10

```
<?php
$text = "Мене звати Семен.";
echo "Вихідний рядок: $text<hr>\n";
/* Наступні два рядки замінять увесь
вихідний рядок рядком 'А мене - Петро' */
echo substr_replace($text,'А мене - Петро', 0)."<br>\n";
echo substr_replace($text,'А мене - Петро', 0,
strlen($text))."<br>\n";
// Наступний рядок додасть слово 'Привіт! '
// у початок вихідного рядка
echo substr_replace($text, 'Привіт! ', 0, 0) . "<br>\n";

// Наступні два рядки замінять ім'я Семен
// на ім'я Федір у вихідному рядку
```

```
echo substr_replace($text, 'Федір', 11, -1) . "<br>\n";
echo substr_replace($text, 'Федір', -6, -1) . "<br>\n";
?>
```

У результаті роботи цієї програми одержимо:



9.4. Поділ і з'єднання рядка

Дуже корисними вважаються функція поділу рядка на частини і зворотна їй функція об'єднання рядків в один рядок.

Чому дуже корисні?

Наприклад, якщо динамічно генерується форма 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, саме і знадобиться вміння розбивати рядок на частини.

Для реалізації такого розбиття у мові PHP можна використати кілька функцій:

```
explode(роздільник,вихідний рядок[, максимальна кількість
елементів])
split (шаблон, вихідний рядок[, максимальна кількість елементів])
preg_split (шаблон, вихідний рядок[, максимальна кількість
елементів[, прапори]])
```

Останні дві функції працюють із регулярними конструкціями, тому тепер їх розглядати не будемо.

Розглянемо більш просту функцію – explode().

Функція explode() ділить вихідний рядок на підрядки, кожен з яких відділений від сусіднього за допомогою зазначеного роздільника, і повертає масив отриманих рядків.

Якщо задано додатковий параметр `максимальна кількість елементів`, то їх у масиві буде не більше ніж зазначено у цьому параметрі, до того ж в останній елемент записується весь залишок рядка.

У випадку, коли як роздільник наведений порожній рядок «""», то функція explode() поверне false.

Якщо символа роздільника у вихідному рядку немає, то вихідний рядок повертається без змін.

Наприклад, необхідно створити елемент форми – випадний список, значення для цього списку повинен увести користувач, не знайомий з мовою html.

Створимо таку форму:

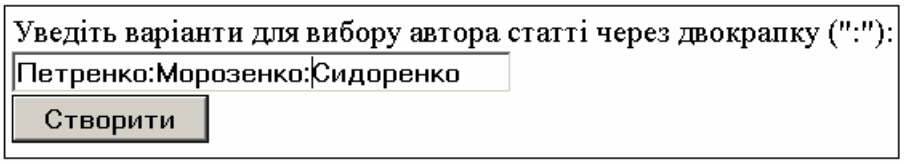
```
<form action=exp.php>
   Уведіть варіанти для вибору автора статті
      через двокрапку (":"):<br>
   <input type=text name=author size=40>
   <br>
   <input type=submit value=Створити елемент>
</form>
```

Програма, що буде її обробляти (exp.php), може бути такою, як наведена у прикладі.

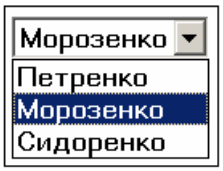
Приклад 9.11

```
<?php
$str = $_GET["author"];
$names = explode(":",$str);
// Розбиваємо рядок, уведений користувачем, за допомогою ":"
$s = "<select name=author>";
// Створюємо випадний список
foreach ($names as $k => $name) {
  // Додаємо елементи до списку
  $s .= "<option value=$k>$name";
}
$s .= "</select>";
echo $s;
?>
```

Якщо введемо такий рядок у форму:



одержимо випадний список:



Крім поділу рядка на частини, іноді виникає необхідність об'єднання декількох рядків в одне ціле. Функція, пропонована для цього мовою PHP, називається implode():

```
implode (масив рядків, об`єднуючий елемент)
```

Ця функція поєднує елементи масиву за допомогою переданого їй об'єднуючого елемента (наприклад коми). На відміну від функції explode(), порядок аргументів у функції implode() не має значення.

Припустімо, зберігаємо ім'я, прізвище і по батькові людини окремо, однак виводити їх на сторінці потрібно разом. Щоб з'єднати їх в один рядок, можна використати функцію implode():

```
<?php
$data = array("Чумак","Гнат","Петрович");
$str = implode($data," ");
echo $str;
?>
```

У результаті роботи цієї програми одержимо рядок:

```
Чумак Гнат Петрович
```

У функції implode() існує псевдонім – функція join(), тобто ці дві функції відрізняються лише іменами.

9.5. Рядки з html-кодом

Досить часто необхідно працювати з рядками, що містять html-теги. Якщо ввести такий рядок у браузер за допомогою звичайних функцій відображення даних echo() або print(), то самих html-тегів не буде видно, а рядок буде відформатований згідно з цими тегами. Браузери обробляють всі html-теги відповідно до стандарту мови HTML.

Іноді треба отримати рядок без обробки його браузером. Щоб цього домогтися, потрібно перед тим як виводити, застосувати до нього функцію htmlspecialchars().

Функція htmlspecialchars (рядок[, стиль лапок[, кодування]]) переводить спеціальні символи, такі як «<», «>», «&», «"» , «'», у відповідні сутності мови HTML «<», «>», «&», «"», «'».

Додатковий аргумент стилю лапок визначає, як повинні інтерпретуватися подвійні та одинарні лапки. Він може мати одне з трьох значень: ENT_COMPAT, ENT_QUOTES або ENT_NOQUOTES.

Константа ENT_COMPAT означає, що подвійні лапки мають бути переведені в спецсимволи, а одинарні – залишитися без змін.

ENT_QUOTES указує, що повинні конвертуватися і подвійні, й одинарні лапки, а ENT_NOQUOTES залишає і ті й інші лапки без змін.

У параметрі кодування можуть бути задані такі кодування, як UTF-8, ISO-8859-1 та інші, але жодне кодування кирилиці тут не підтримується.

```
<?php
$new = htmlspecialchars("<a
href='mailto:TsvirkunL@gmail.com'>Написати лист</a>", ENT_QUOTES);
echo $new;
```

```
/ * Рядок, буде перекодований в такий:
&lt;a href=&#039;mailto:TsvirkunL@gmail.com&#039;&gt;
Написати лист&lt;/a&gt; */
```

У програмі-переглядача побачимо:

```
<a href='mailto:TsvirkunL@gmail.com'>
Написати лист</a>
```

Функція htmlspecialchars() перекодує тільки найчастіше використовувані спецсимволи. Якщо необхідно конвертувати всі символи в сутності мови HTML, варто задіяти функцію htmlentities(). Букви, які подані кирилицею при використанні цієї функції, теж кодуються спеціальними послідовностями.

Наприклад, буква «А» заміняється комбінацією «À». Синтаксис і принцип дії функції htmlentities() аналогічний синтаксису і принципу дії htmlspecialchars().

Висновки

У даному розділі розглянуто такі основні питання:

- визначення рядків;
- методи пошуку в рядках;
- засоби виділення частини рядка;
- механізми заміни підрядків.

Контрольні питання

1. Що таке рядки?
2. Як рядки використовуються у PHP-програмах?
3. Яким чином виконується пошук у рядках?
4. Як можна виділити частину рядка?
5. Для чого служить функція htmlspecialchars()?
6. Для чого служить функція htmlentities()?

ЧАСТИНА 3. ОСНОВИ МОДЕЛІ КЛІЄНТ-СЕРВЕРНІ ТЕХНОЛОГІЇ
10. ОБРОБКА ЗАПИТІВ

Навчальною метою розділу є ознайомлення студентів з принципами обробки запитів.

У результаті вивчення даного розділу студенти повинні знати:

- основи моделі клієнт-серверні технології;
- мережеві протоколи;
- способи передачі даних на сервер;
- форми запиту клієнта;
- методи запитів.

10.1. Модель клієнт-серверні технології

10.1.1. Основні поняття

У перших розділах посібника вже розповідалося про те, що PHP – це мова, яка обробляється сервером. Зараз треба уточнити, що ж таке сервер, які функції він виконує і які взагалі вони бувають.

Якщо мова йде про сервер, мимоволі спливає в пам'яті поняття клієнта, оскільки ці два поняття пов'язані між собою. Їх поєднує комп'ютерна архітектура клієнт-сервер.

Звичайно, коли говорять «сервер», часто мають на увазі сервер в архітектурі клієнт-сервер, а коли говорять «клієнт» – мають на думці клієнт у цій архітектурі.

Так що ж це за архітектура?

Суть її полягає в розділенні функцій між двома підсистемами: клієнтською, що відправляє запит на виконання яких-небудь дій, і серверною, що виконує цей запит. Взаємодія між клієнтом і сервером відбувається за допомогою стандартних спеціальних протоколів, таких як TCP/IP.

Насправді протоколів дуже багато, вони розрізняються за рівнями. Але далі розглянемо тільки протокол прикладного рівня HTTP (трохи пізніше), оскільки для вирішення таких завдань потрібен тільки він.

А поки повернемося до моделі клієнт-серверна архітектура і розберемося, що ж таке клієнт і що таке сервер.

Сервер являє собою комп’ютер, на якому працює набір програм, які контролюють виконання різних процесів. Часто комп'ютер, на якому встановлений сервер, і називають сервером. Основна функція комп'ютера-сервера – на запит клієнта розпочати який-небудь процес і відправити клієнту результати його роботи.

Клієнтом називають будь-який процес, що користується послугами сервера. Клієнтом може бути як користувач, так і програма. Основне завдання клієнта – виконання зв'язку із сервером, тобто клієнт повинен надавати користувачеві інтерфейс для роботи.

Взаємодія між клієнтом і сервером починається з ініціативи клієнта. Клієнт розпочинає сеанс, одержує потрібні йому результати і повідомляє про закінчення роботи.

Послугами одного сервера найчастіше користується декілька клієнтів одночасно. Тому кожен сервер повинен мати досить високу продуктивність і створювати відповідні умови для безпеки даних.

Найбільш логічно встановлювати сервер на комп'ютері, що входить у яку-небудь мережу, локальну або глобальну. Однак можна встановлювати сервер і на окремий комп'ютер (тоді він буде одночасно і клієнтом, і сервером).

Існує безліч типів серверів. Нижче описані деякі з них.

Відеосервер. Такий сервер спеціально пристосований до обробки зображень, зберігання відеоматеріалів, відеоігор і т.д. У зв'язку з цим комп'ютер, на якому встановлено відеосервер, повинен мати високу продуктивність і велику пам'ять.

Пошуковий сервер призначений для пошуку інформації в Інтернеті.

Поштовий сервер надає послуги у відповідь на запити, надіслані по електронній пошті.

Веб-сервер призначений для роботи в Інтернеті.

Сервер баз даних виконує обробку запитів до баз даних.

Сервер захисту даних призначений створювати відповідні умови для безпеки даних (містить, наприклад, засоби для ідентифікації паролів).

Файловий сервер забезпечує функціонування розподілених ресурсів, надає послуги пошуку, зберігання даних і можливість одночасного доступу до них декількох користувачів.

Звичайно на комп'ютері-сервері працює відразу кілька програм. Одна займається електронною поштою, друга розподілом файлів, третя надає веб-сторінки і т.д.

З усіх типів серверів нас в основному цікавить веб-сервер. Часто його називають WWW-сервером, http-сервером або навіть просто сервером.

Що являє собою веб-сервер?

По-перше, це місце зберігання інформаційних ресурсів.

По-друге, ці ресурси зберігаються і надаються користувачам згідно із стандартами Інтернету, наприклад протокол передачі даних HTTP. Як передається інформація відповідно до цього протоколу, ми розглянемо трохи пізніше.

Робота з документами веб-сервера здійснюється за допомогою програми-переглядача (наприклад, IE, Opera або Mozilla), яка відправляє серверу запити, створені відповідно до протоколу HTTP. У процесі виконання завдання сервер може зв'язуватися з іншими серверами.

Далі під терміном «сервер», будемо мати на увазі веб-сервер.

Як приклади веб-серверів можна навести такі сервери: Apache групи Apache; Internet Information Server (IIS) компанії Microsoft; SunOne фірми Sun Microsystems; WebLogic фірми BEA Systems; IAS (Inprise Application Server) фірми Borland; WebSphere фірми IBM; OAS (Oracle Application Server).

10.1.2. Протокол HTTP

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

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

HTTP являє собою набір методів для визначення цілей запиту, що відправляються серверу. Для вказівки ресурсу, до якого повинен бути застосований даний метод, використовується універсальний ідентифікатор ресурсів URI (Universal Resource Identifier) у вигляді місцезнаходження ресурсу URL (Universal Resource Locator) або його універсального імені URN (Universal Resource Name).

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

HTTP використовується для комунікацій між різними програмами і програмами-шлюзами, що надають доступ до існуючих Інтернет-протоколів.

Протокол реалізує принцип запит/відповідь. Запитувальна програма-клієнт ініціює взаємодію з програмою-сервером, що відповідає і надсилає запит, який містить:

– метод доступу;

– адресу URI;

– версію протоколу;

– повідомлення з інформацією про тип переданих даних та про клієнта, який надіслав запит і, можливо, зі змістовною частиною (тілом) повідомлення.

У відповіді сервера:

– інформація про стан (версія протоколу і код повернення);

– повідомлення, у яке входить інформація сервера, метаінформація 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 відправлення відповіді ініціює його закриття.

Розглянемо більш докладно, у якій формі відправляються запити на сервер.

10.1.3. Запит клієнта до сервера

10.1.3.1. Форма запиту

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

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

```
<Простий запит> := <Метод> <символ пробіл>
```

```
<Запитуваний URI> <символ нового рядка>
```

Як методи можуть бути зазначені GET, POST, HEAD, PUT, DELETE та ін. Про найпоширеніші з них поговоримо пізніше.

Як запитуваний URI найчастіше використається URL-адреса ресурсу.

Приклад простого запиту:

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

Тут GET – це метод доступу, тобто метод, що повинен бути застосований до ресурсу, а http://phpbook.info/ – це URL-адреса ресурсу.

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

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

Квадратні дужки тут позначають необов'язкові елементи заголовка, через вертикальну рису перераховані альтернативні варіанти. Елемент <Рядок стану> містить метод запиту і URI ресурсу (як і простий запит) і, крім того, версію протоколу HTTP. Наприклад, для виклику зовнішньої програми можна задіяти такий рядок стану:

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

У цьому випадку використовується метод POST і протокол HTTP версії 1.0.

В обох формах запиту важливе місце займає URI ресурсу.

Найчастіше URI використається у вигляді URL-адреси ресурсу. При звертанні до сервера можна застосовувати як повну форму URL, так і спрощену.

Повна форма містить тип протоколу доступу, адресу сервера ресурсу і адресу ресурсу на сервері (рис. 10.1).

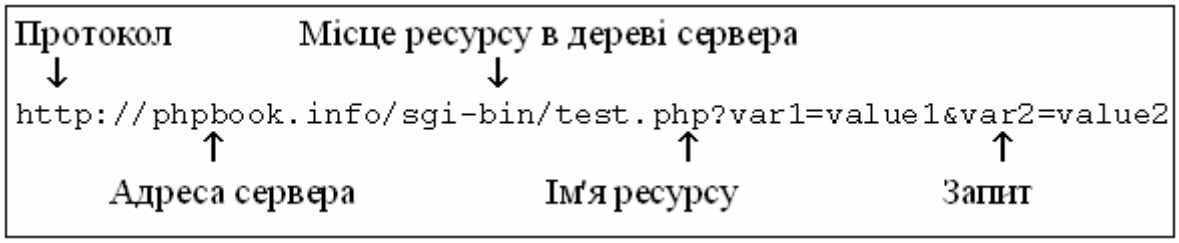


Рис. 10.1. Повна форма URL

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

10.1.3.2. Методи запиту

Розглянемо найпоширеніші методи відправлення запитів.

Будь-який запит клієнта до сервера повинен починатися із 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 одержати будь-які дані, ідентифіковані за допомогою URI у запиті ресурсу. Якщо URI указує 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 рядок статусу).

При використанні методу GET у поле тіла ресурсу повертається викликана інформація (текст HTML-документа, наприклад).

Існує різновид методу GET – умовний GET. Цей метод повідомляє серверу про те, що на запит потрібно відповісти, тільки якщо виконано умову, що втримується в полі if-Modified-Since заголовка запиту.

Якщо говорити більш точно, то тіло ресурсу передається у відповідь на запит, якщо цей ресурс змінювався після дати, зазначеної в if-Modified-Since.

Метод HEAD аналогічний методу GET, тільки не повертає тіло ресурсу і не має умовного аналога. Метод HEAD використовуюсь для одержання інформації про ресурс.

Метод POST розроблений для передачі на сервер такої інформації, як анотації ресурсів, поштові повідомлення, дані для додавання в базу даних, тобто для передачі досить важливої інформації великого обсягу. На відміну від методів GET і HEAD, в POST передається тіло ресурсу, що і є інформацією, одержуваною з форм або інших джерел.

Це тільки основні поняття теорії. Спробуємо використовувати все це на практиці.

Розглянемо, як посилати запити серверу і як обробляти його відповіді.

10.2. Передача даних на сервер

10.2.1. Синтаксис форм для передачі

Як передавати дані на сервер?

Для цього мова HTML має спеціальні конструкції – форми. Форми призначені для того, щоб одержувати від користувача інформацію.

Наприклад, потрібно узнати логін і пароль користувача, щоб визначити, на які сторінки сервера його можна допускати, або необхідні особисті дані його, щоб була можливість з ним зв'язатися. Форми саме і застосовуються для отримання такої інформації. У них можна вводити текст або вибирати варіанти зі списку.

Дані, записані у форму, відправляються спеціальній програмі для обробки (наприклад, PHP-програмі) на сервері. Залежно від вказаних користувачами даних ця програма може формувати різні веб-сторінки, відправляти запити до бази даних, запускати різні програми і т.п.

Розберемося із синтаксисом HTML-форм. Можливо, це відомо, але все-таки повторимо основні моменти, оскільки це важливо.

Отже, для створення форми в мові HTML використовується тег FORM. Усередині його може знаходитися одна або кілька команд INPUT. За допомогою атрибутів action та method тегу FORM задаються ім'я програми, що буде обробляти дані форми, і метод запиту відповідно. Команда INPUT визначає тип і різні характеристики запитуваної інформації. Відправлення даних з форми відбувається після натискання кнопки input типу submit.

Створимо форму для реєстрації учасників семінара у файлі form.html.

Приклад 10.1

```
<h2>Форма для реєстрації учасників семінара “Мови програмування”
</h2>
<form action="1.php" method=POST>
Ім'я<br>
<input type=text name="first_name" value=""><br>
Прізвище<br>
<input type=text name="last_name"><br>
E-mail<br>
<input type=text name="email"><br>
<p>Виберіть назву секції, що 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>Що ви хочете, щоб ми знали про вас? <br>
<textarea name="comment" cols=32 rows=5></textarea>
<p><input name="confirm" type=checkbox checked>
Підтвердити одержання <br>
<input type=submit value="Відправити">
<input type=reset value="Скасувати">
</form>
```

Після обробки браузером цей файл буде виглядати приблизно як показано на рис. 10.2.

Таким чином створюються і виглядають HTML-форми. Як видно, у формі можна вказувати метод передачі даних. Подивимося, що буде відбуватися, якщо вказати метод GET або POST, і в чому буде різниця.

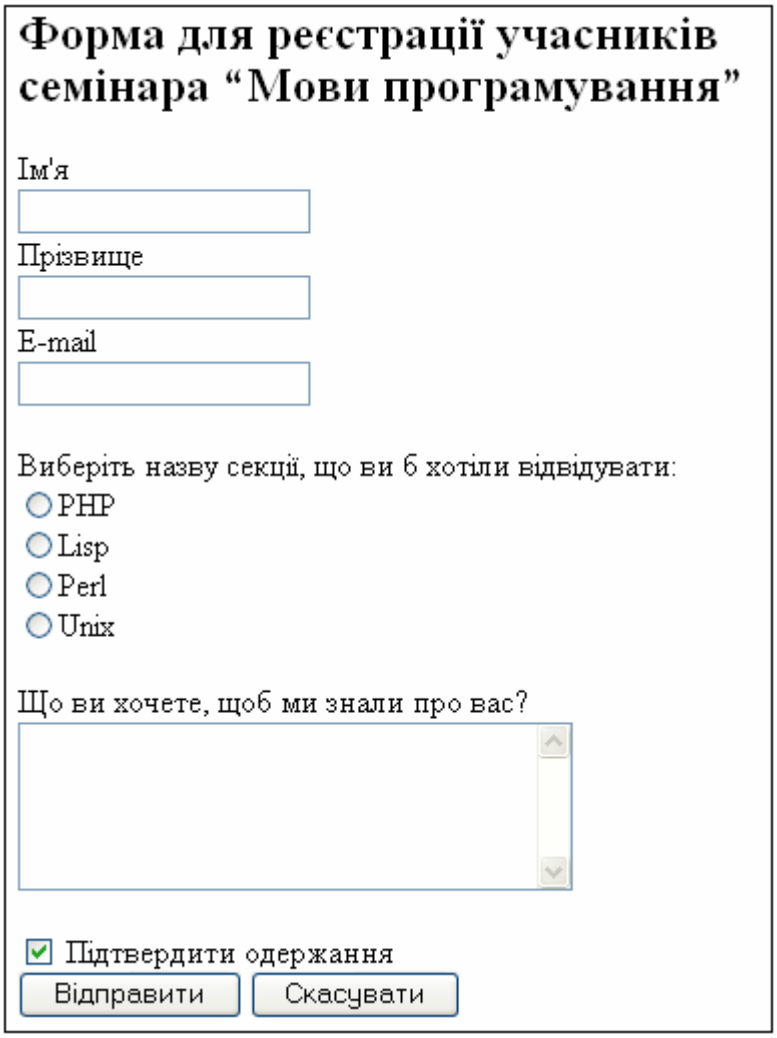


Рис. 10.2. Приклад html-форми

10.2.2. Форми передачі методом GET

При відправленні даних форми за допомогою методу GET зміст форми додається до URL після знака питання у вигляді пар ім'я=значення, об'єднаних за допомогою амперсанта &:

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

Тут action – це URL-адреса програми, що повинна обробляти форму (це або програма, задана в атрибуті action тегу form, або сама поточна програма, якщо цей атрибут опущений). Імена name1, name2, name3 відповідають іменам елементів форми, а value1, value2, value3 – значенням цих елементів. Всі спеціальні символи, які включають = і &, в іменах або значеннях цих параметрів будуть опущені. Тому не варто використовувати в назвах або значеннях елементів форми ці символи і символи кирилиці в ідентифікаторах.

Якщо в поле для введення ввести який-небудь службовий символ, то він буде переданий у його шістнадцатирічному коді, наприклад, символ $ заміниться на %24. Так само передаються й українські букви.

Для полів введення тексту і пароля (це елементи input з атрибутом type=text і type=password) значення буде такими, що запропонує користувач.

Якщо користувач нічого не вводить у таке поле, то в рядку запиту буде присутній елемент name=, де name відповідає імені цього елемента форми.

Для кнопок типу checkbox і radio button значення value визначається атрибутом VALUE у тому випадку, коли кнопка відзначена. Не відзначені кнопки при складанні рядка запиту не використовуються. Кілька 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.

У принципі створювати HTML-форму для передачі даних методом GET не обов'язково.

Можна просто додати в рядок URL потрібні змінні та їхні значення:

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

У зв'язку з цим у передачі даних методом GET є один істотний недолік – кожний користувач може підробити значення параметрів. Тому не рекомендовано використовувати 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, що не дозволено змінювати користувачеві.

Незважаючи на всі ці недоліки, використовувати метод GET досить зручно при налаштуванні програм (тоді можна бачити значення та імена переданих змінних) і для передачі параметрів, що не впливають на безпеку.

10.2.3. Форми передачі методом POST

Зміст форми кодується так само, як для методу GET, але замість додавання рядка до URL уміст запиту посилає блок даних як частину операції POST. Якщо є атрибут ACTION, то значення URL, що там перебуває, визначає, куди посилати цей блок даних. Цей метод, як вже було відзначено, рекомендується для передачі великих за обсягом блоків даних.

Інформація, уведена користувачем і відправлена серверу за допомогою методу POST, подається на стандартне введення програмі, зазначеній в атрибуті action, або поточній, якщо цей атрибут опущений. Довжина файлу, що посилає, передається в змінну оточення CONTENT_LENGTH, а тип даних – у змінну 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, уведене у форму. При використанні POST користувач не бачить передані серверу дані.

Основна перевага POST запитів – це їхня більша безпека і функціональність у порівнянні з GET-запитами.

Тому метод POST частіше використовують для передачі важливої інформації, а також інформації великого обсягу. Проте не варто цілком покладатися на безпеку цього механізму, оскільки дані POST запиту також можна підробити, наприклад створивши html-файл на комп'ютері користувача і заповнивши його потрібними даними. Крім того, не всі клієнти можуть застосовувати метод POST, що обмежує варіанти його використання.

При відправленні даних на сервер будь-яким методом передаються не тільки самі дані, уведені користувачем, але і ряд змінних – змінні оточення, що характеризують клієнта, історію його роботи, шляхи до файлів і т.п. От деякі змінні оточення:

REMOTE_ADDR (IP-адреса хоста (комп'ютера), що відправляє запит);

REMOTE_HOST (ім'я хоста, з якого відправлений запит);

HTTP_REFERER (адреса сторінки, що посилається на поточну програму);

REQUEST_METHOD (метод, що був використаний для відправлення запиту);

QUERY_STRING (інформація, що перебуває в URL після знака питання);

SCRIPT_NAME (віртуальний шлях до програми, що повинна виконуватися);

HTTP_USER_AGENT (інформація про браузер, що використовує клієнт).

10.3. Обробка запитів мовою PHP

Дотепер розглядалося, що запити клієнта обробляються на сервері за допомогою спеціальної програми. Насправді цю програму можно написати (у тому числі й мовою PHP). Для того щоб написати цю програму, необхідно познайомитися з деякими правилами та інструментами мови PHP.

Мова PHP має декілька способів одержання доступу до даних, переданих клієнтом за протоколом HTTP. До версії мови PHP 4.1.0 доступ до таких даних здійснювався за іменами 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 всередині програми з'являлася змінна $first_name зі значенням Nina. Якщо було потрібно 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. Отже, якщо пара first_name=Nina передана методом GET, то $HTTP_GET_VARS["first_name"]="Nina".

Використовувати в програмі імена переданих змінних прямо небезпечно. Тому було вирішено, починаючи з версії мови PHP 4.1.0, задіяти для звертання 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. Його значення можна одержати в будь-якому місці програми, використовуючи як ключ ім'я відповідної змінної (елемента форми).

У прикладі 10.1 подано створену форму для реєстрації учасників семінара. Тоді у програмі прикладу 10.2, що обробляє цю форму, буде запис, наведений далі.

Приклад 10.2

```
<?php
$str = "Доброго дня,
    ".$_REQUEST["first_name"]. "
    ".$_REQUEST["last_name"]."! <br>";
$str .="Ви вибрали секцію мова
    ".$_REQUEST["kurs"];
echo $str;
?>
```

Тоді, якщо у форму введено ім'я «Семене», прізвище «Петренко» і вибрано серед усіх курсів курс лекцій з мови PHP, на екрані одержимо таке повідомлення:

```
Доброго дня, Семене Петренко!
Ви вибрали секцію мова 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 використання не зникли з міркувань сумісності з попередніми версіями мови PHP. На відміну від своїх попередників, масиви $_POST і $_GET стали суперглобальними, тобто доступними прямо і всередині функцій і методів.

Наведемо приклад використання цих масивів.

Припустімо, потрібно обробити форму, що містить елементи введення з іменами first_name, last_name, kurs (наприклад, форму form.html, наведену вище). Дані були передані методом POST.

Приклад 10.3

```
<?php
$str = "Доброго дня,
    ".$_POST["first_name"]."
    ".$_POST["last_name"] ."! <br>";
$str .= "Ви вибрали секцію мова ".
        $_POST["kurs"];
echo $str;
?>
```

Тоді на екрані, якщо ми ввели ім'я «Семене», прізвище «Петренко» і вибрали серед усіх курсів курс лекцій мови PHP, з’явиться таке повідомлення:

```
Доброго дня, Семене Петренко!
Ви вибрали секцію мова PHP
```

Для того, щоб зберегти можливість обробки програм більш ранніх версій, ніж PHP 4.1.0, була введена директива register_globals, що дозволяє або забороняє доступ до змінних безпосередньо за їхніми іменами.

Якщо у файлі налаштувань PHP параметр register_globals=On, то для доступу до змінних, переданих методами GET і POST, можна просто вказувати їхні імена (тобто писати $first_name). А якщо параметр register_globals=Off, то потрібно писати $_REQUEST["first_name"] або $_POST["first_name"], $_GET["first_name"], $HTTP_POST_VARS["first_name"], $HTTP_GET_VARS ["first_name"].

З погляду безпеки цю директиву краще відключати (тобто register_globals=Off). При включеній директиві register_globals перераховані вище масиви також будуть містити дані, передані клієнтом.

Іноді виникає необхідність довідатися значення якої-небудь змінної оточення, наприклад методу, що використовувався для передачі запиту або IP-адреси комп'ютера, з якого надіслано запит.

Одержати таку інформацію можна за допомогою функції getenv(). Вона повертає значення змінної оточення, ім'я якої передане їй як параметр. Використання функції getenv() наведено в прикладі 10.4.

Приклад 10.4

```
<?
getenv('REQUEST_METHOD');
    // поверне використаний метод
echo getenv ('REMOTE_ADDR');
    // виведе IP-адресу комп'ютера,
    // запит, з якого надіслано
?>
```

При використані методу GET дані передаються додаванням рядка запиту у вигляді пар «ім'я_змінної=значення до URL-адреси ресурсу». Усе, що записано в URL після знака питання, можна одержати за допомогою команди

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

Завдяки цьому методом GET можна передавати дані в якому-небудь іншому вигляді.

Наприклад, указувати тільки значення декількох параметрів через знак плюс, а в програмі розбирати рядок запиту на частини або передавати значення всього одного параметра.

У цьому випадку в масиві $_GET з'явиться порожній елемент із ключем, що дорівнює цьому значенню (всьому 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 відправляються серверу. Щоб їх побачити, хакер повинен підмінити нашу форму своєю. Тоді сервер відправить результати обробки неправильної форми не туди, куди потрібно. Щоб цього уникнути, можна перевіряти адресу сторінки, з якої були надіслані дані. Це можна зробити знову ж таки за допомогою функції getenv():

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

10.4. Вирішення завдання

Сформулюємо завдання. Потрібно написати форму для реєстрації учасників семінара і після реєстрації відправити їм повідомлення.

Хоча це повідомлення є універсальним листом, але воно буде не набагато відрізнятися від того листа, що кінець кінцем з’явиться на екрані. Початковий варіант форми реєстрації вже наводився раніше.

Змінимо його таким чином, щоб кожен зареєстрований міг вибрати скільки завгодно лекцій для відвідування, і не будемо підтверджувати одержання реєстраційної форми form_final.html.

Приклад 10.5

```
<h2>Форма для реєстрації учасників семінара</h2>
<form action="1.php" method=POST>
Ім'я <br><input type=text name="first_name"
    value="Уведіть Ваше ім'я"><br>
Прізвище <br><input type=text name="last_name"><br>
E-mail <br><input type=text name="email"><br>
<p> Виберіть секцію, що 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="Скасувати">
</form>
```

Тут все дуже просто і зрозуміло. Єдине, що можна відмітити, – це спосіб передачі значень елемента checkbox. Коли пишемо в імені елемента kurs[], це означає, що перший елемент checkbox буде записаний у перший елемент масиву kurs, другий – у другий елемент масиву і т.д.

Можна, звичайно, просто дати різні імена елементам checkbox, але це ускладнить обробку даних, якщо курсів буде багато.

Програма, що все це буде розбирати й обробляти, називається 1.php (форма посилається саме на цей файл, що записаний в її атрибуті action). За умовчанням використовується для передачі метод GET, але було вказано POST. З отриманих від зареєстрованого учасника відомостей програма генерує відповідне повідомлення.

Якщо учасник вибрав якісь лекції, то йому виводиться повідомлення про час проведення і дані про лекторів. А якщо учасник нічого не вибрав, то виводиться повідомлення про наступний семінар. У прикладі 10.6 подана програма 1.php, що обробляє форму form_final.html.

Приклад 10.6

```
<?

// Створимо масиви відповідностей лекція – час
// проведення – 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 Петрович", "Unix"=>"Семен
Семенович");

define("SIGN","З повагою адміністрація");
    // визначаємо підпис листа як константу
$date1 = "12 вересня 2012 р.";
    // задаємо дату проведення семінара
$date2 = "26 вересня 2012 р.";
    // задаємо дату проведення наступного семінара
    //починаємо формувати текст повідомлення
$str = "Здрастуйте, шановний " . $_POST["first_name"]
    . " " . $_POST["last_name"]."!<br>";
$str .= "<br>Повідомляємо Вам, що ";
$kurses = $_POST["kurs"];   // збережемо в цієї змінної
                            // список обраних секцій
if (!isset($kurses)) { // якщо не обрана жодна секція
    $event = "наступний семінар";
    $str .= "$event відбудеться $date2 ". "<br>";
} else { // якщо хоча б одна секція обрана
    $event = "обрані Вами секції відбудуться $date1 <ul>";
        //функція count обчислює кількість елементів у масиві
    $lect = "";
    for ($i=0;$i<count($kurses);$i++){
                            // для кожної обраної секції
        $k = $kurses[$i]; // запам'ятовуємо назву
        $lect = $lect . "<li>лекція по $k в $times[$k]";
                            // становимо повідомлення
        $lect .= " (Ваш лектор, $lectors[$k])";
    }
    $event = $event . $lect . "</ul>";
    $str .= "$event";
}
```

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

Висновки

У даному розділі розглянуто такі основні питання:

– основи моделі клієнт-серверні технології;
– мережеві протоколи;
– способи передачі даних на сервер;
– форми запиту клієнта;
– методи запитів.

Контрольні питання

1. Дайте характеристику серверу і клієнту?
2. Яким чином сервер і клієнт взаємодіють один з одним?
3. Розкажіть про типи серверів?
4. Що таке HTTP?
5. Що являє собою форма запиту клієнта?
6. Чим відрізняються повна та скорочена форми запиту клієнта?
7. Які методи доступу існують?
8. Яким чином можна обробляти запити за допомогою мови PHP?

11. РОБОТА З ФАЙЛАМИ

Навчальною метою розділу є ознайомлення студентів з особливостями роботи з файлами мовою PHP.

У результаті вивчення даного розділу студенти повинні знати:

- принципи роботи операцій відкриття і закриття з'єднання з файлами;
- організацію запису у файли;
- функції читання з файлів і їх видалення;

уміти:

- завантажувати файли на сервер.

11.1. Відкриття файлів

Мова PHP, на жаль, взагалі не має функції, що призначена саме для створення файлів. Більшість функцій працюють з уже існуючими файлами у файловій системі сервера. Є кілька функцій, які дозволяють створювати тимчасові файли, або, що те саме, файли з унікальним для поточної директорії ім'ям.

Тому для того, щоб створити звичайний файл, потрібно скористатися функцією, що відкриває локальний або вилучений файл.

Називається ця функція fopen().

Що значить «відкриває файл»? Це означає, що функція fopen зв'язує даний файл з потоком керування програми. Причому зв'язування буває різним залежно від того, що необхідно робити з цим файлом: читати його, записувати в нього дані або виконувати обидві дії.

Синтаксис цієї функції такий:

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

У результаті роботи ця функція повертає покажчик (типу ресурс) на відкритий нею файл. Як параметри цієї функції передаються: ім'я_файлу, який потрібно відкрити, тип_доступу до файлу (визначається тим, що ми збираємося робити з ним) і, можливо, параметр, що визначає, чи шукати зазначений файл в include_path.

Обговоримо докладніше кожний з цих трьох параметрів.

Параметр ім'я_файлу повинен бути рядком, що містить правильне локальне ім'я файлу або URL-адресу файлу в мережі. Якщо ім'я_файлу починається з вказівки протоколу доступу (наприклад, http://... або ftp://...), то інтерпретатор вважає це ім'я адресою URL і шукає оброблювач зазначеного в URL протоколу.

Коли оброблювач знайдений, то мова PHP перевіряє, чи дозволено працювати з об'єктами URL, як із звичайними файлами (директива allow_url_fopen).

Після дії директиви allow_url_fopen=off функція fopen викликає помилку, а потім генерується попередження.

Якщо ім'я_файлу не починається з протоколу, то вважається, що зазначено ім'я локального файлу. Для відкриття локального файлу потрібно, щоб мова PHP мала відповідні права доступу до цього файлу.

Параметр use_include_path, установлений у значення 1 або TRUE, змушує інтерпретатор в include_path шукати зазначений в fopen() файл.

Нагадаємо, що include_path – це директива з файлу налаштувань мови PHP, що задає список директорій, у яких можуть перебувати файли для включення. Крім функції fopen(), він використовується функціями include() і require().

Параметр тип_доступу може приймати одне із значень, наведених в табл. 11.1.

Таблиця 11.1

Значення параметра тип_ доступу

Тип доступу	Опис
r	Відкриває файл тільки для читання, покажчик позиції файлу встановлюється на його початок
r+	Відкриває файл для читання і запису, покажчик позиції файлу встановлюється на його початок
w	Відкриває файл тільки для запису, покажчик позиції файлу встановлюється на його початок і зменшує файл до нульової довжини. Якщо файл не існує, то намагається створити його
w+	Відкриває файл для читання і запису, покажчик позиції файлу встановлюється на його початок і зменшує файл до нульової довжини. Якщо файл не існує, то намагається створити його
a	Відкриває файл тільки для запису, покажчик позиції файлу встановлюється на його кінець. Якщо файл не існує, то намагається створити його
a+	Відкриває файл для читання і запису, покажчик позиції файлу встановлюється на його кінець. Якщо файл не існує, то намагається створити його
x	Створює і відкриває файл тільки для запису, покажчик позиції файлу встановлюється на його початок. Якщо файл уже існує, то fopen() повертає false і тоді генерується попередження. Якщо файл не існує, то робиться спроба створити його. Цей тип доступу підтримується, починаючи з версії мови PHP 4.3.2 і працює тільки з локальними файлами
x+	Створює і відкриває файл для читання і запису, покажчик позиції файлу встановлюється на його початок. Якщо файл уже існує, то fopen() повертає false і відразу 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

Отже, щоб створити файл, потрібно відкрити неіснуючий файл для запису.

Приклад 11.1

```
<?php
/* Відкриває на запис файл my_file.html, якщо він існує,
або створює новий файл із таким самим ім'ям,
якщо його ще немає */
$h = fopen("my_file.html","w");

// Відкриває для запису і читання або створює
// файл another_file.txt у директорії dir */
$h = fopen("dir/another_file.txt","w+");

// Відкриває для читання файл, що перебуває за вказаною адресою
$h = fopen("http://www.nmu.org.ua/dir/file.php","r");
?>
```

Коли створюється файл, потрібно враховувати, під якою операційною системою (ОС) він працює і буде читатися.

Справа в тому, що різні ОС по-різному відзначають кінець рядка.

В Unix-подібних ОС кінець рядка позначається символом \n, у системах типу Windows – \r\n.

Windows пропонує спеціальний прапор t для заміни символів кінця рядка систем типу Unix на свої символи кінця рядка. На противагу цьому існує прапор b, який використовується найчастіше для бінарних файлів, завдяки якому така заміна не відбувається. Використовувати ці прапори можна, якщо дописати їх після останнього символа обраного типу доступу до файлу.

Наприклад, якщо відкривати файл на читання, замість r треба використати rt, щоб перекодувати всі символи кінця рядка в \r\n. Якщо не використовувати прапор b при відкритті бінарних файлів, то можуть з'являтися помилки, пов'язані із зміною вмісту файлу. З метою перенесення програми на різні платформи рекомендується завжди використовувати прапор b для відкриття файлів функцією fopen().

Що робити, якщо відкрити або створити файл за допомогою fopen() не вдається?

У цьому випадку мова PHP генерує попередження, а функція fopen() повертає як результат своєї роботи значення false. Такого роду попередження можна «придушити» (заборонити), скориставшись символом @ .

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.

11.2. Закриття з'єднання з файлом

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, установлене з цим файлом функцією fopen(), потрібно закрити. Для цього використають функцію fclose(). Синтаксис її такий:

```
fclose (покажчик на файл)
```

Ця функція повертає TRUE, якщо з'єднання закрите, і FALSE – якщо відкрите. Параметр цієї функції повинен указувати на файл, 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 програми не відбудеться. Але в цілому для сервера це може мати серйозні наслідки.

Наприклад, сторонній користувач може скористатися відкритим з'єднанням і записати у файл вірус, не говорячи вже про зайву витрату ресурсів сервера.

Тому рекомендується завжди закривати з'єднання з файлом після виконання необхідних дій.

11.3. Запис даних у файл

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

Синтаксис у неї такий:

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

Ця функція записує вміст рядка у файл, на який указує `покажчик на файл`. Якщо зазначено додатковий аргумент `довжина`, то запис закінчується після того як записана кількість символів, що дорівнює значенню цього аргументу, або коли буде досягнутий кінець рядка.

У результаті своєї роботи функція fwrite() повертає кількість записаних байтів або false – у випадку помилки.

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

Приклад 11.2

```
<?php
$h = fopen("my_file.html","w");
$text = "Цей текст запишемо у файл.";
```

```
if (fwrite($h,$text))
  echo "Запис пройшов успішно";
else
  echo "Відбулася помилка при записі даних";
fclose($h);
?>
```

У результаті роботи цієї програми в браузері буде виведено 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 було виконано цю програму, всі дані цього файлу були б вилучені.

Напишемо таку програму.

Приклад 11.3

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

У результаті роботи цієї програми до рядка, що вже існує у файлі my_file.html, додасться ще сім символів з рядка, що знаходиться у змінній $add_text, тобто слово «Додамо».

Функція fwrite() має псевдонім fputs(), використовуваний у такий же спосіб, що і сама функція.

Розглянемо далі, які методи читання даних з файлу пропонує мова PHP.

11.4. Читання даних з файлу

Якщо необхідно прочитати дані з існуючого файлу, однієї функції fopen(), як і у випадку із записом даних, недостатньо. Вона лише повертає покажчик на відкритий файл, але не зчитує жодного рядка з цього файлу.

Тому для того, щоб прочитати дані з файлу, потрібно скористатися однією зі спеціальних функцій: file, readfile, file_get_contents, fread, fgets і т.п.

11.4.1. Функція fread

Ця функція здійснює читання даних з файлу. Її можна використовувати і для читання даних з бінарних файлів, не побоюючись їхнього ушкодження.

Синтаксис fread() такий:

```
string fread (покажчик на файл, довжина)
```

При виклику цієї функції відбувається читання даних відповідно параметру `довжина` (у байтах) з файлу, на який указує `покажчик на файл`.

Параметр `покажчик на файл` повинен бути реально існуючою змінною типу ресурс, що містить у собі зв'язок з файлом, відкритим, наприклад, за допомогою функції fopen().

Читання даних відбувається доти, поки не зустрінеться символ кінця файлу або поки не буде прочитане зазначена параметром `довжина` кількість байтів.

У результаті роботи функція fread() повертає рядок із зчитаною з файлу інформацією.

У цій функції параметр довжина – обов'язковий. Отже, якщо необхідно ввести весь файл у рядок, потрібно знати його довжину.

Мова PHP дозволяє обчислити довжину зазначеного файлу. Для цього потрібно скористатися функцією filesize(ім'я файлу). У випадку помилки ця функція поверне false. На жаль, її можна використовувати тільки для одержання розміру локальних файлів.

Приклад 11.4

```
<?php
$h = fopen("my_file.html","r+");
// відриваємо 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 браузера
?>
```

Для того, щоб зчитати вміст бінарного файлу, наприклад зображення, у таких ОС, як Windows, рекомендується відкривати файл за допомогою прапора rb або йому подібного, який утримує символ b наприкінці.

Функція filesize() записує результати своєї роботи у проміжну пам'ять. Якщо змінити вміст файлу my_file.html і знову запустити наведену раніше програму, то результат її роботи не зміниться. Більше того, якщо запустити скрипт, що зчитує дані з цього файлу за допомогою іншої функції (наприклад fgetss), то результат може виявитися таким, як і до зміни файлу.

Щоб цього уникнути, потрібно очистити статичну пам'ять (cache), додавши в код програми команду clearstatcache(); .

11.4.2. Функція fgets

За допомогою функції fgets() можна зчитати з файлу рядок тексту. Синтаксис цієї функції такий самий, як і в fread(), за винятком того, що довжину зчитуваного рядка вказувати необов'язково:

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

У результаті роботи функція fgets() повертає рядок довжиною зазначеною параметром `довжина` кількість байт із файлу, на який указує `покажчик на файл`.

Читання закінчується, якщо прочитано певну кількість символів і зустрівся символ кінця рядка або кінець файлу.

Нагадаємо, що у мові PHP один символ – це один байт.

Якщо довжина зчитуваного рядка не зазначена (ця можливість з'явилася, починаючи з версії мови PHP 4.2.0), то зчитується 1 Кбайт (1024 байт) тексту або, що теж саме, 1024 символи.

Починаючи з версії мови PHP 4.3, якщо параметр довжина не заданий, зчитується рядок цілком. У випадку помилки функція fgets() повертає false. Для версій мови PHP, починаючи з мови PHP 4.3, ця функція безпечна для двійкових файлів.

Приклад 11.5

```
<?php
$h = fopen("my_file.html","r+");
$content = fgets($h,2);
// зчитує 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 символ кінця файлу.

Мова PHP має спеціальну функцію, що перевіряє, чи вказує покажчик позиції файлу на кінець файлу. Це булева функція feof(), у якій як параметр передається покажчик на з'єднання з файлом.

У прикладі 11.6 показано, як можна зчитати всі рядки файлу my_file.html.

Приклад 11.6

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

11.4.3. Функція fgetss

Існує різновид функції fgets() – функція fgetss(). Вона теж дозволяє зчитувати рядок із зазначеного файлу, але при цьому видаляє з нього всі html-теги, що зустрілися, за винятком, може бути, деяких.

Синтаксис fgetss() такий:

```
string fgetss(покажчик на файл,
   довжина [, припустимі теги])
```

Тут аргумент довжина обов'язковий.

Нехай є файл my_file.html наступного змісту:

```
<h1>Гарно того вчити, хто хоче знати.</h1>
<b>Людей питай, а свій розум май.</b>
Хто мови своєї цурається,  <i> хай сам себе стидається. </i>.
```

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

Приклад 11.7

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

У результаті одержимо:

```
Гарно того вчити, хто хоче знати.
Людей питай, а свій розум май.
Хто мови своєї цурається, хай сам себе стидається.
```

11.4.4. Функція fgetc

Якщо можна зчитувати інформацію з файлу порядково, то можна зчитувати її і посимвольно. Для цього призначена функція fgetc().

Синтаксис у неї такий:

```
string fgetc (покажчик на файл)
```

Ця функція повертає символ з файлу, на який посилається `покажчик на файл`, і значення, що обчислює як FALSE, якщо зустрінуто символ кінця рядка.

От так, наприклад, можна зчитати файл по одному символу.

Приклад 11.8

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

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

11.4.5. Функция readfile

Синтаксис цієї функції такий:

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

Функція readfile() зчитує файл, ім'я якого передане їй як параметр ім'я_файлу, і виводить його на екран. Якщо додатковий аргумент use_include_path має значення TRUE, то пошук файлу із заданим ім'ям виконується і за директоріями, що входять в include_path.

В програму ця функція повертає кількість зчитаних байтів (символів) файлу, а у випадку помилки – FALSE. Повідомлення про помилку в цій функції можна “видалити” оператором @ .

Наступна програма виведе на екран файл my_file1.html і його розмір, якщо він існує. У противному разі виведеться повідомлення про помилку – рядок "Error in readfile".

Приклад 11.9

```
<?php
$n = @readfile ("my_file1.html");
/* виводить на екран файл і
записує його розмір у 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
  // кількість зчитаних символів
?>
```

За допомогою функції readfile() можна читати файли віддалено, указуючи їхню URL-адресу як ім'я файлу, якщо ця опція не відключена в налаштуваннях сервера.

Відразу ж виводити файл на екран не завжди зручно. Іноді потрібно записати інформацію з файлу в змінну, щоб надалі виконати з нею які-небудь дії. Для цього можна використати функцію file() або file_get_contents().

11.4.6. Функція file

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

```
array file (ім'я_файлу[, use_include_path])
```

Який масив повертає ця функція?

Кожен елемент даного масиву є рядком у файлі, інформацію з якого ми зчитуємо (його ім'я задане аргументом ім'я_файлу). Символ нового рядка теж включається в кожний з елементів масиву.

У випадку помилки функція file(), як і всі вже розглянуті, повертає false.

Додатковий аргумент use_include_path знову ж визначає, шукати чи ні даний файл у директоріях include_path.

Відкривати вилучені файли за допомогою цієї функції теж можна, якщо не заборонено сервером. Починаючи з версії мови PHP 4.3, робота з бінарними файлами за допомогою цієї функції стала безпечніше.

Наприклад, є файл my_file.html такого змісту:

```
<h1> Гарно того вчити, хто хоче знати.</h1>
<b> Людей питай, а свій розум май.</b>
```

Прочитаємо його за допомогою функції file().

Приклад 11.10

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

У результаті на екран буде виведене таке повідомлення:

```
0: <h1> Гарно того вчити, хто хоче знати.</h1>
1: <b> Людей питай, а свій розум май.</b>
```

11.4.7. Функція file_get_contents

У версіях мови PHP, починаючи з 4.3, з'явилася можливість зчитувати зміст файлу порядково. Робиться це за допомогою функції file_get_contents().

Подібно до двох попередніх функцій, як параметри вона приймає значення імені файлу і, можливо, вказівку шукати його в директоріях include_path.

То ж синтаксис її такий:

```
string file_get_contents (ім'я_файлу[, use_include_path])
```

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

11.4.8. Перевірка існування файлу

Що робити, якщо файлу з яким намагаються проробити операції читаня або запису, не існує?

Очевидно, що в такому випадку жодна з розглянутих функцій працювати не буде й інтерпретатор мови PHP видасть повідомлення про помилку.

Щоб відслідковувати такого роду помилки, можна використовувати функції file_exists(), is_writable(), is_readable().

Функция file_exists:

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

Функція file_exists() перевіряє, чи існує файл або директорія, ім'я якої передане їй як аргумент.

Якщо директорія або файл у файловій системі сервера існує, то функція повертає TRUE, у противному разі – FALSE. Результат роботи цієї функції записується у проміжну пам'ять (кешується).

Відповідно можна очистити цю пам'ять, як вже відзначалося, за допомогою функції clearstatcache().

Для нелокальних файлів використати функцію file_exists() не можна.

Приклад 11.11

```
<?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 крім перевірки існування файлу потрібно довідатися ще, чи дозволено записувати інформацію в цей файл, слід використати функцію is_writable() або її псевдонім – функцію is_writeable().

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

Ця функція повертає TRUE, якщо файл (або 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).

Якщо крім перевірки існування файлу потрібно довідатися ще, чи дозволено читати інформацію з нього, потрібно використати функцію is_readable().

```
bool is_readable (ім'я файлу)
```

Ця функція працює подібно функції is_writable().

Приклад 11.12

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

?>
```

11.5. Видалення файлу

Для того, щоб видалити файл у мові PHP, потрібно скористатися функцією unlink().

Синтаксис цієї функції такий:

```
bool unlink (ім'я_файлу)
```

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

Приклад 11.13

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

11.6. Завантаження файлу на сервер

Тепер вирішимо більш складне і часто виникаюче на практиці завдання завантаження файлу на сервер. Перше, що потрібно 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 у 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>
```

Відзначимо, що у форму додане сховане поле, що містить у собі максимальний припустимий розмір завантажуваного файлу у байтах. При спробі завантажити файл, розмір якого більше від зазначеного в цьому полі значення, буде зафіксована помилка. У браузері створена форма буде виглядати як рядок для введення тексту з додатковою кнопкою для вибору файлу з локального диска (рис 11.1).

Рис. 11.1. Приклад форми для завантаження файлу на сервер

Тепер потрібно написати програму, що буде обробляти отриманий файл.

Вся інформація про завантажений на сервер файл знаходиться у глобальному масиві $_FILES. Цей масив з'явився, починаючи з версії мови PHP 4.1.0. Якщо включено директиву register_globals, то для отримання значення переданих змінних достатньо вказати тільки їх імена.

Якщо завантажити з комп'ютера-клієнта файл із ім'ям critics.htm розміром 15136 байт, то програма з єдиною командою print_r($_FILES) виведе на екран таке:

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

Масив $_FILES завжди має такі елементи:

$_FILES['myfile']['name'] (ім'я, що мав файл на комп'ютері клієнта);

$_FILES['myfile']['type'] (mime-тип відправленого файлу, якщо браузер надав цю інформацію. У цьому 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'являється при завантаженні).

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

Якщо включено директиву register_globals=On, то доступні також додаткові змінні, такі як $myfile_name, що еквівалентно $_FILES['myfile']['name'], і т.п.

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

0 – помилки не відбулося, файл завантажений успішно;

1 – завантажується файл, що перевищує розмір, установлений директивою upload_max_filesize у файлі налаштувань php.ini;

2 – завантажується файл, що перевищує розмір, установлений елементом MAX_FILE_SIZE форми html;

3 – файл був завантажений частково;

4 – файл не був завантажений.

За умовчанням завантажені файли зберігаються у тимчасовій директорії сервера, якщо інша директорія не зазначена за допомогою опції upload_tmp_dir у файлі налаштувань php.ini. Перемістити завантажений файл у потрібну директорію можна за допомогою функції move_uploaded_file().

Функція move_uploaded_file() має такий 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 за допомогою механізму 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 зроблено не буде і move_uploaded_file() поверне FALSE. Те саме відбудеться, якщо файл із якихось причин не може бути переміщений. У цьому випадку інтерпретатор виведе відповідне попередження.

Якщо файл, заданий параметром `місце_призначення` існує, то функція move_uploaded_file() перезапише його.

Приклад 11.14

```
<?php

// Будемо зберігати файли, що завантажуються, в цю директорію
$uploaddir = 'c:/uploads/';

// Ім'я файлу залишимо незмінним
$destination = $uploaddir . $_FILES['myfile']['name'];

print "<pre>";

// Переміщаємо файл із тимчасової папки в обрану
// директорію для зберігання
if (move_uploaded_file( $_FILES['myfile']['tmp_name'],
$destination)) {
    print "Файл успішно завантажений <br>";
} else {
  echo "Відбулася помилка при завантаженні файлу.
    Деяка додаткова інформація:<br>";
    print_r($_FILES);
}
print "</pre>";

?>
```

Висновки

У даному розділі розглянуто такі основні питання:

- створювання файлів;
- запис даних у файли;
- зчитування з файлів інформації різними способами;
- перевірка існування і доступність файлів для запису і читання;
- завантаження файлів на сервер.

Контрольні питання

1. Яка функція виконує відкриття файлів?
2. Яка функція виконує закриття з’єднання з файлом?
3. Яка функція виконує запис даних у файл?
4. Яка функція виконує видалення файлу?
5. Яка функція виконує завантаження файлу на сервер?
6. Які є варіанти функції читання даних з файлу?

ЧАСТИНА 4. БАЗИ ДАНИХ
12. СИСТЕМИ КЕРУВАННЯ БАЗАМИ ДАНИХ

Навчальною метою розділу є ознайомлення студентів з базами даних.

У результаті вивчення даного розділу студенти повинні знати:

- основні поняття баз даних;
- особливості ідентифікації записів у таблицях баз даних;
- призначення індексування баз даних;
- функції системи керування базами даних (СКБД);
- особливості підготовки до роботи СКБД MySQL.

12.1. Основні поняття баз даних

12.1.1. Види СКБД

Протягом життя людям необхідно зберігати інформацію.

Наприклад, для зберігання номерів телефонів друзів і планування свого часу можно використовувати записну книжку.

Телефонна книга міста має інформацію про людей, які живуть в цьому місті.

Все це свого роду бази даних.

База даних – це сукупність пов'язаних між собою даних, організованих за певними правилами, що передбачає загальні принципи їх опису, зберігання і маніпулювання, незалежно від прикладних програм.

Як же у них зберігаються дані.

Наприклад, телефонна книга являє собою таблицю (табл. 12.1).

У цій таблиці дані – це номери телефонів, адреси і ПІБ, наприклад «Чумак Гнат Петрович, 370-43-12» і т.п., а назви стовпців цієї таблиці, тобто «ПІБ», «Номер телефону» і «Адреса» задають зміст цих даних, їхню семантику.

Таблиця 12.1

Фрагмент телефонної книги

ПІБ	Номер телефону	Адреса
Чумак Гнат Петрович	370-43-12	вул. О. Гончара, 12, кв. 43
Олійник Сергій Федорович	370-32-34	просп. К. Маркса, 32, кв. 45

Якщо записів у цій таблиці не дві, а дві тисячі, то при створенні цього довідника і у разі виправлення якихось помилок треба витратити багато часу.

Ймовірно, складно буде знайти і виправити цю помилку вручну. Потрібно скористатися якимись засобами автоматизації.

Робота з базами даних здійснюється за допомогою системи керування базами даних. У порівнянні з текстовими базами даних електронні СКБД мають величезну кількість переваг, від можливості швидкого пошуку інформації, взаємозв'язку даних між собою до використання цих даних у різних прикладних програмах і одночасному доступі до даних декількох користувачів.

СКБД забезпечує централізоване керування й організацію доступу до них різних користувачів.

Функції будь-якої СКБД включають:

- створення, видалення і зміну бази даних (БД);
- додавання, зміну, видалення, призначення прав користувача;
- внесення, видалення і зміну даних у БД (таблиць і записів);
- вибірку даних із БД.

До перших двох функцій мають доступ тільки адміністратори СКБД або привілейовані користувачі.

Як зберігати дані та за якими правилами вони повинні бути організовані?

Способів (моделей) подання або зберігання даних існує безліч. Найбільш популярні – об'єктна і реляційна моделі даних.

В основу об'єктної моделі покладена концепція об’єктно-орієнтованого програмування, у якій дані подані у вигляді набору об'єктів і класів, пов'язаних між собою родинними відносинами, а робота з об'єктами здійснюється за допомогою схованих (інкапсульованих) у них методів.

Приклади об'єктних СКБД: Cache, GemStone (від Servio Corporation), ONTOS (ONTOS) і т.п.

Автором реляційної моделі вважається Е. Кодд, який першим запропонував використовувати для обробки даних апарат теорії множин і показав, що будь-яке подання даних зводиться до сукупності двовимірних таблиць особливого виду.

Таким чином, реляційна база даних являє собою набір таблиць (таких, що наведені вище), пов'язаних між собою. Рядок у таблиці відповідає сутності реального світу (у наведеному вище прикладі це інформація про людину).

Приклади реляційних СКБД: MySql, PostgreSql і т.п.

Останнім часом виробники СКБД прагнуть з'єднати цих два підходи і пропонують объектно-реляційону модель подання даних. Приклади таких СКБД – IBM DB2 for Common Servers, Oracle8 і т.п.

Розглянемо більш детально застосування реляційних СКБД, а також важливі поняття з цієї області – ключі та індексування.

12.1.2. Ключі таблиць баз даних

Для початку подумаємо над таким питанням: яку інформацію потрібно дати про людину, щоб співрозмовник точно сказав, що це саме та людина, сумнівів бути не може, другої такої немає?

Повідомити прізвище, мабуть, недостатньо, оскільки існують люди з однаковими прізвищами. Також можно приблизно пояснити, про кого йде мова, наприклад, згадати якийсь вчинок, що зробила та людина, або ще щось.

Комп'ютер же такого пояснення не зрозуміє, йому потрібні чіткі правила, як визначити, про кого йде мова. У СКБД для вирішення такого завдання ввели поняття первинного ключа.

Первинний ключ (primary key, PK) – мінімальний набір полів, унікально ідентифікуючий запис у таблиці. Виходить, що первинний ключ – це, по-перше, набір полів таблиці, по-друге, кожен набір значень цих полів повинен визначати єдиний запис (рядок) у таблиці й, по-третє, цей набір полів повинен бути мінімальним з усіх з такою самою властивістю.

Оскільки первинний ключ визначає тільки один унікальний запис, то ніякі два записи таблиці не можуть мати однакових значень первинного ключа.

Наприклад, у таблиці (табл. 12.1) ім’я та адреса дозволяють однозначно виділити запис про людину. Якщо ж говорити загалом, без зв'язку з розв'язуваним завданням, то такі знання не дозволяють точно вказати на єдину людину, оскільки існують люди з однаковими прізвищами, які мають однакову адресу, крім міста. Вся справа в границях, які задаються.

Якщо вважаємо, що знання імені, номера телефону та адреси без вказівки міста для наших цілей достатньо, то все прекрасно, тоді поля імені та адреси можуть утворювати первинний ключ. У кожному разі проблему створення первинного ключа вирішує той, хто проектує базу даних (розробляє структуру зберігання даних).

Вирішенням цієї проблеми може стати або виділення характеристик, які природно визначають запис у таблиці (завдання так називаного логічного, або природного, PK), або створення додаткового поля, призначеного саме для однозначної ідентифікації записів у таблиці (завдання так званого сурогатного, або штучного, PK). Прикладом логічного первинного ключа є номер паспорта в базі даних або ПІБ і адреса в телефонній книзі (табл. 12.1).

Для завдання сурогатного первинного ключа в таблицю можна додати поле id (ідентифікатор), значенням якого буде ціле число, унікальне для кожного рядка таблиці. Використання таких сурогатних ключів має сенс, якщо природний первинний ключ являє собою великий набір полів або його виділення нетривіально.

Крім однозначної ідентифікації запису, первинні ключі використовуються для організації зв'язків з іншими таблицями.

Наприклад (рис. 12.1), є три таблиці: з інформацією про історичних особистостей (Persons), з інформацією про їхні винаходи (Artifacts) та із зображеннями як особистостей, так і артефактів (Images).

Первинним ключем у всіх цих таблицях є поле id (ідентифікатор). У таблиці Artifacts є поле author, у якому записаний ідентифікатор, присвоєний авторові винаходу в таблиці Persons. Кожне значення цього поля є зовнішнім ключем для первинного ключа таблиці Persons. Крім того, у таблицях Persons і Artifacts є поле photo, що посилається на зображення в таблиці Images. Ці поля також є зовнішніми ключами для первинного ключа таблиці Images і встановлюють однозначний логічний зв'язок Persons-Images і Artifacts-Images.

Отже, якщо значення зовнішнього ключа photo в таблиці особистості дорівнює 10, то це означає, що фотографія цієї особистості має id=10 у таблиці зображень.

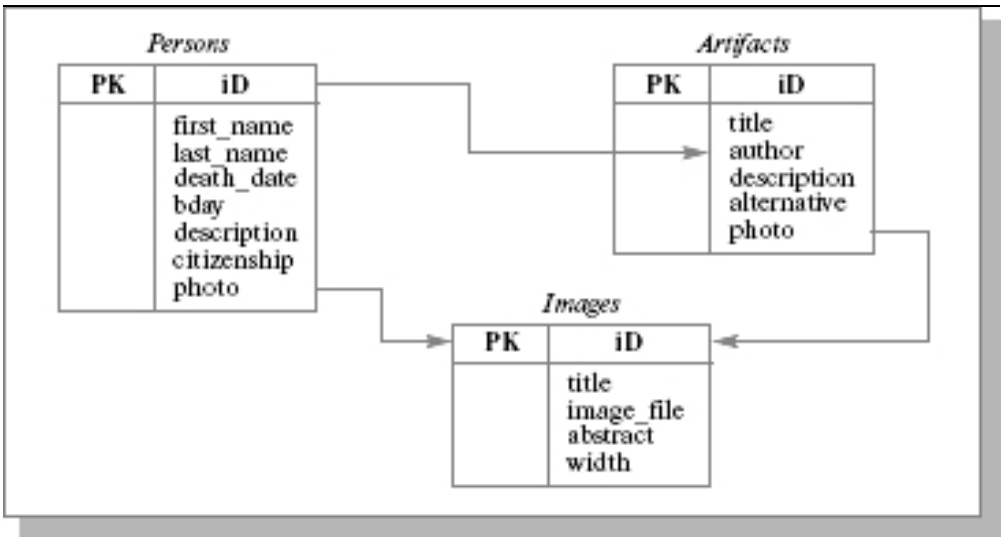


Рис. 12.1. Приклад використання первинних ключів для організації зв'язків з іншими таблицями

Таким чином, зовнішні ключі використовують для організації зв'язків між таблицями бази даних (батьківськими і дочірніми) і для підтримки обмежень посилальної цілісності даних.

12.1.3. Індексування таблиць баз даних

Одне з основних завдань, що виникають при роботі з базами даних, – це завдання пошуку. При цьому, оскільки інформації в базі даних, як правило, утримується багато, перед програмістами постає завдання не просто пошуку, а ефективного пошуку, тобто пошуку за порівняно невеликий час і з достатньою точністю.

Для цього (для оптимізації продуктивності запитів) роблять індексування деяких полів таблиці. Використовувати індекси доцільно для швидкого пошуку рядків із вказаним значенням одного стовпця.

Без індексу читання таблиці здійснюється по всій таблиці, починаючи з першого запису, поки не будуть знайдені відповідні рядки. Чим більше таблиця, тим більші витрати часу. Якщо ж таблиця містить індекс відповідно до розглянутих стовпців, то база даних може швидко визначити позицію для пошуку в середині файлу даних без перегляду всіх даних.

Це відбувається тому, що база даних поміщає проіндексовані поля ближче в пам'яті, так щоб можна було швидше знайти їхні значення. Для таблиці, що містить 1000 рядків, це буде як мінімум в 100 разів швидше в порівнянні з послідовним переглядом всіх записів. Однак у випадку, коли необхідний доступ майже до всіх 1000 рядків, швидше буде послідовне читання.

Так що іноді індекси бувають тільки перешкодою.

Наприклад, якщо копіюється великий обсяг даних у таблицю, то краще не мати ніяких індексів. Однак у деяких випадках потрібно задіяти відразу кілька індексів (наприклад, для обробки запитів до часто використовуваних таблиць).

12.2. СКБД MySQL

12.2.1. Режими роботи програми MySQL

MySQL – це реляційна система керування базами даних. Дані в її базах зберігаються у вигляді логічно зв'язаних між собою таблиць, доступ до яких здійснюється за допомогою мови запитів SQL.

MySQL – вільно розповсюджувана система, тобто платити за її застосування не потрібно. Крім того, це досить швидка, надійна й, головне, проста у використанні СКБД, що цілком підходить для не занадто глобальних проектів.

Працювати з MySQL можна не тільки в текстовому режимі, але й у графічному.

Існує дуже популярний візуальний інтерфейс (до речі, написаний на мові PHP) для роботи з цією СКБД. Називається він PhpMyAdmin. Цей інтерфейс дозволяє значно спростити роботу з базами даних в MySQL.

У текстовому режимі робота з базою даних виглядає просто як введення команд у командний рядок, а результати вибірок повертаються у вигляді відповідних таблиць.

PhpMyAdmin дозволяє користуватися всіма достоїнствами браузера, включаючи прокручування зображення, якщо воно не вміщається на екран.

Багато з базових SQL-функцій роботи з даними в PhpMyAdmin зведені до інтуїтивно зрозумілих інтерфейсів і дій, що нагадують перехід за допомогою посилань в Інтернет.

MySQL має три види індексів: PRIMARY, UNIQUE і INDEX, а слово ключ (KEY) використовується як синонім слова індекс (INDEX).

Всі індекси зберігаються в пам'яті у вигляді B-дерев.

PRIMARY – унікальний індекс (ключ) з обмеженням, тобто всі індексовані їм поля не можуть мати порожнього значення (тобто вони NOT NULL).

Таблиця може мати тільки один первинний індекс, який складається з одного або з декількох полів.

UNIQUE – ключ (індекс), що задає поля, які можуть мати тільки унікальні значення.

INDEX – звичайний індекс (як було описано вище).

В MySQL, крім того, можна індексувати рядкові поля відповідно до заданої кількості символів від початку рядка.

12.2.2. Підготовка до роботи програми СКБД MySQL

Разглянемо більш детально підготовку до роботи програми СКБД MySQL.

Спочатку установимо програму MySQL.

MySQL установлюється дуже просто – автоматично, пару раз необхідно натиснути OK і все.

Після цього треба перейти в директорію (для ОС Windows XP це c:\mysql\bin), де знаходяться файли mysql.exe, mysqld.exe і т.ін. Останній файл запускає Mysql-сервер.

У деяких ОС сервер запускається у вигляді сервісу. Після запуску сервера слід запустити mysql-клієнт (програму mysql.exe). Вона запускається без пароля.

Більше того, якщо набрати

```
shell> mysql.exe -u root
```

або

```
shell> mysql -u root mysql,
```

то можна одержати всі права адміністратора mysql-сервера.

Програма MySQL після установки має два недоліки: відсутність пароля для адміністратора СКБД та можливість обслуговування анонімних користувачів.

Виправимо ці недоліки.

Всі дані про користувачів MySQL зберігає в таблиці user у спеціальній базі даних mysql, доступ до якої має тільки адміністратор сервера.

Тому, щоб змінити який-небудь пароль, потрібно змінити цю таблицю. Пароль задається за допомогою функції PASSWORD, що кодує введені дані.

```
shell> mysql -u root mysql
mysql> UPDATE user SET Password=PASSWORD('new_password')
    WHERE user='root';
```

Крім зміни пароля адміністратора, потрібно ще видалити всіх користувачів, що не мають логіна (команда DELETE).

```
mysql> DELETE FROM user WHERE user='';
```

Далі виконаємо команду Flush Privileges, яка змушує вступити в дію зміни, що відбулися в системній базі даних (mysql).

```
mysql> FLUSH PRIVILEGES;
```

Тепер створимо базу даних, з якою будемо працювати (ще працюємо як адміністратор сервера):

```
mysql>create database book;
```

Як можна помітити, всі команди в MySQL закінчуються крапкою з комою. Якщо цей знак не поставити, то програма буде нагадувати, поки це не буде зроблено:

```
mysql> show tables
    ->
    ->
```

Тепер остання дія – створимо простого користувача (aks2011), надамо йому доступ до створеної бази даних і почнемо працювати.

```
mysql> GRANT ALL PRIVILEGES ON book.* TO aks2012@localhost
    IDENTIFIED BY 'qwerty123';
```

Команда GRANT наділяє користувача aks2012, який зайшов на сервер з цієї ж машини (з localhost) та ідентифікований паролем «qwerty123», певними правами (у цьому випадку всіма) на всі таблиці бази даних book.

Тепер можемо зайти як користувач aks2012 з відповідним паролем:

```
shell>mysql -u aks2012 -p
Enter password: *********
Welcome to the MySQL monitor!...
mysql>
```

Висновки

У даному розділі розглянуто такі основні питання:

- способи зберігання даних;
- ключі таблиць баз даних і їх призначення;
- види систем керування базами даних;
- функції систем керування базами даних;
- СКБД MySQL.

Контрольні питання

1. Що таке база даних?
2. Які є найбільш відомі способи організації баз даних?
3. Які основні функції повинна мати СКБД?
4. Що таке реляційна та об’єктна СКБД?
5. Для чого необхідні ключі в таблицях баз даних?
6. Для чого необхідно індексування в таблицях баз даних?
7. Які види індексів існують в СКБД MySQL?
8. Які є режими роботи СКБД MySQL?
9. Які дії необхідно виконати для підготовки до роботи СКБД MySQL?
10. Які дії необхідно виконати для видалення в СКБД MySQL всіх користувачів, що не мають логіна?

13. ОСНОВИ МОВИ SQL

Навчальною метою розділу є ознайомлення студентів з мовою SQL.

У результаті вивчення даного розділу студенти повинні знати:

- призначення мови SQL;
- оператори відкриття, видалення таблиць і зміни їх структури мови SQL;
- оператори роботи з елементами таблиць мови SQL.

SQL – структурована мова запитів, що дозволяє створювати, редагувати і видаляти інформацію, що зберігається в базах даних, створювати нові бази даних і багато чого іншого. SQL є стандартом ANSI (Американський національний інститут стандартів) та ISO (Міжнародна організація по стандартизації).

Перш ніж що-небудь робити з даними, потрібно створити таблиці, у яких ці дані будуть зберігатися, навчитися змінювати структуру цих таблиць і видаляти їх, якщо буде потрібно.

Для цього мова SQL має оператори CREATE TABLE, ALTER TABLE і DROP TABLE.

13.1. Оператори відкриття, видалення таблиць і зміни їх структури

13.1.1. Оператор CREATE TABLE

Оператор CREATE TABLE створює таблицю із заданим ім'ям у поточній базі даних. Якщо немає активної поточної бази даних або зазначена таблиця вже існує, то виникає помилка виконання команди.

У версії MySQL 3.22 і більш пізніх ім'я таблиці може бути зазначене як ім'я_бази_даних, ім'я_таблиці. Ця форма запису працює незалежно від того, чи є зазначена база даних поточною.

У версії MySQL 3.23 для створення таблиці можна використати ключове слово TEMPORARY. Тимчасова таблиця автоматично видаляється після завершення з'єднання, а її ім'я дійсно тільки протягом даного з'єднання. Це означає, що у двох різних з'єднаннях можуть використовуватися тимчасові таблиці з однаковими іменами без конфлікту один з одним або з існуючою таблицею з тим самим ім'ям (існуюча таблиця ховається, доки не вилучена тимчасова таблиця).

У версії MySQL 4.0.2 для створення тимчасових таблиць необхідно мати привілею CREATE TEMPORARY TABLES.

У версії MySQL 3.23 і більш пізніх можна використовувати ключові слова IF NOT EXISTS для того, щоб не виникала помилка, якщо зазначена таблиця вже існує. Варто враховувати, що ідентичність структур цих таблиць при цьому не перевіряється.

Кожна таблиця являє собою набір певних файлів у директорії бази даних.

Синтаксис оператора CREATE TABLE:

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS]
     ім'я_таблиці [(визначення_стовпця,...)]
    [опції_таблиці] [select_вираження]
```

В операторі `визначення_стовпця` вказує, які стовпці повинні бути створені в таблиці.

Кожен стовпець таблиці може бути порожнім (NULL), мати значення за умовчанням, бути ключем або автоінкрементом. Крім того, для кожного стовпця обов'язково вказується тип даних, які будуть у ньому зберігатися.

Якщо не вказується ні NULL, ні NOT NULL, то стовпець інтерпретується відповідно до NULL.

Якщо поле позначають як автоінкремент (AUTO_INCREMENT), то його значення автоматично збільшується на одиницю щораз, коли відбувається додавання даних у таблицю.

Автоінкрементом у таблиці може бути позначено тільки одне поле, і при цьому воно обов'язково повинно бути проіндексовано. Послідовність AUTO_INCREMENT починається з 1.

Наявність автоінкремента є однією з особливостей MySQL. Формальний опис стовпця (`визначення_стовпця`) виглядає так:

```
Ім'я_стовпця тип [NOT NULL | NULL]
    [DEFAULT значення_за_умовчанням]
    [AUTO_INCREMENT][PRIMARY KEY]
    [reference_definition]
```

Тип стовпця може бути одним з таких:

- цілий (INT[(length)] [UNSIGNED] [ZEROFILL]);
- дійсний (REAL[(length,decimals)] [UNSIGNED] [ZEROFILL]);
- символьний (CHAR(length) [BINARY] і VARCHAR(length) [BINARY]);
- дата і час (DATE і TIME);
- для роботи з великими об'єктами (BLOB);
- текстовий (TEXT);
- перелічений безліч (ENUM(value1, value2, value3, ...) і SET(value1, value2, value3, ...)).

Повний список типів можна подивитися в документації MySQL.

Замість вказаних стовпців і їхніх властивостей у `визначення_стовпця` можна задавати списки ключових та індексних полів, обмеження і перевірки:

```
PRIMARY KEY (ім'я_індексованого_стовпця, ...)
```

або

```
KEY [ім'я_індексу] (ім'я_індексованого_стовпця, ...)
```

або

```
INDEX [ім'я_індексу] (ім'я_індексованого_стовпця, ...)
```

або

```
UNIQUE [INDEX] [ім'я_індексу] (ім'я_індексованого_стовпця, ...)
```

або

```
FULLTEXT [INDEX] [ім'я_індексу] (ім'я_індексованого_стовпця, ...)
```

або

```
[CONSTRAINT symbol]
FOREIGN KEY [ім'я_індексу]
    (ім'я_індексованого_стовпця, ...)
[reference_definition]
```

або

```
CHECK (expr)
```

При завданні всіх цих елементів указується список полів (стовпців), які будуть входити в індекс, ключ або обмеження.

Значення `ім'я_индексованого_стовпця` записується в такий спосіб:

```
ім'я_стовпця [(довжина_індексу)]
```

FOREIGN KEY, CHECK і REFERENCES насправді нічого не роблять в MySQL. Вони додані тільки для сумісності з іншими SQL-серверами.

Крім усього вказаного, при створенні таблиці можна додати деякі її властивості (`опції_таблиці`), наприклад такі:

- тип таблиці (TYPE = {BDB | HEAP | ISAM | InnoDB | MERGE | MRG_MYISAM | MYISAM });
- початкове значення лічильника автоінкремента (AUTO_INCREMENT = число);
- середня довжина рядків у таблиці (AVG_ROW_LENGTH = число);
- коментарі до таблиці, як рядок з 60 символів (COMMENT = "рядок");
- максимальне і мінімальне передбачуване число рядків (MAX_ROWS = число і MIN_ROWS = число).

І останній (знову ж опціональний) елемент команди CREATE – це `select_вираження`.

Синтаксис такий:

```
[IGNORE | REPLACE] SELECT ...
    (будь-яке коректне вираження SELECT)
```

Якщо при створенні таблиці у команді CREATE вказується елемент SELECT, то всі поля, які отримані вибіркою, додаються в цю таблицю.

Створимо таблицю Persons, структура якої була наведена на рис. 12.1.

```
mysql>CREATE TABLE Persons
    (id INT PRIMARY KEY AUTO_INCREMENT,
   first_name VARCHAR(50), last_name
    VARCHAR(100), death_date INT,
   description TEXT, photo INT,
   citienship CHAR(50) DEFAULT 'Ukraine');
```

За допомогою специфічної для MySql команди SHOW можна переглянути існуючі бази даних, таблиці в базі даних і поля в таблиці:

– показати всі бази даних:

```
mysql>SHOW databases;
```

– зробити поточну базу даних book і показати всі таблиці в ній:

```
mysql>use book;
mysql>show tables;
```

– показати всі стовпці в таблиці Persons:

```
mysql> show columns from Persons;
```

13.1.2. Оператор DROP TABLE

Оператор DROP TABLE видаляє одну або кілька таблиць. Всі табличні дані та визначення видаляються, так що при роботі з цією командою слід бути обережними.

Синтаксис оператора DROP TABLE:

```
DROP TABLE [IF EXISTS] ім'я_таблиці
    [, ім'я_таблиці, ...]
    [RESTRICT | CASCADE]
```

У версії MySQL 3.22 і більш пізніх можна використати ключові слова IF EXISTS, щоб попередити помилку, якщо зазначені таблиці не існують.

Опції RESTRICT і CASCADE дозволяють спростити перенесення програми з інших СКБД. У цей момент вони не задіяні.

```
mysql> DROP TABLE IF EXISTS Persons, Artifacts, test;
```

13.1.3. Оператор ALTER TABLE

Оператор ALTER TABLE дає можливість змінювати структуру існуючої таблиці. Наприклад, можна додавати або видаляти стовпці, створювати або знищувати індекси або перейменовувати стовпці або саму таблицю. Можна також змінювати коментар для таблиці та її тип.

Синтаксис оператора ALTER TABLE:

```
ALTER [IGNORE] TABLE ім'я_таблиці alter_specification
    [, alter_specification ...]
```

13.2. Оператори роботи з елементами таблиць

13.2.1. Оператор SELECT

Оператор SELECT застосовується для зчитування рядків, які обрані з однієї або декількох таблиць. Це означає, що з його допомогою задаються стовпці або вираження, які треба витягти (select_вираження), таблиці (table_references), з яких повинна виконуватися вибірка, і, можливо, умова (where_definition), якій повинні відповідати дані в цих стовпцях, а також порядок, у якому ці дані потрібно видати.

Також оператор SELECT можна використати для зчитування рядків, обчислених без посилання на яку-небудь таблицю.

Наприклад, щоб обчислити, чому дорівнює 2·2, потрібно просто написати

```
mysql> SELECT 2·2;
```

Спрощено структуру оператора SELECT можна подати в такий спосіб:

```
SELECT select_вираження1, select_вираження2,
    ...
    [FROM table_references
        [WHERE where_definition]
        [ORDER BY {число | ім'я_стовпця |
                формула}
            [ASC | DESC], ...]]
```

Квадратні дужки [ ] означають, що записаний у них оператор використовувати необов'язково, а вертикальна риса | вказує на можливі варіанти.

Після ключового слова ORDER BY указують ім'я стовпця, число (ціле беззнакове) або формулу і спосіб упорядкування (за зростанням – ASC або за спаданням – DESC). За умовчанням використається впорядкування за спаданням.

Коли в `select_вираження` записано «*», то це означає вибрати всі стовпці. Крім «*», можуть використовуватися функції типу max, min і avg. Наприклад, треба вибрати з таблиці Persons всі дані, для яких поле first_name має значення 'Олександр':

```
mysql> SELECT * FROM Persons
       WHERE first_name='Олександр';
```

Або, наприклад, вибрати назву й опис (title, description) артефакту під номером 10:

```
mysql> SELECT title,description FROM Artifacts WHERE id=10;
```

13.2.2. Оператор INSERT

Оператор INSERT вставляє нові рядки в існуючу таблицю. Він має кілька форм. Параметр `ім'я_таблиці` у всіх цих формах задає таблицю, у яку повинні бути внесені рядки. Стовпці, для яких задаються значення, вказуються у списку імен стовпців (ім'я_стовпця) або в частині SET.

Синтаксис оператора INSERT:

```
INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
[INTO] ім'я_таблиці [(ім'я_стовпця,...)]
VALUES (вираження,...),(...),...
```

Ця форма команди INSERT вставляє рядки відповідно до точно вказаного в команді значення. У дужках після імені таблиці перераховуються стовпці, а після ключового слова VALUES – їхні значення.

Наприклад:

```
mysql> INSERT INTO Persons (last_name, bday)
VALUES ('Петренко', '1974');
```

У результаті буде вставлено в таблицю Persons рядок, у якому значення прізвища (last_name) і дати народження (bday) будуть задані відповідно як «Петренко» та «1974».

```
INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
[INTO] ім'я_таблиці [(ім'я_стовпця,...)]
SELECT ...
```

Ця форма команди INSERT вставляє рядки, які вибрані з іншої таблиці (таблиць).

Наприклад:

```
mysql> INSERT INTO Artifacts (author)
SELECT id FROM Persons
WHERE last_name='Петренко'
AND bday='1974';
```

У результаті буде вставлено в таблицю Artifacts у поле «автор» (author) значення ідентифікатора, вибраного з таблиці Persons за умовою, що прізвище людини Петренко.

```
INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
[INTO] ім'я_таблиці
SET ім'я_стовпця=вираження,
ім'я_стовпця=вираження, ...
```

Наприклад:

```
mysql> INSERT INTO Persons SET last_name='Петренко',
first_name='Семен';
```

Ця команда вставить у таблицю Persons у поле last_name значення «Петренко», а у поле first_name – рядок «Семен».

13.2.2. Оператор UPDATE

Оператор UPDATE обновляє значення існуючих стовпців таблиці відповідно до введених значень.

Синтаксис оператора UPDATE:

```
UPDATE [LOW_PRIORITY] [IGNORE] ім'я_таблиці
SET ім'я_стовпця1=вираз1
[, ім'я_стовпця2=вираз2, ...]
[WHERE where_definition]
[LIMIT число]
```

В елементі SET указується, які саме стовпці слід модифікувати і які величини повинні бути в них встановлені.

В елементі WHERE, якщо воно присутнє, задається, які рядки підлягають відновленню. В інших випадках обновляються всі рядки. Якщо задано елемент ORDER BY, то рядки будуть обновлятися відповідно до зазначеного у цьому елементі порядку.

Якщо вказується ключове слово LOW_PRIORITY, то виконання команди UPDATE затримується доти, поки інші клієнти не завершать читання цієї таблиці.

Якщо вказується ключове слово IGNORE, то команда відновлення не буде перервана, навіть якщо виникне помилка дублювання ключів. Рядки, через які виникають конфліктні ситуації, обновлені не будуть.

Якщо в елементі, що задає нове значення стовпця, використовується ім'я цього поля, то команда UPDATE використає для цього стовпця його поточне значення.

Наприклад, наступна команда встановлює стовпець death_date у значення, на одиницю більше за його поточну величину:

```
mysql> UPDATE Persons SET death_date=death_date+1;
```

У версії MySQL 3.23 можна використати параметр LIMIT #, щоб переконатися, що було змінено тільки задану кількість рядків.

Наприклад, така операція замінить у першому рядку таблиці експонатів назву title на рядок «Персональна ЕОМ»:

```
mysql> UPDATE Artifacts SET title='Персональна ЕОМ' Limit 1;
```

13.2.4. Оператор DELETE

Оператор DELETE видаляє з таблиці `ім'я_таблиці` рядка, що задовольняє задану в where_definition умову, і повертає число вилучених записів. Якщо оператор DELETE запускається без визначення WHERE, то видаляються всі рядки.

Синтаксис оператора DELETE:

```
DELETE [LOW_PRIORITY] FROM ім'я_таблиці
[WHERE where_definition]
[LIMIT rows]
```

Наприклад, така команда видалить із таблиці Persons всі записи, у яких поле «рік народження» (bday) більше 2003:

```
mysql> DELETE FROM Persons WHERE bday>2003;
```

Видалити всі записи в таблиці можна ще і за допомогою такої команди:

```
mysql> DELETE FROM Persons WHERE 1>0;
```

Але цей метод працює набагато повільніше, ніж використання тієї ж команди без умови:

```
mysql> DELETE FROM Persons;
```

Специфічна для MySQL опція LIMIT для команди DELETE указує серверу максимальну кількість рядків, які треба видалити до повернення керування клієнтові. Вона може використовуватися для гарантії того, що ця команда не витратить занадто багато часу для виконання.

Висновки

У даному розділі розглянуто такі основні питання:

- особливості мови SQL;
- оператори відкриття, видалення таблиць і зміни їх структури мовою SQL;
- оператори роботи з елементами таблиць мовою SQL.

Контрольні питання

1. Для чого необхідна мова SQL?
2. Які функції мають СКБД?
3. До яких функцій мають доступ адміністратори СКБД?
4. Як створити таблицю БД мовою SQL?
5. Як видалити таблицю БД мовою SQL?
6. Як змінити структуру таблиці БД мовою SQL?
7. Які оператори застосовуються у мові SQL для роботи з елементами таблиць?
8. Як зчитувати рядки з таблиці БД мовою SQL?
9. Як додати або видалити рядки таблиці БД мовою SQL?
10. Як обновити рядки таблиці БД мовою SQL?

14. ПРИКЛАД ВЗАЄМОДІЇ ПРОГРАМИ МОВОЮ PHP ІЗ СКБД MYSQL

Навчальною метою розділу є ознайомлення студентів з розробкою приклада взаїмодії програми мовою PHP із СКБД MySQL.

У результаті вивчення даного розділу студенти повинні знати:

- етапи розробки програми мовою PHP для взаємодії із СКБД MySQL;

вміти:

- установити з’єднання з СКБД MySQL мовою PHP;
- вибрати БД;
- одержати список полів БД;
- відобразити поля в html-формі й записати їх у БД;
- відобразити дані, що зберігаються в MySQL.

Мова PHP містить вбудовані функції для роботи із СКБД MySQL.

Познайомимося з деякими основними функціями мови PHP для роботи з MySQL, які будуть потрібні для вирішення завдань побудови веб-інтерфейсів з метою відображення і наповнення бази даних.

Виникає питання – для чого необхідно будувати такі інтерфейси?

Вносити інформацію в базу даних і переглядати її зміст можуть тільки підготовлені користувачі, які знайомі з мовою запитів SQL.

Для додавання інформації в базу даних через веб-інтерфейс достатньо ввести ці дані в html-форму і відправити їх на сервер, а програма вже зробить усе інше.

Щоб переглянути таблиці бази даних, досить у браузері перейти за посиланням на потрібну сторінку і прочитати необхідну інформацію.

Наприклад, спробуємо побудувати ці інтерфейси для таблиці Artifacts, у якій утримується інформація про експонати віртуального музею інформатики.

У попередніх розділах уже наводилася структура цієї колекції, а також її зв'язки з таблицями опису персон (Persons) і зображень (Images). Нагадаємо, що кожен експонат у таблиці Artifacts описується за допомогою таких характеристик:

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

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

Для побудови веб-інтерфейсу додавання інформації в таблицю Artifacts бази даних відобразимо її структуру (тобто набір полів) в html-форму.

Розіб'ємо це завдання на такі підзадачі:

- установлення з'єднання з БД;

– вибір робочої БД;
– одержання списку полів таблиці;
– відображення полів в html-форму;
– запис даних з форми у БД;
– відображення даних, що зберігаються, в MySQL.

Розглянемо всі ці підзадачі.

14.1. Початок взаємодії

14.1.1. Установлення з'єднання

Для того щоб встановити з'єднання з базою даних, скористаємося функцією mysql_connect.

Синтаксис функції mysql_connect:

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

Ця функція з'єднує із сервером MySQL і повертає покажчик на це з'єднання або на FALSE у випадку невдачі.

Для відсутніх параметрів установлюються такі значення за умовчанням:

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

Якщо функція викликається двічі з тими самими параметрами, то посилання повертається на старе з'єднання, оскільки нове не встановлюється. Щоб цього уникнути, використовують параметр new_link, що змушує в кожному разі відкривати ще одне з'єднання.

Параметр client_flags – це комбінація таких констант:
– MYSQL_CLIENT_COMPRESS (використати протокол стиску);
– MYSQL_CLIENT_IGNORE_SPACE (дозволяє ставити пробіли після імен функцій);
– MYSQL_CLIENT_INTERACTIVE (чекати interactive_timeout секунд, замість wait_timeout, до закриття з'єднання).

З'єднання із сервером закривається після завершення виконання програми, якщо вона до цього не була закрита за допомогою функції mysql_close().

Отже, установлюємо з'єднання з базою даних на локальному сервері для користувача aks2012 з паролем "qwerty123":

```
<?php

$conn = mysql_connect("localhost", "aks2012","qwerty123")
or die("Неможливо встановити з'єднання: ". mysql_error());
```

```
echo "З'єднання встановлене";
mysql_close($conn);

?>
```

Дія mysql_connect подібна команді shell>mysql -u aks2012 -pqwerty123

14.1.2. Вибір бази даних

Після установлення з'єднання потрібно вибрати базу даних, з якою будемо працювати. Припустімо, що дані зберігаються в базі даних book. В MySQL вибір бази даних здійснюється за допомогою команди use:

```
mysql>use book;
```

Мова PHP має для цього функцію mysql_select_db.

Синтаксис функції mysql_select_db:

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

Ця функція повертає TRUE у випадку успішного вибору бази даних і FALSE – у противному разі.

Зробимо базу даних book робочою.

Приклад 14.1

```
<?php

$conn = mysql_connect("localhost","aks2012","qwerty123")
or die("Неможливо встановити з'єднання: ". mysql_error());
echo "З'єднання встановлене";
mysql_select_db("book");

?>
```

14.2. Робота з базою даних

14.2.1. Одержання списку полів таблиці

Тепер можна зайнятися власно вирішенням завдання.

В PHP для одержання списка полів таблиці є 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 у таблиці table_name у базі даних database_name.

Результат роботи цієї функції – змінна типу ресурс. Проте це не зовсім те, що було потрібно. Це є посилання, яке можна використати для одержання інформації про поля таблиці, включаючи їхні назви, типи і прапори.

Можливості деяких функцій:

– повертає ім'я поля, отриманого в результаті виконання запиту (mysql_field_name);

– повертає довжину поля (mysql_field_len);

– повертає тип поля (mysql_field_type);

– повертає список прапорів поля, записаних через пробіл (mysql_field_flags).

Типи поля можуть бути такі: int, real, string, blob і т.д.; прапори – not_null, primary_key, unique_key, blob, auto_increment і т.д.

Синтаксис у всіх 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),
```

де result – це ідентифікатор результату запиту (наприклад, запиту, відправленого функцією mysql_list_fields або mysql_query), а field_offset – порядковий номер поля у цьому результаті.

Загалом кажучи, те, що повертають функції типу mysql_list_fields або mysql_query, являє собою таблицю, а, точніше, покажчик на неї. Щоб одержати із цієї таблиці конкретні значення, потрібно задіяти спеціальні функції, які порядково читають цю таблицю.

До таких функцій і належить mysql_field_name і т.п.

Щоб прочитати всі рядки в таблиці результату запиту, потрібно знати число рядків у цій таблиці. Команда mysql_num_rows(ресурс result) повертає число рядків у множині результатів result.

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

Приклад 14.2

```
<?php

$conn = mysql_connect("localhost","aks2012","qwerty123")
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>";
}
?>
```

У результаті повинне вийти приблизно от що (якщо в таблиці всього два поля):

```
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:
```

14.2.2. Відображення списку полів у html-формі

Тепер підкорегуємо попередній приклад.

Будемо не просто виводити інформацію про поле, а відображати його в підходящий елемент html-форми. Так, елементи типу BLOB переведемо в textarea (помітимо, що поле description, що створюється з типом TEXT, відображається як тип BLOB), числа і рядки відобразимо в текстових рядках введення <input type=text>, а елемент, що має мітку автоінкремента, взагалі не будемо відображати, оскільки його значення встановлюється автоматично.

Все це вирішується досить просто, за винятком виділення зі списку прапора auto_increment.

Для цього потрібно скористатися функцією explode.

Синтаксис функції 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 як роздільник потрібно взяти пробіл " ", а як вихідний рядок для розбиття – рядок прапорів поля.

Отже, створимо форму для введення даних у таблицю Artifacts.

Приклад 14.3

```
<?php
$conn=mysql_connect("localhost","aks2012","qwerty123");
          // установлюємо з'єднання
$database = "book";
$table_name = "Artifacts";
mysql_select_db($database); // вибираємо базу даних для
          // роботи
$list_f = mysql_list_fields($database,$table_name);
          // одержуємо список полів у базі
$n = mysql_num_fields($list_f); // число рядків у результаті
          // попереднього запиту (тобто, скільки всього
          // полів у таблиці Artifacts)
echo "<form method=post action=insert.php>";
          // створюємо форму для введення даних
echo " <TABLE BORDER=0 CELLSPACING=0 width=50% ><tr>
    <TD  BGCOLOR='#005533' align=center><font color='#FFFFFF'>
    <b> Add new row in
$table_name</b></font></td></tr><tr><td></td></tr></TABLE>";
echo "<table border=0 CELLSPACING=1 cellpadding=0 width=50% >";
// для кожного поля одержуємо його ім'я, тип, довжину і прапори
for($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 прапорів робимо масив,
    // де кожен елемент масиву - прапор поля
    $flags = explode(" ", $flags_str);
    foreach ($flags as $f){
        if ($f == 'auto_increment') $key = $name_f;
            // запам'ятовуємо ім'я автоінкремента
    }
/* для кожного поля, що не є автоінкрементом, залежно
від його типу виводимо підходящий елемент форми */
if ($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>";
?>
```

14.2.3. Запис даних з форми у БД

Після створення форми потрібно зробити найголовніше – надіслати дані з цієї форми в базу даних. Як відомо, 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 (або будь-якою іншою командою SQL) в PHP-програмі?

Для цього існує функція mysql_query().

Синтаксис функції mysql_query:

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

Функція mysql_query() посилає SQL-запит активній базі даних MySQL сервера, що визначається за допомогою покажчика link_identifier (це посилання на якесь з'єднання із сервером MySQL).

Якщо параметр link_identifier відсутній, використовується останнє відкрите з'єднання.

Якщо відкрите з'єднання відсутнє, функція намагається з'єднатися із СКБД аналогічно функції mysql_connect() без параметрів.

Результат запиту записується у проміжну пам'ять.

Зауваження: рядок запиту не повинен закінчуватися крапкою з комою.

Тільки для запитів SELECT, SHOW, EXPLAIN, DESCRIBE функція mysql_query() повертає покажчик на результат запиту або FALSE, якщо запит не був виконаний.

Для інших запитів функція mysql_query() повертає TRUE, якщо запит виконаний успішно, і FALSE – у випадку помилки.

Значення, яке не дорівнює FALSE, говорить про те, що запит був виконаний успішно. Воно не повідомляє про кількість порушених або повернутих рядків.

Цілком можлива ситуація, коли успішний запит не торкнеться жодного рядку.

Функція mysql_query() також вважає запит помилковим і поверне FALSE, якщо у користувача недостатньо прав для роботи із зазначеною в запиті таблицею.

Отже, тепер відомо, як відправити запит на додавання рядків у базі даних. Відзначимо, що в попередньому прикладі елементи форми були названі іменами полів таблиці. Тому вони будуть доступні у програмі insert.php, що обробляє дані форми, як змінні виду $_POST['ім'я_поля'].

Приклад 14.3

```
<?
$conn=mysql_connect("localhost","aks2012","qwerty123");
// установлюємо з'єднання
$database = "book";
$table_name = "Artifacts";
mysql_select_db($database); // вибираємо базу даних
$list_f = mysql_list_fields($database,$table_name);
              // одержуємо список полів у базі
$n = mysql_num_fields($list_f); // число рядків у результаті
              // попереднього запиту
// складемо один запит відразу для всіх полів таблиці
$sql = "INSERT INTO $table_name SET "; // починаємо створювати
    // запит, перебираємо всі поля таблиці
for($i=0;$i<$n; $i++){
    $name_f = mysql_field_name ($list_f,$i); // обчислюємо імена
поля
    $value = $_POST[$name_f]; // обчислюємо значення поля
    $j = $i + 1;
    $sql = $sql . $name_f." = '$value'"; // дописуємо в
              // рядок $sql пари ім'я=значення
    if ($j <> $n) $sql = $sql . ", "; // якщо поле не
              // останнє в списку, то ставимо кому
}
// перед тим як записувати щось у базу,
// можна подивитися, який запит вийшов
//echo $sql;
$result = mysql_query($sql,$conn); // відправляємо запит
// виводимо повідомлення чи успішно виконаний запит
if (!$result) echo " Can't add ($table_name) ";
    else echo "Success!<br>";
?>
```

Отже, завдання додавання даних за допомогою веб-інтерфейсу вирішено. Однак тут є одне зауваження.

Під час вирішення не було враховано той факт, що значення деяких полів (author, photo) повинні братися з інших таблиць (Persons, Images). Оскільки MySQL із зовнішніми ключами не працює, це необхідно враховувати розробникам програми.

Потрібно дописати програму таким чином, щоб була можливість уводити в такі поля правильні значення.

Тепер перейдемо до іншого завдання – відображення даних, що зберігаються в базі даних СКБД MySQL.

14.2.4. Відображення даних, що зберігаються в MySQL

Щоб відобразити якісь дані в браузері за допомогою мови PHP, потрібно спочатку одержати ці дані у вигляді змінних PHP.

При роботі з MySQL без посередника (такого як PHP) вибірка даних виконується за допомогою команди SELECT мови SQL:

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

У попередньому розділі мова йшла про те, що будь-який запит, у тому числі й на зчитування, можна відправити на сервер за допомогою функції mysql_query(). Тоді виконувалося трохи інше завдання – одержати дані з форми і відправити їх за допомогою запиту на додавання в базу даних.

Результатом роботи функції mysql_query() там могло бути тільки TRUE або FALSE.

Тепер же потрібно відправити запит на зчитування всіх полів, а результат відобразити в браузері. І тут результат – це ціла таблиця значень, а точніше, покажчик на цю таблицю.

Так що потрібні якісь аналоги функції mysql_field_name(), тільки щоб вони витягали з результату запитів не ім'я, а значення поля.

Таких функцій в мові PHP достатньо. Найбільш популярні – mysql_result() і mysql_fetch_array().

Синтаксис функції mysql_result:

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

Функція mysql_result() повертає значення одного поля результату запиту. Аргумент field може бути порядковим номером поля в результаті, ім'ям поля або ім'ям поля з ім'ям таблиці через крапку tablename.fieldname.

Якщо для імені поля в запиті застосовувався синонім ('select foo as bar from...'), його можна використовувати замість реального імені поля.

Працюючи з великими за розміром результатами запитів, треба задіяти одну з функцій, що обробляють відразу цілий ряд результатів (наприклад, mysql_fetch_row(), mysql_fetch_array() і т.д.). Такі функції повертають значення декількох полів відразу, тому вони набагато швидше mysql_result(). Крім того, потрібно врахувати, що вказівка чисельного зсуву (номера поля) працює набагато швидше, ніж вказівка колонки або колонки і таблиці через крапку.

Виклики функції mysql_result() не повинні змішуватися з іншими функціями, що працюють із результатом запиту.

Синтаксис функції mysql_fetch_array:

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

Ця функція обробляє рядок результату запиту, повертаючи масив (асоціативний, чисельний або обоє) з уже обробленим рядком або FALSE, якщо рядків більше немає.

Функція mysql_fetch_array() – це розширена версія функції mysql_fetch_row(). Крім зберігання значень у масиві з числовими індексами, функція повертає значення в масиві з індексами за назвою колонок.

Якщо частина колонок у результаті буде мати однакові назви, тоді буде повернута остання колонка. Щоб одержати доступ до перших, слід використати числові індекси масиву або синоніми в запиті. Проте для синонімів не можна використати дійсні імена колонок, як, наприклад, неможливо використати "photo" в описаному нижче прикладі.

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

Важливо відзначити, що функція mysql_fetch_array() працює не повільніше, ніж функція mysql_fetch_row(), і забезпечує більш зручний доступ до даних.

Опціональний аргумент result_type у функції mysql_fetch_array() є константою і може приймати такі значення: MYSQL_ASSOC, MYSQL_NUM і MYSQL_BOTH. Ця можливість додана в мову PHP 3.0.7. Значенням за умовчанням є MYSQL_BOTH.

Якщо використовувати MYSQL_BOTH, одержимо масив, що складається як з асоціативних індексів, так і з числових.

MYSQL_ASSOC поверне тільки асоціативні відповідності, а MYSQL_NUM – тільки числові.

Імена полів, що повертаються цією функцією, – регістрозалежні.

Тепер відобразимо дані з таблиці Artifacts у браузері.

Приклад 14.5

```
<?
/ * спочатку робимо те саме, що і раніше: установлюємо
з'єднання, вибираємо базу й одержуємо список і число полів у
таблиці Artifacts */
$conn=mysql_connect("localhost","aks2012","qwerty123");
$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 у масиві $names
for($j=0;$j<$n1; $j++){
    $names[] = mysql_field_name ($list_f,$j);
}
$sql = "SELECT * FROM $table_name"; // створюємо SQL-запит
$q = mysql_query($sql,$conn) or die(); // відправляємо
```

```
          // запит на сервер
$n = mysql_num_rows($q); // одержуємо число рядків результату
//створюємо 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; $i++){ // перебираємо всі рядки в
                // результаті запиту на вибірку
        echo "<tr>";
        foreach  ($names as $k => $val) { // перебираємо всі
                // імена полів
        $value = mysql_result($q,$i,$val); // одержуємо
                // значення поля
        echo "<td><font size=2> $value</font></td>";
                // виводимо значення поля
        }
    echo "</tr>";
    }
echo "</table>";
```

Зробимо те саме за допомогою функції mysql_fetch_array().

Приклад 14.6

```
<?
/* ... початок  той самий, що і у попередньому прикладі */
// відображаємо значення полів
for($i=0;$i<$n; $i++){
// одержуємо значення поля у вигляді асоціативного масиву
     while($row = mysql_fetch_array($q, MYSQL_ASSOC)) {
        echo "<tr>";
        foreach  ($names as $k => $val){
            echo "<td><font size=2> $row[$val]</font></td>";
                // виводимо значення поля
        }
    echo "</tr>";
    }
}
echo "</table>";
?>
```

Висновки

У даному розділі розглянуто такі основні питання:

- особливості побудови веб-інтерфейсу мовою PHP для додавання інформації у БД;
- робота з БД MySQL мовою PHP;
- відображення даних, що зберігаються в MySQL.

Контрольні питання

1. Які можна виділити підзадачі при розробці веб-інтерфейсу мовою PHP у роботі з БД MySQL?
2. Як установити з'єднання з БД MySQL у програмі мовою PHP?
3. Як вибрати БД у програмі мовою PHP?
4. Як одержати список полів таблиці у програмі мовою PHP?
5. Як відобразити список полів в html-формі у програмі мовою PHP?
6. Як записати дані з форми БД у програмі мовою PHP?
7. Як відобразити дані, що зберігаються в БД MySQL у програмі мовою PHP?
8. Для яких задач можна використати наведену у розділі технологію розробки програм?

ЧАСТИНА 5. ЛАБОРАТОРНИЙ ПРАКТИКУМ

Лабораторний практикум містить у розділах 15–18 цикл взаємозв'язаних лабораторних робіт, у яких розглядаються і вивчаються питання ознайомлення із синтаксисом і структурою програм мовою PHP, розробки програм для обчислювальних розгалужуваних, а також для повторюваних процесів, використання одновимірних, багатовимірних і динамічних масивів і функцій.

Перед виконанням лабораторної роботи студенти повинні ознайомитися з методичними вказівками, а по закінченні – скласти звіт, який включає теоретичні відомості, рисунки, таблиці та графіки.

Студентам також необхідно відповісти (усно) на контрольні питання, які наведені в методичних вказівках у кінці лабораторної роботи.

Під час проведення лабораторної роботи студенти повинні:

- обов'язково дотримуватися правил охорони праці;
- виконувати лабораторну роботу за відповідною методикою.

Після завершення лабораторної роботи вони показують виконане завдання викладачу, потім оформляють звіт і захищають його.

15. ОСНОВИ ПРОГРАМУВАННЯ МОВОЮ PHP

15.1. Структура програм мовою PHP

Мета лабораторної роботи

Вивчити структуру програм мовою PHP, ознайомитися із особливостями розміщення їх на сервері.

Організація виконання лабораторної роботи

Для виконання лабораторної роботи необхідно вивчити, використовуючи рекомендовану літературу, конспект лекцій і методичні вказівки, такі питання:

- структура веб-сторінок;
- структура програм мовою PHP;
- розміщення програм мовою PHP на сервері.

Далі виконати такі дії:

- розробити програму мовою PHP, яка виводить на екран напівжирним шрифтом ім'я, прізвище і назву навчального закладу, в якому навчається автор програми;
- розробити сторінку мовою html і встроїти в неї розроблену мовою PHP програму;
- розмістити отриману сторінку на сервері й перевірити її роботу;
- записати результат роботи програми з екрана дисплея у файли з форматом jpg.

Підготувати звіт з виконання лабораторної роботи, який повинен включати:

- тему і мету лабораторної роботи;

– опис завдання з початковими умовами і даними;
– текст програми мовою PHP згідно з варіантом;
– результат виконання програми.

Питання для підготовки до захисту лабораторної роботи

1. Дати визначення веб-сервера.
2. Для чого потрібно встановлювати PHP-інтерпретатор?
3. Які операційні системи можна використовувати для виконання PHP-програм?
4. Які файли необхідно налаштувати для того, щоб забезпечити роботу веб-сервера та PHP-інтерпретатора?
5. Яким програмним забезпеченням можна користуватися для написання програм мовою PHP?
6. Як виконати програму, написану мовою PHP?
7. Чому в браузер не передається код PHP-програми?
8. Яку структуру має PHP-програма?

15.2. Змінні, константи та оператори

Мета лабораторної роботи

Ознайомитись із змінними, константами та операторами мови PHP.

Організація виконання лабораторної роботи

Для виконання лабораторної роботи необхідно вивчити, використовуючи рекомендовану літературу, конспект лекцій і методичні вказівки, такі питання:

– структура програм мовою PHP;
– змінні, константи і оператори мови PHP: визначення, види, використання;
– типи даних мови PHP.

Далі виконати такі дії:

– розробити PHP-програму, яка відповідно до завдання (табл. 15.1) виконує арифметичні дії із змінними x та y, виводить на екран ці змінні, а також формулу, згідно з якою виконувалися арифметичні дії, і результат обчислення z;
– врахувати, що розроблена PHP-програма повинна мати всі види коментарів;
– розмістити отриману програму на сервері й перевірити її роботу;
– записати результат роботи програми з екрана дисплея у файли з форматом jpg.

Підготувати звіт з виконання лабораторної роботи, який повинен включати:

– тему і мету лабораторної роботи;
– опис завдання з початковими умовами і даними;
– текст програми мовою PHP згідно з варіантом;
– результат виконання програми.

Таблиця 15.1

Варіанти завдань

№ вар.	Завдання	№ вар.	Завдання
1.	x=216, y=11, $Z=\frac{x+21y}{2x+86}-1066$	15.	x=116, y=25, $Z=2012+\frac{x+15y}{4+86x}$
2.	x=226, y=12, $Z=\frac{x+21y}{4x-86}+1067$	16.	x=126, y=26, $Z=\frac{x+15y}{4+86x}-955$
3.	x=236, y=13, $Z=\frac{x+21y}{6x+86}-1069$	17.	x=136, y=27, $Z=2014+\frac{x+15y}{4+86x}$
4.	x=246, y=14, $Z=\frac{x+21y}{8x-86}+1070$	18.	x=146, y=28, $Z=\frac{x+15y}{4+86x}-956$
5.	x=256, y=15, $Z=\frac{x+21y}{10x+86}+1071$	19.	x=156, y=29, $Z=2016+\frac{x+15y}{4+86x}$
6.	x=276, y=16, $Z=\frac{x+21y}{12x-86}+1072$	20.	x=166, y=30, $Z=\frac{x+15y}{4+86x}-957$
7.	x=286, y=17, $Z=\frac{x+21y}{14x+86}-1073$	21.	x=176, y=31, $Z=2018+\frac{x+15y}{4+86x}$
8.	x=296, y=18, $Z=\frac{x+21y}{14x-86x}+1074$	22.	x=186, y=32, $Z=\frac{x+15y}{4+86x}-958$
9.	x=306, y=19, $Z=\frac{x+21y}{12+86}-1075$	23.	x=196, y=33, $Z=2020+\frac{x+15y}{4+86x}$
10.	x=316, y=20, $Z=\frac{x+21y}{10x-86}+1076$	24.	x=366, y=34, $Z=\frac{x+15y}{4+86x}-959$
11.	x=326, y=21, $Z=\frac{x+21y}{8x+86}-1077$	25.	x=376, y=35, $Z=2022+\frac{x+15y}{4+86x}$
12.	x=336, y=22, $Z=\frac{x+21y}{6x-86}+1078$	26.	x=386, y=36, $Z=\frac{x+15y}{4+86x}-960$
13.	x=346, y=23, $Z=\frac{x+21y}{4x+86}-1079$	27.	x=396, y=37, $Z=2024+\frac{x+15y}{4+86x}$
14.	x=356, y=24, $Z=\frac{x+21y}{2x-86}+1080$	28.	x=406, y=38, $Z=\frac{x+15y}{4+86x}-961$

Питання для підготовки до захисту лабораторної роботи

1. Яким чином визначається код PHP-програм?
2. Для чого потрібні коментарі в програмах?
3. Які відомі типи коментарів?
4. Що таке змінні мови PHP?
5. Які відомі правила іменування змінних?
6. Яку роль у змінних відіграє знак амперсанд &?
7. Яка різниця між змінними і константами?
8. Які відомі оператори мови PHP?
9. Чим відрізняються оператори «==» та «===»мови PHP?
10. Опишіть роботу зі змінними типу object мови PHP?

15.3. Типи даних

Мета лабораторної роботи

Ознайомитися з типами даних мови PHP.

Організація виконання лабораторної роботи

Для виконання лабораторної роботи необхідно вивчити, використовуючи рекомендовану літературу, конспект лекцій і методичні вказівки, такі питання:

- структура програм мовою PHP;
- строкові масиви мови PHP.

Далі виконати такі дії:

- розробити PHP-програму, яка створює масив із семи елементів зі строковим значенням і заповнює його сімома фразами;
- при кожному оновленні сторінки повинна виводитися одна із фраз цього масиву;
- розмістити отриману програму на сервері й перевірити її роботу;
- записати результат роботи програми з екрана дісплея у файли з форматом jpg.

Підготувати звіт з виконання лабораторної роботи, який повинен включати:

- тему і мету лабораторної роботи;
- опис завдання з початковими умовами і даними;
- текст програми мовою PHP згідно з варіантом;
- результат виконання програми.

Питання для підготовки до захисту лабораторної роботи

1. Які відомі прості типи даних мови PHP?
2. Які відомі скалярні типи даних мови PHP?
3. Які відомі змішані типи даних мови PHP?
4. Які відомі спеціальні типи даних мови PHP?
5. Чи має мова PHP вбудовану підтримку Unicode?
6. Чи має мова PHP обмеження на довжину рядка?
7. Якими способами у мові PHP визначається рядок?
8. Якими способами у мові PHP визначається масив?
9. Як отримати значення елемента масива мовою PHP?
10. Який тип даних мови PHP прийшов з об’єктно-орієнтованого програмування?

15.4. Керуючі конструкції

Мета лабораторної роботи

Ознайомитися з керуючими конструкціями мови PHP.

Організація виконання лабораторної роботи

Для виконання лабораторної роботи необхідно вивчити, використовуючи рекомендовану літературу, конспект лекцій і методичні вказівки, такі питання:

- структура програм мовою PHP;
- керуючі конструкції мови PHP.

Далі виконати такі дії:

- розробити PHP-програму, яка реалізує розпізнання користувачів веб-сторінки. При цьому в програмі необхідно:
  1) створити змінну `$user_name`, якій надалі буде присвоюватися ім'я користувача;
  2) для перевірки роботи програми ввести 5 різних імен зареєстрованих користувачів;
  3) щоб виводилися підготовлені заздалегідь вітання при введенні імені кожного із зареєстрованих користувачів;
  4) у випадку введення імені незареєстрованного користувача вивести повідомлення «Користувач не пізнаний»;
- розмістити отриману програму на сервері й перевірити її роботу;
- записати результат роботи програми з екрана дісплея у файли з форматом jpg.

Підготувати звіт з виконання лабораторної роботи, який повинен включати:

- тему і мету лабораторної роботи;
- опис завдання з початковими умовами і даними;
- текст програми мовою PHP згідно з варіантом;
- результат виконання програми.

Питання для підготовки до захисту лабораторної роботи

1. Які є керуючі конструкції мови PHP?
2. Який синтаксис оператора if?
3. Яким чином можна використовувати оператор else без оператора if?
4. Що таке альтернативний синтаксис керуючих конструкцій?
5. У яких випадках використовують оператор switch?
6. Що таке цикли і яким чином їх використовують у програмах?
7. Вкажіть типи циклів, які ви знаєте?
8. З якими даними зручно використовувати цикл foreach?
9. Яким чином можна зупинити роботу циклу?
10. За допомогою яких операторів можна включити код з іншого файлу в програму мовою PHP?

15.5. Застосування класів та обєктів

Мета лабораторної роботи

Ознайомитися з класами та об’єктами мови PHP.

Організація виконання лабораторної роботи

Для виконання лабораторної роботи необхідно вивчити, використовуючи рекомендовану літературу, конспект лекцій і методичні вказівки, такі питання:

- структура програм мовою PHP;
- класи та об’єкти мови PHP.

Далі виконати такі дії:

– розробити PHP-програму, яка реалізує гру «кубики» за таким алгоритмом:

1) при кожному оновленні сторінки генерується рядок з назвою гри, зображення верхніх боків двох кубиків, рядок з повідомленням про набрані бали і результат гри (рис. 15.1). Бік кожного з кубиків з різною кількістю кружечків на ньому (рис. 15.2) вибирається за законом випадкових чисел;

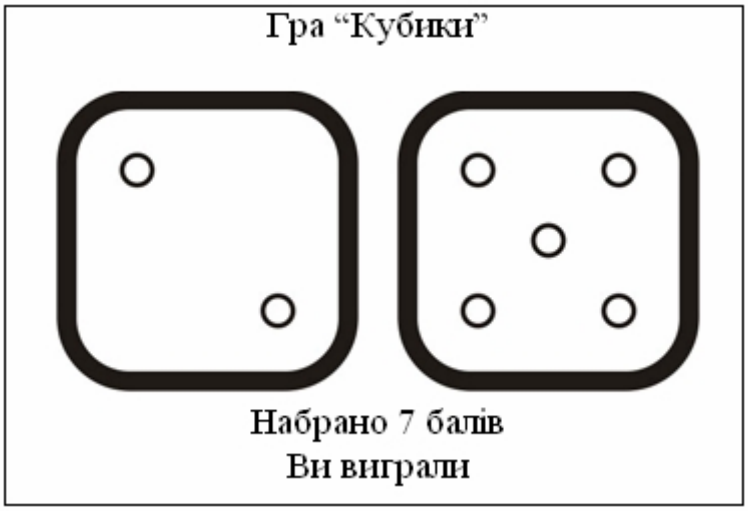


Рис. 15.1. Виведене на екран зображення результату гри «Кубики»

2) сума набраних балів розраховується за кількістю кружечків на виведених боках кубиків;

3) у випадку, якщо сума набраних балів більше за 5, то виводиться рядок “Ви виграли”, а якщо сума менше або дорівнює 5 – “Ви програли”;

- розмістити отриману програму на сервері й перевірити її роботу;
- записати результат роботи програми з екрана дисплея у файли з форматом jpg.

Підготувати звіт з виконання лабораторної роботи, який повинен включати:

- тему і мету лабораторної роботи;
- опис завдання з початковими умовами і даними;

– текст програми мовою PHP згідно з варіантом;
– результат виконання програми.

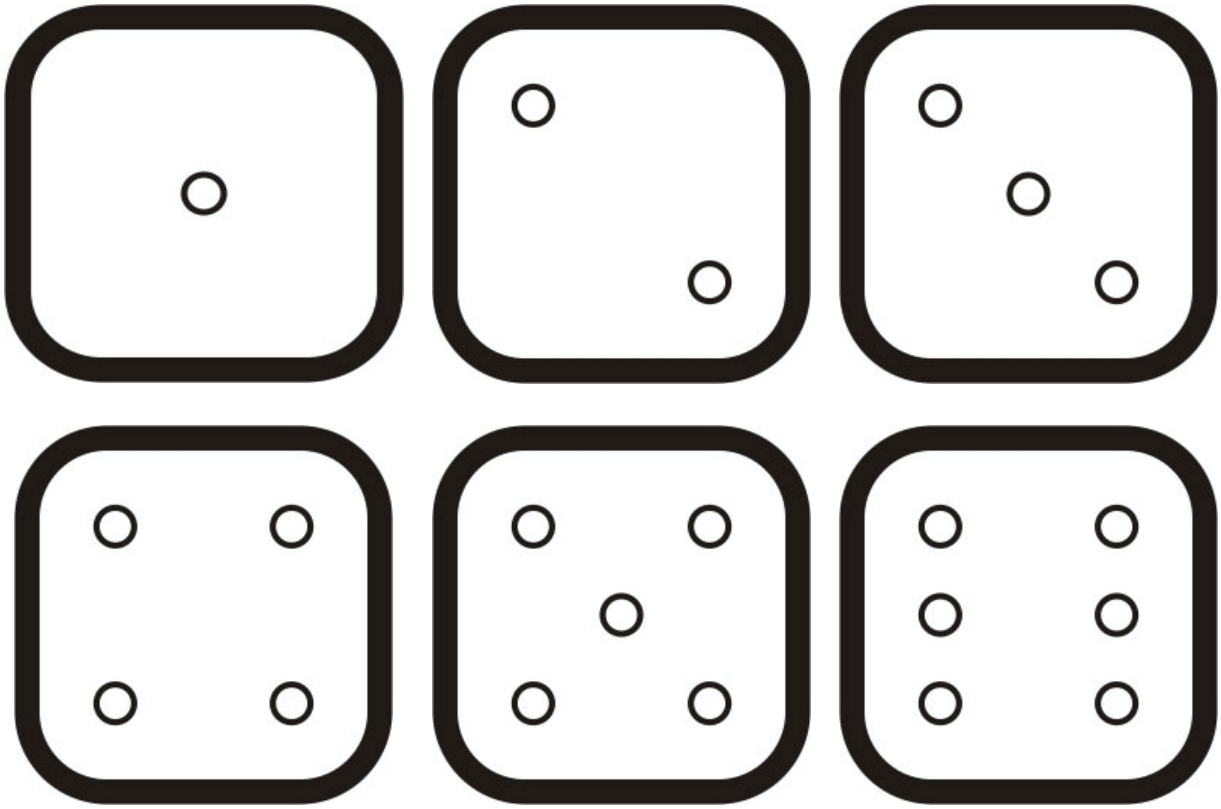


Рис. 15.2. Варіанти зображення кубиків

Питання для підготовки до захисту лабораторної роботи

1. Що таке клас? Який синтаксис визначення класу?
2. З яких частин складається клас?
3. Яким чином можна виконати ініціалізацію змінних класу?
4. Що являє собою об’єкт?
5. Чи потрібно ставити знак $ перед назвою властивості або методу класу?
6. Яким чином виконується механізм успадкування?
7. Що таке конструктор класу?
8. Яким чином виконується посилання на функції або змінні з базового класу?
9. Коли використовується оператор parent?
10. Опишіть об’єктну модель мови PHP 5.0.

16. РОБОТА З МАСИВАМИ

16.1. Розробка масивів

Мета лабораторної роботи

Ознайомитися з особливостями роботи з масивами мовою PHP.

Організація виконання лабораторної роботи

Для виконання лабораторної роботи необхідно вивчити, використовуючи рекомендовану літературу, конспект лекцій і методичні вказівки, такі питання:

- структура програм мовою PHP;
- робота з масивами мовою PHP.

Далі виконати такі дії:

- розробити PHP-програму, яка:
  1) створює масив, що має кількість елементів, обчислених за формулою:
  $$K_e = (N_{вар} + 9) \cdot 10, \quad (16.1)$$
  де K_e – кількість елементів масиву, $N_{вар}$ – номер варіанта завдання.
  Значення елементів повинно бути випадковими цілими числами у діапазоні від –20 до 20;
  2) виводити на екран усі елементи отриманого масиву із зазначенням індексу кожного елемента;
  3) виводити на екран тільки ті елементи масиву із зазначенням індексу, які відповідають умові згідно з варіантом завдання (табл. 16.1);
  4) виводити на екран повідомлення про кількість елементів, які знайдені за умовою згідно з варіантом завдання (табл. 16.1). Якщо таких елементів не знайдено, виводиться відповідне повідомлення;

Таблиця 16.1

Варіанти завдань

№ вар.	Завдання	№ вар.	Завдання
1.	Парні	10.	Менші за 4 і кратні 8
2.	Більші або дорівнюють 3	11.	Дорівнюють 9, 1, 14, 18
3.	Кратні 6	12.	Менші за 2
4.	Дорівнюють 2, 5, 6 або –15	13.	Парні від'ємні
5.	Квадрат яких більше за 4	14.	Дорівнюють 19 або 15
6.	Від'ємні або кратні 6	15.	Від'ємні
7.	Кратні 3	16.	Непарні
8.	Дорівнюють 1, 2, 5 або 15	17.	Дорівнюють 18 або непарні
9.	Квадрат яких більше за 7 або число парне	18.	Діляться на 2 без остачі

Продовження табл. 16.1

№ вар.	Завдання	№ вар.	Завдання
19.	Менші або дорівнюють –3	25.	Від'ємні або дорівнюють 0
20.	Дорівнюють –19, 12, 2	26.	Дорівнюють 0, 3, 5 або 16
21.	Додатні або дорівнюють –9	27.	Парні або дорівнюють 2
22.	Від'ємні	28.	Більші або дорівнюють 7
23.	Менші або дорівнюють –3	29.	Кратні 4 або 5
24.	Дорівнюють 5 чи від'ємні	30.	Дорівнюють –19 або 19, або 4

– розмістити отриману програму на сервері й перевірити її роботу;

– записати результат роботи програми з екрана дисплея у файли з форматом jpg.

Підготувати звіт з виконання лабораторної роботи, який повинен включати:

– тему і мету лабораторної роботи;
– опис завдання з початковими умовами і даними;
– текст програми мовою PHP згідно з варіантом;
– результат виконання програми.

Питання для підготовки до захисту лабораторної роботи

1. Що таке масив?
2. Як створити новий масив?
3. Як виконується об'єднання масивів?
4. Як виконується порівняння масивів?
5. Чим відрізняються однакові та еквівалентні масиви?
6. Як установити, чи міститься в заданому масиві шукане значення?
7. Яким чином підрахувати кількість елементів масиву?
8. Наведіть приклад пошуку елементів у масиві.
9. Якою функцією потрібно скористатися, щоб одержати всі ключі масиву?
10. Як получити масив, у якому нема повторюваних елементів?

16.2. Сортування масивів

Мета лабораторної роботи

Ознайомитися з особливостями сортування масивів мовою PHP.

Організація виконання лабораторної роботи

Для виконання лабораторної роботи необхідно вивчити, використовуючи рекомендовану літературу, конспект лекцій і методичні вказівки, такі питання:

- сортування масивів мовою PHP;
- сортування масивів за значеннями ключів;
- сортування за допомогою функції, яка задана користувачем.

Далі виконати такі дії:

- розробити PHP-програму, яка:
  1) створює масив, що має кількість елементів, обчислених за формулою 16.1;
  2) сортує створений масив за зростанням (для парних варіантів) або за спаданням (для непарних варіантів).
  3) виводить на екран усі елементи отриманого масиву із вказівкою індексу кожного елемента;
- розмістити отриману програму на сервері й перевірити її роботу;
- записати результат роботи програми з екрана дисплея у файли з форматом jpg.

Підготувати звіт з виконання лабораторної роботи, який повинен включати:

- тему і мету лабораторної роботи;
- опис завдання з початковими умовами і даними;
- текст програми мовою PHP згідно з варіантом;
- результат виконання програми.

Питання для підготовки до захисту лабораторної роботи

1. За допомогою яких функцій можна виконати сортування елементів масиву?
2. Чим відрізняються функції asort, rsort, arsort?
3. Що таке сортування за допомогою функцій?
4. Як виконується сортування масивів за значеннями ключів?
5. Як виконується сортування за допомогою функції, що задана користувачем?
6. Які є додаткові функції для роботи з масивами?
7. Які аргументи має користувацька функція?
8. Як розглядається третій аргумент користувацької функції?
9. Як вивільнити частину масиву?
10. Як розбити масив на декілька масивів заданої довжини?

17. РОБОТА З ФАЙЛАМИ

Мета лабораторної роботи

Отримати уявлення про роботу з файлами мовою PHP.

Організація виконання лабораторної роботи

Для виконання лабораторної роботи необхідно вивчити, використовуючи рекомендовану літературу, конспект лекцій і методичні вказівки, такі питання:

- структура програм мовою PHP;
- робота з файлами мовою PHP: перевірка наявності, відкривання, закривання, зчитування інформації з файлів та запис у них.

Далі виконати такі дії:

– розробити PHP-програму, яка реалізує функції книги для заміток відвідувачів. При цьому програма повинна:

1) складатися з add.php, main.php і текстового файлу data.txt.
   Шаблон для розробки add.php наведено у прикладі 17.1. У файлі add.php описується процедура прийняття від відвідувача інформації і запис її в текстовий файл data.txt. Після запису інформації повідомлення про цю подію повинне відобразитися на дисплеї. Приклад оформлення повідомлення показано на рис. 17.1. У файлі main.php описується процедура зчитування даних з текстового файлу data.txt і відображення всіх отриманих повідомлень;
2) виводити повідомлення із зазначенням номера у верхній частині;
3) зберігати дату надходження повідомлення. Для цього можна використати функцію повернення значення дати та часу, що має такий вигляд:

```
$date = date("d.m.y H:i:s");
```

Повідомлення № 1	
ПІБ:	Головко Семен Петрович
Тел.:	+380-50-1234567
ICQ:	123456789
Дата:	05.09.2011 15:35
Добрий день, шановні Адміністратори! Це повідомлення є тестовим і дозволяє перевірити функціональні можливості книги для відвідувачів. Щиро вдячний за увагу. Гість	

Рис. 17.1. Приклад оформлення повідомлення

4) відображати повідомлення про помилку і не зберігати його у файлі data.txt, якщо якесь поле не було введене;

5) виводити на екран після додавання повідомлення напис «Повідомлення додане успішно»;

– розмістити отриману програму на сервері й перевірити її роботу;

– записати результат роботи програми з екрана дисплея у файли з форматом jpg.

Підготувати звіт з виконання лабораторної роботи, який повинен включати:

– тему і мету лабораторної роботи;

– опис завдання з початковими умовами і даними;

– текст програми мовою PHP згідно з варіантом;

– результат виконання програми.

Приклад 17.1

```
<html>
     <head>
      <title>Сторінка додавання запису</title>
      <meta http-equiv="Content-Type" content="text/html;
charset=windows-1251">
     </head>

     <body>

     <?
     if($HTTP_POST_VARS['seenform'] == "y")
     {
          $name = $_POST['name'];
          $tel = $_POST['tel'];
          $icq = $_POST['icq'];
          $msg = $_POST['msg'];

// ************************************************************
     // ! Початок блоку з кодом програми студента!
     // За межами цього блоку код програми не змінювати!!!
     // Після введення відвідувачем інформації у форму
     // Дані запам'ятають змінні $name, $tel, $icq, $msg
     // Ці змінні необхідно зберегти у файлі data.txt
     print "Було введене ПІБ: $name<br>";
     print "Був введений номер телефону: $tel<br>";
     print "Був введений icq: $icq<br>";
     print "Було введене повідомлення: $msg<br>";

     // ! Кінець блоку з кодом програми студента!
// ************************************************************

     }
     else
```

```
     {
     print "<div align=\"left\">\n";
     print "<form name=\"form\" action=\"add.php\"
method=\"POST\">\n";
     print      "<input      type=\"hidden\"      name=\"seenform\"
value=\"y\">\n";
     print "<p>\n";

     print "Введіть Ваше прізвище, ім'я та по батькові (ПІБ):
<br>\n";
     print      "<input      type=\"input\"name=\"name\"value=\"\"
style=\"width:400;\">\n";
     print "</p>\n";
     print "<p>\n";

     print "Введіть Ваш номер телефона:<br>\n";
     print       "<input       type=\"input\"name=\"tel\"value=\"\"
style=\"width:400;\">\n";
     print "</p>\n";
     print "<p>\n";

     print "Введіть Ваш номер icq:<br>\n";
     print   "<input   type=\"input\"   name=\"icq\"   value=\"\"
style=\"width:400;\">\n";
     print "</p>\n";
     print "<p>\n";

     print "Введіть текст повідомлення:<br>\n";
     print        "<textarea        name=\"msg\"style=\"width:400;
height:200\"></textarea>\n";
     print "</p>\n";
     print "<p>\n";

     print " <input type=\"submit\" value=\"Зберегти\">\n";
     print " <input type=\"reset\" value=\"Очистити\">\n";
     print "</p>\n";
     print "</div>\n";
     print "</form>\n";
     }
     ?>
     </body>
     </html>
```

Питання для підготовки до захисту лабораторної роботи

1. Як виконати відкриття файлів?
2. Як виконати закриття з’єднання з файлом?
3. Як виконати запис даних у файл?
4. Як виконати видалення файлу?
5. Як виконати завантаження файлу на сервер?
6. Які є варіанти функції читання даних з файлу?

18. РОБОТА З БАЗАМИ ДАНИХ

18.1. Підключення до бази даних

Мета лабораторної роботи

Отримати уявлення про особливості роботи з базами даних мовою PHP.

Організація виконання лабораторної роботи

Для виконання лабораторної роботи необхідно вивчити, використовуючи рекомендовану літературу, конспект лекцій і методичні вказівки, такі питання:

- структура програм мовою PHP;
- особливості роботи з базами даних мовою PHP.

Далі виконати такі дії:

- розробити PHP-програму, яка:
  1) підключається до створеної раніше бази даних;
  2) виконує указані викладачем SQL-запити;
- розмістити отриману програму на сервері й перевірити її роботу;
- записати результат роботи програми з екрана дисплея у файли з форматом jpg.

Підготувати звіт з виконання лабораторної роботи, який повинен включати:

- тему і мету лабораторної роботи;
- опис завдання з початковими умовами і даними;
- текст програми мовою PHP згідно з варіантом;
- результат виконання програми.

Питання для підготовки до захисту лабораторної роботи

1. Що таке база даних?
2. Які є найбільш відомі способи організації баз даних?
3. Які основні функції повина мати СКБД?
4. Що таке реляційна та об’єктна СКБД?
5. Для чого необхідні ключі в таблицях баз даних?
6. Що таке первинний ключ в таблицях баз даних?
7. Для чого необхідно індексування в таблицях баз даних?
8. Які види індексів існують в СКБД MySQL?
9. Які є режими роботи СКБД MySQL?
10. Які дії необхідно виконати, щоб підготовити до роботи СКБД MySQL?

18.2. Додавання і видалення елементів таблиць баз даних

Мета лабораторної роботи

Ознайомитися з додаванням і видаленням елементів таблиць баз даних мовою PHP.

Організація виконання лабораторної роботи

Для виконання лабораторної роботи необхідно вивчити, використовуючи рекомендовану літературу, конспект лекцій і методичні вказівки, такі питання:

- структура програм мовою PHP;
- додавання і видалення елементів таблиць баз даних мовою PHP.

Далі виконати такі дії:

- розробити PHP-програму, яка:
  1) підключається до створеної раніше бази даних;
  2) виводить на екран усі таблиці, що знаходяться у базі даних, при чому із зазначенням їх імен та заголовків стовпчиків;
  3) для кожної таблиці встановлює клавішу, за допомогою якої додається рядок з довільними даними;
  4) для кожного рядка обраної таблиці встановлює клавішу, при натисканні на яку видаляється весь рядок;

- розмістити отриману програму на сервері й перевірити її роботу;
- записати результат роботи програми з екрана дисплея у файли з форматом jpg.

Підготувати звіт з виконання лабораторної роботи, який повинен включати:

- тему і мету лабораторної роботи;
- опис завдання з початковими умовами і даними;
- текст програми мовою PHP згідно з варіантом;
- результат виконання програми.

Питання для підготовки до захисту лабораторної роботи

1. Для чого необхідна мова SQL?
2. Які функції мають СКБД?
3. До яких функцій мають доступ адміністратори СКБД?
4. Як створити таблицю БД мовою SQL?
5. Як видалити таблицю БД мовою SQL?
6. Як змінити структуру таблиці БД мовою SQL?
7. Які оператори застосовуються у мові SQL для роботи з елементами таблиць?
8. Як зчитати рядки з таблиці БД мовою SQL?
9. Як додати або видалити рядки таблиці БД мовою SQL?
10. Як обновити рядки таблиці БД мовою SQL?

18.3. Редагування елементів таблиць баз даних

Мета лабораторної роботи

Ознайомитися з редагуванням елементів таблиць баз даних мовою PHP.

Організація виконання лабораторної роботи

Для виконання лабораторної роботи необхідно вивчити, використовуючи рекомендовану літературу, конспект лекцій і методичні вказівки, такі питання:

- структура програм мовою PHP;
- редагуванням елементів таблиць баз даних мовою PHP.

Далі виконати такі дії:

- розробити PHP-програму, яка:
  1) підключається до створеної раніше бази даних;
  2) виводить на екран усі таблиці, що знаходяться у базі даних, при чому із зазначенням їх імен та заголовків стовпчиків;
  3) для рядків обраної таблиці встановлює клавішу, натискання на яку дозволяє редагувати весь рядок;

- розмістити отриману програму на сервері й перевірити її роботу;
- записати результат роботи програми з екрана дисплея у файли з форматом jpg.

Підготувати звіт з виконання лабораторної роботи, який повинен включати:

- тему і мету лабораторної роботи;
- опис завдання з початковими умовами і даними;
- текст програми мовою PHP згідно з варіантом;
- результат виконання програми.

Питання для підготовки до захисту лабораторної роботи

1. Які можна виділити підзадачі при розробці веб-інтерфейсу мовою PHP у роботі з БД MySQL?
2. Як установити з’єднання з БД MySQL мовою PHP?
3. Як вибрати БД мовою PHP?
4. Як одержати список полів таблиці мовою PHP?
5. Як відобразити список полів в html-формі мовою PHP?
6. Як записати дані з форми у БД мовою PHP?
7. Як відобразити дані, що зберігаються в БД MySQL мовою PHP?
8. Для яких задач можна використати наведену в розділі технологію розробки програм?
9. Як виконати перехід у редагування рядків таблиці БД мовою PHP?
10. Як відобразити таблиці, що зберігаються в БД MySQL мовою PHP?

19. ОСНОВИ ПРОГРАМУВАННЯ МОВОЮ HTML

19.1. Структура програм мовою HTML

Мета лабораторної роботи

Вивчити структуру програм мовою HTML, ознайомитися із особливостями розміщення їх на сервері.

Організація виконання лабораторної роботи

Для виконання лабораторної роботи необхідно вивчити, використовуючи рекомендовану літературу, конспект лекцій і методичні вказівки, такі питання:

- структура програм мовою HTML;
- застосування тегів для форматування текстів.

Далі виконати такі дії:

- розробити HTML-програму, яка:

  1) складається із заголовної частини й тіла документа. У тілі документа необхідно застосувати такі теги: <Hn> і </Hn>, <br>, <p> з додатковими параметрами, <nobr> і </nobr> та ін.;

  2) виводить на екран назву навчального закладу, в якому навчається автор програми, його ім’я та прізвище і перелік його друзів (не менше п’яти). Перелік друзів повинен містити таку інформацію: ПІБ, дату народження, e-mail, номер контактного телефону;

- перевірити роботу отриманої програми;
- записати результат роботи програми з екрана дисплея у файли з форматом jpg.

Підготувати звіт з виконання лабораторної роботи, який повинен включати:

- тему і мету лабораторної роботи;
- опис завдання з початковими умовами і даними;
- текст програми мовою HTML згідно з варіантом;
- результат виконання програми.

Питання для підготовки до захисту лабораторної роботи

1. Що являє собою HTML-текст, документ або програму?
2. Яку структуру має HTML-програма?
3. Що таке тег мови HTML?
4. Який тег застосовується в HTML-програмі першим?
5. Які теги застосовуються у заголовній частині HTML-програми?
6. Які типи тегів відомі для форматування текстів?
7. Які додаткові параметри має тег <p> ?
8. Які теги застосовуються для заборони перенесення рядка?
9. Які теги застосовуються для спеціального форматування?
10. Яким програмним забезпеченням можна користуватися для написання програм мовою HTML?

19.2. Застосування списків в HTML-програмах

Мета лабораторної роботи

Ознайомитися з видами списків в HTML-програмах.

Організація виконання лабораторної роботи

Для виконання лабораторної роботи необхідно вивчити, використовуючи рекомендовану літературу, конспект лекцій і методичні вказівки, такі питання:

- структура програм мовою HTML;
- списки в HTML-документах: види, використання.

Далі виконати такі дії:

- розробити HTML-програму, яка виводить на екран відповідно до завдання (табл. 19.1) та інформації з попередньої роботи перелік друзів (непронумерований і пронумерований списки). Перелік друзів, як і у попередній роботі, повинен містити таку інформацію: ПІБ, дату народження, e-mail, номер контактного телефону;
- поділити перелік друзів на дві частини і розробити HTML-програму, яка виводить на екран список з вкладеними списками;
- перевірити роботу отриманої програми;
- записати результат роботи програми з екрана дисплея у файли з форматом jpg.

Підготувати звіт з виконання лабораторної роботи, який повинен включати:

- тему і мету лабораторної роботи;
- опис завдання з початковими умовами і даними;
- тексти програм мовою HTML згідно з варіантом;
- результати виконання програм.

Таблиця 19.1

Варіанти завдань

№ вар.	Вид маркера для списку		
	непронумеро-ваного	пронумерованого	вкладеного
1.	За умовчанням	Великі латинські літери	Великі римські цифри
2.	Круглий	Маленькі латинські літери	Маленькі римські цифри
3.	Квадратний	Великі римські цифри	Арабські цифри
4.	За умовчанням	Маленькі римські цифри	Великі латинські літери
5.	Круглий	Арабські цифри	Маленькі латинські літери
6.	Квадратний	Великі латинські літери	Великі римські цифри
7.	За умовчанням	Маленькі латинські літери	Маленькі римські цифри
8.	Круглий	Великі римські цифри	Арабські цифри
9.	Квадратний	Маленькі римські цифри	Великі латинські літери
10.	За умовчанням	Арабські цифри	Маленькі латинські літери

Продовження табл. 19.1

№ вар.	Вид маркера для списку		
	непронумеро-ваного	Пронумерованого	Вкладеного
11.	Круглий	Великі латинські літери	Великі римські цифри
12.	Квадратний	Маленькі латинські літери	Маленькі римські цифри
13.	За умовчанням	Великі римські цифри	Арабські цифри
14.	Круглий	Маленькі римські цифри	Великі латинські літери
15.	Квадратний	Арабські цифри	Маленькі латинські літери
16.	За умовчанням	Великі латинські літери	Великі римські цифри
17.	Круглий	Маленькі латинські літери	Маленькі римські цифри
18.	Квадратний	Великі римські цифри	Арабські цифри
19.	За умовчанням	Маленькі римські цифри	Великі латинські літери
20.	Круглий	Арабські цифри	Маленькі латинські літери
21.	Квадратний	Великі латинські літери	Великі римські цифри
22.	За умовчанням	Маленькі латинські літери	Маленькі римські цифри
23.	Круглий	Великі римські цифри	Арабські цифри
24.	Квадратний	Маленькі римські цифри	Великі латинські літери
25.	За умовчанням	Арабські цифри	Маленькі латинські літери

Питання для підготовки до захисту лабораторної роботи

1. Які основні види списків застосовуються в HTML-програмах?
2. Як сформувати непронумерований список?
3. Як сформувати пронумерований список?
4. Як сформувати вкладений список?
5. Як сформувати список визначень?
6. Які параметри може мати тег <ul> ?
7. Які параметри може мати тег <ol> ?
8. Які параметри може мати тег <li> ?
9. Який список служить для створення списків типу "термін"–"опис"?
10. Що в списках позначається тегами <dt> і <dd>?

19.3. Розробка складних web-сторінок

Мета лабораторної роботи

Ознайомитися з особливостями застосування посилань для розробки складних web-сторінок мовою HTML.

Організація виконання лабораторної роботи

Для виконання лабораторної роботи необхідно вивчити, використовуючи рекомендовану літературу, конспект лекцій і методичні вказівки, такі питання:

– структура програм мовою HTML;

– гіпертекстові посилання і посилання на мережні сервіси у складних web-сторінках мовою HTML.

Далі виконати такі дії:

– розробити web-сторінку, яка складається з головної HTML-програми (index.htm) і трьох додаткових (link1.htm, link2.htm, photo.htm). Головна сторінка має посилання на програми link1.htm та link2.htm. Програма link1.htm – це пронумерований список друзів, а програма link2.htm – список друзів з вкладеними списками з попередньої роботи. З програм link1.htm та link2.htm є посилання на програму photo.htm, яка має архів фотографій друзів. При кожному відображенні web-сторінки за допомогою додаткових програм повинна бути можливість повернення на головну або вихідну сторінку;

– перевірити роботу отриманих програм;

– записати результат роботи програм з екрана дисплея у файли з форматом jpg.

Підготувати звіт з виконання лабораторної роботи, який повинен включати:

– тему і мету лабораторної роботи;
– опис завдання з початковими умовами і даними;
– тексти програми мовою HTML згідно з варіантом;
– результати виконання програм.

Питання для підготовки до захисту лабораторної роботи

1. Що використовує мова HTML для подання гіпертекстових посилань і посилань на мережні сервіси?
2. На що вказує перша частина URL?
3. Як інтерпретується друга частина URL (після двокрапки)?
4. Які відомі варіанти компоненти URL “метод”?
5. Якими тегами відзначається URL?
6. Як веб-браузер відображає текст посилання на URL?
7. Як зробити посилання на різні ділянки або розділи того самого документа?
8. Як зробити посилання на різні ділянки або розділи іншого документа?
9. Для чого роблять посилання на різні ділянки або розділи документа?
10. Що повідомляють браузеру символи "#r1" ?

19.4. Розміщення таблиць на web-сторінках

Мета лабораторної роботи

Ознайомитися з особливостями розробки таблиць для web-сторінок мовою HTML.

Організація виконання лабораторної роботи

Для виконання лабораторної роботи необхідно вивчити, використовуючи рекомендовану літературу, конспект лекцій і методичні вказівки, такі питання:

- структура програм мовою HTML;
- форматування таблиць мовою HTML.

Далі виконати такі дії:

– розробити web-сторінку, яка складається з головної HTML-програми (index.htm) і двох додаткових (link1.htm, photo.htm). Головна сторінка має посилання на програму link1.htm. Програма link1.htm – це пронумерований перелік друзів, з попередньої роботи, який поданий у вигляді таблиці. Таблиця має назву і стовпці, у рядки яких занесено ПІБ, дата народження, e-mail, номер контактного телефону та назва файлу фотографії друга. З програми link1.htm є посилання на програму photo.htm, яка має архів фотографій друзів. При кожному відображенні web-сторінки за допомогою додаткових програм повинна бути можливість повернення на головну або вихідну сторінку;

– перевірити роботу отриманих програм;

– записати результат роботи програм з екрана дисплея у файли з форматом jpg.

Підготувати звіт з виконання лабораторної роботи, який повинен включати:

- тему і мету лабораторної роботи;
- опис завдання з початковими умовами і даними;
- тексти програм мовою HTML згідно з варіантом;
- результати виконання програм.

Питання для підготовки до захисту лабораторної роботи

1. Які теги є основними для опису таблиці?
2. Як визначається кількість рядків таблиці?
3. Якими атрибутами визначається розташування даних в комірці таблиці за умовчанням?
4. Які теги використовуються для опису заголовка таблиці?
5. Який атрибут має тег <caption> за умовчанням?
6. Яким тегом визначається місце (зверху або знизу) назви таблиці?
7. Як визначається кількість комірок, які будуть об'єднані по горизонталі?
8. Як визначається кількість комірок, які будуть об'єднані по вертикалі?
9. Які html-елементи можуть містити комірки таблиць?
10. Яким атрибутом додається обрамлення до таблиць в HTML-документах?

СПИСОК ЛІТЕРАТУРИ

1. Олифер, В.Г. Компьютерные сети. Принципы, технологии, протоколы [Текст] : учеб. для вузов / В.Г. Олифер, Н.А. Олифер. – 3-е изд. – С.Пб.: Питер, 2006. – 958 с.
2. Тадеушевич, Р. Спільнота Інтернету [Текст]: навч. посібник / Р. Тадеушевич, Г.Г. Півняк, В.В. Ткачов. – Д.: Національний гірничий університет, 2005. – 258 с.
3. Страуструп, Б. Язык программирования С++ [Текст] / Б. Страуструп : пер. с англ. – М.: БИНОМ, С.Пб.: Невский Диалект, 2002. – 1099 с.
4. Цвіркун, Л.І. Розробка програмного забезпечення комп'ютерних систем. Програмування [Текст]: навч. посібник / Л.І. Цвіркун, А.А. Євстігнєєва, Я.В. Панферова. – 2-ге вид., випр. – Д.: Національний гірничий університет, 2011. – 222 с.
5. Мельник, Р.А. Програмування для Web- та SQL-серверів (PHP та Perl) [Текст]: / Р.А. Мельник. – Л.: Нац. ун-т "Львівська політехніка", 2006. – 131 с.
6. Гешвинде, Э. Разработка Web-приложений на PHP и Postgre SQL [Текст]: руководство разработчика и администратора / Э. Гешвинде, Г.Ю. Шениг. – С.Пб.: DiaSoft, 2003. – 598 с.
7. Дронов, В.А. PHP, MySQL и Dreamweaver MX 2004. Разработка интерактивных Web-сайтов [Текст]: / В.А. Дронов. – С.Пб.: БХВ-Петербург, 2005. – 433 с.
8. Цвіркун, Л.І. Робототехніка та мехатроніка [Текст]: навч. посібник / Л.І. Цвіркун, Г. Грулер. – 2-ге вид., випр. – Д.: Національний гірничий університет, 2010. – 222 с.

ПЕРЕЛІК СКОРОЧЕНЬ

АКС	–	автоматизації та комп'ютерні системи
БД	–	база даних
КПК	–	кишеньковий персональний комп'ютер
ARPA	–	Advanced Research Projects Agency
CGI	–	Common Gateway Interface
DNS	–	Domain name server
IP	–	Internet Protocol
TCP	–	Transmission Control Protocol
ICQ	–	омонім слів «I seek you»
FI	–	Forms Interpreter
FTP	–	File Transfer Protocol
PHP	–	Personal Home Page
HTML	–	Hyper Text Markup Language
WWW	–	World wide web

ПРЕДМЕТНИЙ ПОКАЖЧИК

Навчальне видання

Цвіркун Леонід Іванович
Липовий Роман Володимирович

ГЛОБАЛЬНІ КОМП'ЮТЕРНІ МЕРЕЖІ ПРОГРАМУВАННЯ МОВОЮ PHP

Навчальний посібник

Під загальною редакцією професора Л.І. Цвіркуна

Редактор Ю.В. Рачковська

Підп. до друку 24.05.13. Формат 30х42/4.
Папір офсет. Ризографія. Ум. друк. арк. 13,4.
Обл.-вид. арк. 13,4. Тираж 300 пр. Зам. №

Підготовлено до друку та видрукувано
у Державному ВНЗ "Національний гірничий університет".
Свідоцтво про внесення до Державного реєстру ДК № 1842 від 11.06.2004.
49005, м. Дніпропетровськ, просп. К. Маркса, 19.