Моя первая книга в TEX

Вася Пупкин

3 вересня 2012 р.

Зміст

Перелік ілюстрацій

Перелік таблиць

Лабораторная работа № 1

Основи мережі Internet

1.1 Структура мережі Internet

Інтернет — всесвітня інформаційна комп'ютерна мережа, що являє собою об'єднання безлічі регіональних комп'ютерних мереж і комп'ютерів, що обмінюються один з одним інформацією по каналах громадських телекомунікацій (виділених телефонних аналогових і цифрових лініях, оптичним каналам зв'язку і радіоканалах, в тому числі супутникових лініях зв'язку). Комп'ютери, підключені до мережі Інтернет, можуть мати будь-які апаратні і програмні платформи, але при цьому вони повинні підтримувати стек протоколів (сімейство протоколів) зв'язку TCP / IP.

Інформація в Інтернет зберігається на серверах. Сервери мають свої адреси і управляються спеціалізованим серверним програмним забезпеченням, яке дозволяє пересилати пошту, файли, проводити конференції, тощо.

Доступ окремих користувачів до інформаційних ресурсів Інтернету зазвичай здійснюється через провайдера або корпоративну мережу.

Провайдер (ISP, Internet Service Provider) — постачальник мережевих послуг — особа або організація надають послуги з підключення до комп'ютерних мереж. В якості провайдера виступає деяка організація, що має модемний пул для з'єднання з клієнтами та виходу у всесвітню мережу.

Користувачі підключаються до мережі через маршрутизатори місцевих постачальників послуг Інтернету, які мають постійне підключення до Інтернет через регіональних провайдерів. Регіональний провайдер, підключається до більшого провайдеру національного масштабу, що має вузли в різних містах країни.

Кожен комп'ютер, підключений до мережі Інтернет, має унікальну адресу. Адреси комп'ютерів бувають двох видів:

1. IP-адреса (обов'язкова)
2. DNS-имя (Domain Name System, доменне ім'я)

MAC-адреса не розглядається, оскільки у глобальній мережі вона не використовується для ідентифікації комп'ютерів.

IP-адрес (Internet Protocol Address) — унікальна мережева адреса вузла в комп'ютерній мережі, побудованій на основі IP-протоколу. **IPv4-адреси** складаються з чотирьох байтів, тобто з 32-розрядного двійкового числа, яке для зручності поділяється на чотири блоки по 8 бітів. **IPv6-адреси** відображаються як вісім груп по чотири шістнадцяткові цифри, розділені двокрапкою. Приклади IPv4 та IPv6 адрес:

```
IPv4: 192.0.2.235
IPv6: 2001:0db8:11a3:09d7:1f34:8a2e:07a0:765d
```

IP-адреса складається з двох частин: номера мережі й номера вузла (комп'ютера) в мережі. Якщо окремий комп'ютер (хост-комп'ютер) або мережа є складовою частиною мережі Інтернет, то IP-адреса присвоюється організація ICANN (Інтернет корпорація з присвоєння імен і номерів)

Доменне (DNS) ім'я — символьне ім'я, що служить для ідентифікації областей — одиниць адміністративної автономії в мережі Інтернет — у складі вищестоящої по ієрархії такої області. Доменну адресу побудований на основі ієрархічної класифікації, тобто доменну адресу включає в себе кілька рівнів доменів, наприклад: `google.com.ua`. Домен верхнього рівня розташовується в імені правіше, а домен нижнього рівня — лівіше. Користувач

мережі Інтернет працює не з IP-адресами, а тільки з доменними адресами. Перетворення доменного імені в IP-адресу здійснюють **DNS-сервера**.

Для повноцінної роботи в мережі передбачений ряд протоколів прикладного рівня. Протокол прикладного рівня — протокол верхнього (7-ого) рівня мережевої моделі OSI, забезпечує взаємодію мережі і користувача. Рівень дозволяє додаткам користувача мати доступ до мережевих служб, таким як обробник запитів до баз даних, доступ до файлів, пересилання електронної пошти. Також відповідає за передачу службової інформації, надає додаткам інформацію про помилки й формує запити до рівня подання. Приклад: HTTP, POP3, SMTP

1.2 Передача даних через протокол HTTP

Протокол передачі гіпертексту (Hypertext Transfer Protocol, HTTP) — протокол прикладного рівня для розподілених мультимедійних інформаційних систем.

Перші версії, такі, як HTTP/0.9, являли собою прості протоколи для передачі даних через Інтернет. Версія HTTP/1.0, поліпшила протокол, дозволивши використання повідомлень в форматі MIME, що містять метаінформацію про переданих даних, і модифікатори для запитів / відгуків.

Для опису характеру, найменування та місця розташування інформаційних ресурсів введені: **URI — Uniform Resource Indicator** (уніфікований ідентифікатор ресурсу), **URL — Uniform Resource Locator** (уніфікований визначник місцезнаходження ресурсу) **URN — Unifrorm Resource Name** (уніфіковане ім'я ресурсу). URI: Позначає ім'я та адресу ресурсу в мережі. Як правило ділиться на URL і URN, тому URL і URN це складові URI.

URL: Адреса деякого ресурсу в web. URL визначає місцезнаходження ресурсу і спосіб звертання до нього.

URN: Ім'я деякого ресурсу в web. Сенс URN в тому, що він визначає тільки назва конкретного предмета, який може знаходиться в безлічі конкретних місць.

Як приклад можна представити наступні посилання:

```
URI = http://handynotes.ru/2009/09/uri-url-urn.html
URL = http://handynotes.ru
URN = / 2009/09/uri-url-urn.html
```

HTTP використовується також в якості базового протоколу для комунікації користувацьких агентів з проксі-серверами і іншими системами Інтернет, у тому числі й використовуючих протоколи SMTP, NNTP і FTP.

Всі HTTP-транзакції мають один загальний формат. Кожен запит клієнта і відповідь сервера складається з трьох частин: рядки запиту (відповіді), розділу заголовка і тіла. Клієнт ініціює транзакцію наступним чином:

1. Клієнт встановлює зв'язок з сервером на номер телефону порту (за замовчуванням — 80). Потім клієнт посилає запит документа, вказавши HTTP-команду, яка називається методом, адреса документа і номер версії HTTP. Наприклад, в запиті

   ```
   GET / index.html HTTP/1.0
   ```

2. Клієнт посилає інформацію заголовка (необов'язкову), щоб повідомити серверу інформацію про свою конфігурації і дані про формати документів, які він може приймати.

   ```
   User-Agent: Mozilla/4.05 (WinNT; 1)
   Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
   ```

3. Надіславши запит і заголовки, клієнт може відправити і додаткові дані. Клієнти також можуть використовувати їх для приміщення відредагованій сторінки назад на Web-сервер.

Сервер відповідає на запит клієнта наступним чином:

1. Перша частина відповіді сервера — рядок стану, що містить три поля: версію HTTP, код стану й опис. Поле версії містить номер версії HTTP, яку даний сервер користується для передачі відповіді.

   Код стану — це триразрядне число, що позначає результат обробки сервером запиту клієнта. Опис, наступне за кодом стану, є просто зрозумілий для людини текст, що пояснює код стану. Наприклад, рядок стану `HTTP/1.0 200 OK` говорить про те, що сервер для відповіді використовує версію HTTP 1.0. Код стану 200 означає, що запит клієнта був успішним і витребувані дані будуть передані після заголовків.

2. Після рядка стану сервер передає клієнтові інформацію заголовка, що містить дані про самого сервері і викликаній документі. Нижче наведено приклад заголовка:

   ```
   Date: Fri, 10 Jan 1998 08:17:58 GMT
   Server: Apache/1.2.6
   Last-modified: Mon, 12 Jun 1997 21:53:08 GMT
   Content-type: text/html
   Content-length: 2482
   ```

3. Якщо запит клієнта успішний, то надсилаються витребувані дані. Це може бути копія файлу або результат виконання CGI-програми. Якщо запит клієнта задовольнити не можна, передаються додаткові дані у вигляді зрозумілого для користувача роз'яснення причин, за якими сервер не зміг виконати даний запит.

1.3 Мови програмування для web, web-сервери, мережеві СКБД

Мови програмування

Серверні мови web-програмування можуть бути умовно розділені по операційним системам, на яких вони працюють: Windows і *nix. Це розділення в деякій мірі умовно, тому що практично всі популярні мови і фреймворки портовано на усі популярні сімейства ОС. Тим не менш, вони рідко використовуються на нерідних ОС.

Якщо говорити про ОС Windows, то тут панує технологія **ASP.NET**, розроблена компанією Microsoft. За допомогою ASP.NET можна створювати сайти будь-якого рівня складності — від найпростіших, що складаються їх декількох сторінок, до дуже складних, що обробляють мільйони запитів на день.

Самим популярним мовою web-програмування є, безумовно, **PHP**. Його основними перевагами є: простий синтаксис, висока швидкодія, підтримка більшістю хостингів. Дуже вагомою перевагою є те, що на PHP написано багато популярних CMS.

Perl — це інтерпретована мова програмування. Завдяки цьому, грамотно написаний Perl-скрипт може працювати як в *nix, так і в Windows, як на процесорах x86, так і на Alpha-або Power PC.

JSP (Java Server Pages) — це частина технології J2EE, призначена для створення сайтів за допомогою мови Java. JSP має дуже багато спільного з ASP.NET і вибір між цими двома технологіями найчастіше ґрунтується на суб'єктивних перевагах та досвіду роботи з якоюсь іх технологій.

Останнім часом високу популярність придбав мову **Ruby** і, зокрема, фреймворк **Ruby On Rails**. З його допомогою можна дуже швидко створити сайт з необхідною функціональністю. Одним з істотних недоліків Ruby, є низька швидкодія.

Мова програмування **Python** сьогодні є одною із самих популярних інтерпретованих мов. Python, що є об'єктно-орієнтованою мовою, відмінно справляється з найрізноманітнішими завданнями, а міжплатформенність для цієї мови реалізована в повному обсязі.

Web-сервери

На даний момент найбільш поширеним web-сервером, що займає більше 65% ринку, є **Apache** — безкоштовний web-сервер, найбільш часто використовується в UNIX-подібних операційних системах.

У середовищі Windows, дуже поширений **IIS (Internet Information Services)** від компанії Microsoft, розповсюджуваний з серверними версіями ОС.

Nginx — перспективний web-сервер і поштовий проксі-сервер, розроблений для Unix-подібних операційних системах. Починаючи з версії 0.7.52 з'явилася бінарна збірка під Microsoft Windows.

Lighttpd — компактний web-сервер, розроблений для Unix-подібних операційних систем, портований надалі на платформу Windows. **Google Web Server (GWS)** — web-сервер, який використовується Google для організації своєї web-інфраструктури.

Мережеві СКБД

Реляційна модель баз даних являє собою централізоване сховище таблиць, що забезпечує безпечний одночасний доступ до інформації з боку багатьох користувачів. У рядках таблиць частина полів містить дані, що відносяться безпосередньо до запису, а частина — посилання на записи інших таблиць. Таким чином, зв'язки між записами є невід'ємною властивістю реляційної моделі.

Кожен запис таблиці має однакову структуру. Наприклад, у таблиці, яка містить описи автомобілів, у всіх записів буде один і той же набір полів: виробник, модель, рік випуску, пробіг і т.д. Такі таблиці легко зображувати в графічному вигляді.

В реляційній моделі СКБД досягається інформаційна та структурна незалежність. Записи не пов'язані між собою настільки, щоб зміна однієї з них торкнулося інші, а зміни структура СКБД, бази даних не обов'язково призводить до перекомпіляції працюючих з нею додатків. Для побудови сайтів використовують багатокористувацькі реляційні СКБД з підтримкою SQL. Як правило це **MS SQL Server**, **MySQL, Oracle**, **Interbase**, **DB2** або **PostgreSQL**. Вибір конкретної СКБД залежить від призначення сайту,

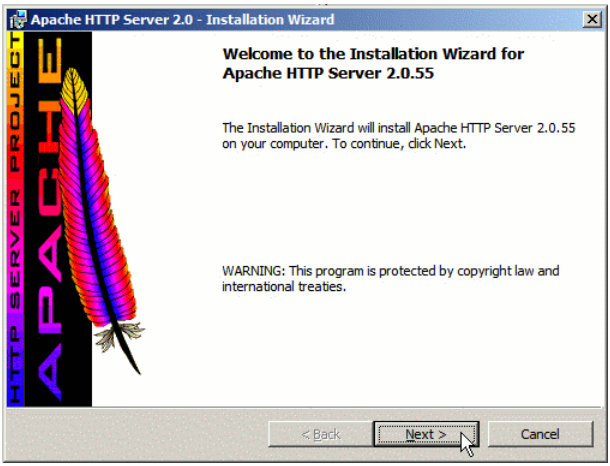

Рис. 1.1. Інсталятор Apache

планованих обсягів бази даних, навантаження на сервер, оптимальної для програміста ліцензії. Величезний відсоток web-хостингів базується на використанні MySQL і PostgreSQL. Цей вибір обумовлений широкими можливостями, наданими даними продуктами, а також їх вартістю.

1.4 Налаштування web-сервера Apache

Завантажити Apache можна з дзеркал наведених на офіційному сайті `http://www.apache.org/`. При пошуку слід пам'ятати, що Apache так само може називатися HTTPD, на ім'я його демона в UNIX. На дзеркалах зазвичай багато різних файлів, необхідно скачати бінарний файл для Windows *.exe або *.msi. У випадку якщо у вас система *nix-сімейства вам необхідно встановити пакет httpd потрібної версії з вашого репозиторію. Інсталяція програмного пакету для Windows не змінюється вже багато років (Рис. 1.1). Вам потрібно буде занести домен, в якому знаходиться сервер, ім'я сервера, e-mail адміністратора сервера. У даному випадку ці дані не важливі, їх можна залишити стандартними (Рис. 1.2). Також треба вибрати каталог, куди буде встановлюватися Apache, або залишити його стандартним

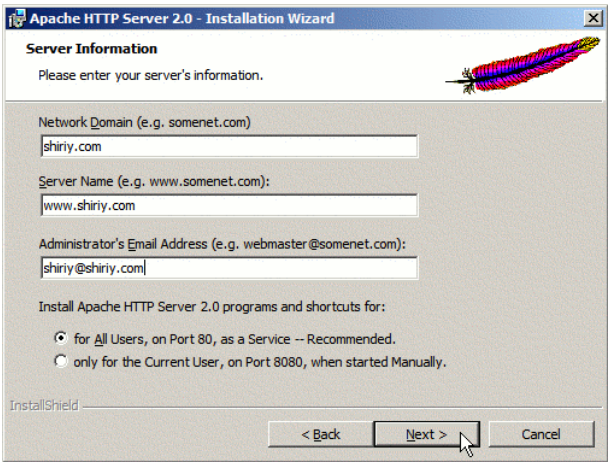

Рис. 1.2. Налаштування доменних імен

(Рис. 1.3). Подальші зміни в конфігурації Apache можна вносити, використовуючи файл «httpd.conf».

Щоб було зручно маніпулювати файлами ваших проектів створіть папку зі зрручним для вас коротким шляхом, наприклад `D:\Site`, в якій будуть зберігатися всі інші програми і дані сайту. Далі створіть папку `D:\Site\localhost\`, в якій створіть директорії WWW і CGI відповідно. WWW міститиме матеріали сайту, а CGI - скрипти CGI, якщо такі у вас будуть. В директорії ...`\Apache2\Conf\` знайдіть файл «httpd.conf» — це файл з налаштуваннями. У ньому знайдіть рядок

```
ServerRoot "C :/Program Files/Apache Group/Apache2"
```

Він повинна містити шлях до самого Апач, тобто на ту папку, куди у вас Апач встановлений. Зверніть увагу, що в шляху слеш прямий і закінчується адреса без слеша.

Далі прив'язуємо Apache до конкретного порту:

```
Listen 80
```

При деяких помилках сервера Apache відсилає електронні листи адміністратору, адреса поштової скриньки налаштовується у рядку

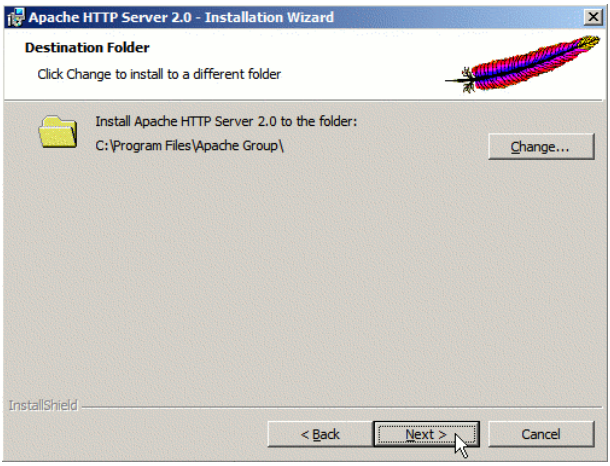

Рис. 1.3. Каталог розміщення Apache

```
ServerAdmin your@email.name
```

Тепер прописуємо шлях до даних сайту

```
DocumentRoot "D:/Site/localhost/WWW"
```

Знайдіть блок

```
<Directory "C:/Program Files/Apache Group/Apache2/htdocs">
```

і замініть його на

```
<Directory "D:/Site">
    Options Indexes Includes
    AllowOverride All
    Order allow,deny
    Allow from all
</Directory>
```

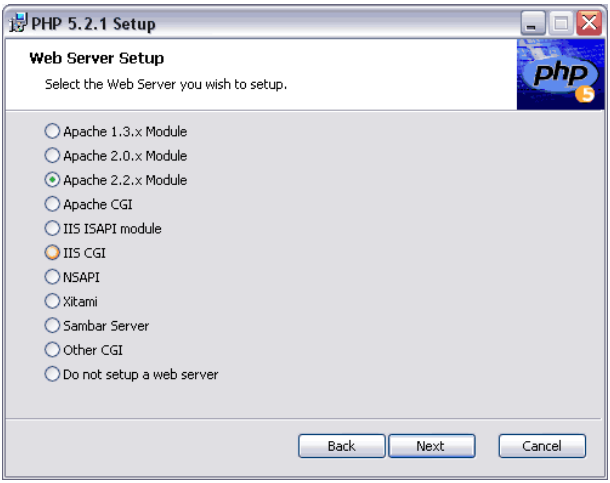

Рис. 1.4. Каталог розміщення PHP

1.5 Налаштування PHP

Встановлення PHP в середовищі Windows також не створює проблем. Завантажте встановлювач і запустіть його.

Необхідно буде вибрати каталог для встановлення інтерпретатору, встановлену версію Apache (Рис. 1.4). Також необхідно задати місцезнаходження файлу «httpd.conf» (Рис. 1.5). Виберіть усі розширення PHP, що йдуть у комплекті, так ви не зіткнетесь з проблемами недостачі бібліотек під час навчання (Рис. 1.6).

У випадку проблеми прив'язки PHP до Apache його можна підключити безпосередньо у файлі «httpd.conf». Для цього треба додати такі рядки:

```
LoadModule php5_module c:/(каталог з PHP)/php5apache2_2.dll
AddType application/x-httpd-php phtml php
PHPIniDir "c:/(каталог з PHP)/"
```

Для перевірки роботи PHP та Apache створіть файл у каталозі вашого сайту «phpinfo.php» з такими рядками:

```
<?php
```

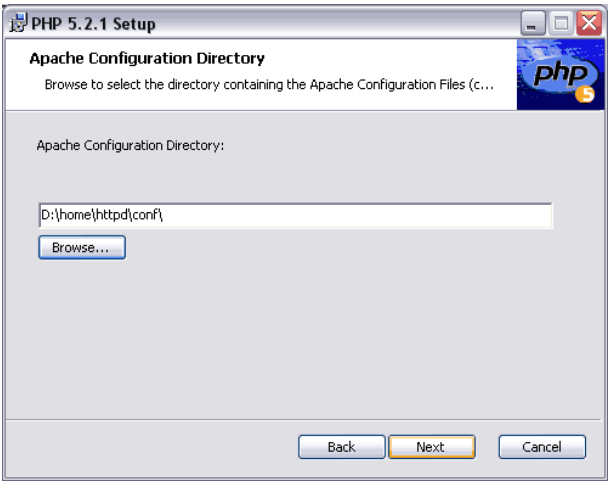

Рис. 1.5. Вибір місцезнаходження файлу «httpd.conf»

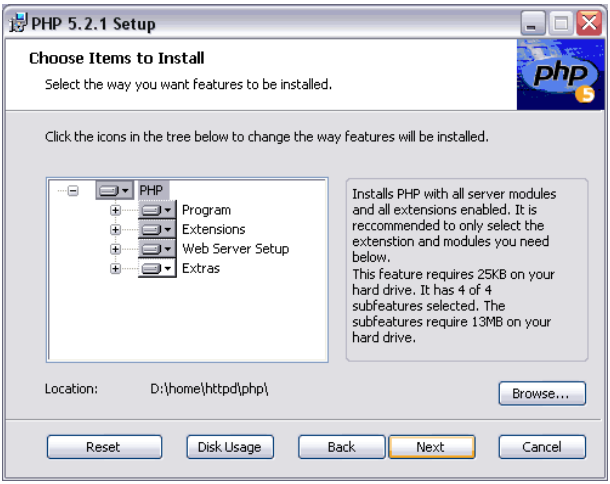

Рис. 1.6. Вибір встановлюваних бібліотек

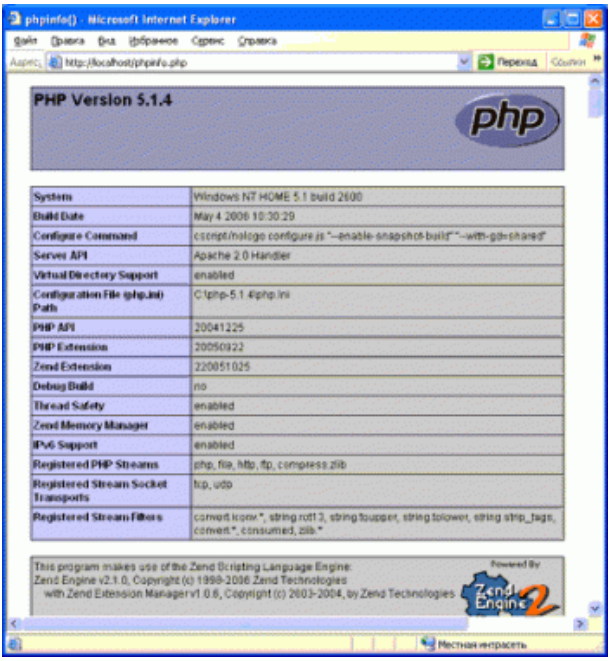

Рис. 1.7. Результат роботи функції phpinfo()

```
  echo phpinfo();
?>
```

В адресній строці браузера введіть `http://localhost/phpinfo.php`. Якщо ви все виконали правильно, ви повинні побачити сторінку, зображену на Рис. 1.7:

1.6 Налаштування СКБД MySQL

При встановленні СКБД MySQL вам необхідно запустити файл-інсталятор, даних за замовчуванням достатньо для встановлення повнофункціонального пакету. Після встановлення запуститься майстер налаштувань. Стандартних даних для роботи СКБД достатньо, одначе необхідно буде вказати пароль доступу адміністратора до СКБД в діалозі, зображеному на Рис. 1.8.

Рис. 1.8. Введення пароля адміністратора СКБД

Після встановлення пароля ви можете під'єднуватися до бази під логіном «root» та вказаним вами паролем.

1.7 Індивідуальне завдання

Завдання до лабораторної роботи

1. Вивчити теоретичний матеріал
2. Відповісти на контрольні запитання
3. Скласти звіт
4. Захистити роботу

Контрольні запитання

1. Що таке Internet? З яких структурних частин складається Internet?
2. Що таке IP-адреса?
3. Що таке доменне ім'я, з чого воно складається?

4. Який сервіс Internet перетворює IP-адреси в доменні імена і навпаки?
5. Яка служба займається розподіленням блоків IP-адрес?
6. Протокол HTTP. Рівень у моделі OSI, призначення.
7. Значення URI, URL, URN.
8. Мови web-програмування, які ви знаєте.
9. Веб-сервери, які ви знаєте.
10. Мережеві СКБД, які ви знаєте.

Лабораторная работа № 2

Форми HTML. Змінні, константи, масиви та функції в PHP

2.1 Формування HTML-сторінки засобами PHP

Код PHP зазвичай об'єднується з тегами XHTML. PHP є вбудовуваним мовою — це означає, що можна переміщатися між чистим кодом HTML і PHP, не жертвуючи можливістю читання тексту. Щоб вбудувати код PHP в XHTML, PHP повинен задаватися відокремлено, за допомогою початкового та кінцевого тегів PHP. Теги PHP кажуть інтерпретатору, де починається і закінчується код PHP. Аналізатор PHP розпізнає три варіанти початкового і кінцевого тегів.

1. Перший варіант тегів PHP називається тегами в стилі XML і є кращим стилем. Він працює в документах розширювана мова розмітки (XML). Цей метод повинен використовуватися при з'єднанні PHP з докумен-

тами XML і XHTML. Приклади в цьому підручнику застосовують цей формат тегів XML.

Стиль XML

```
<? PHPБлок
// коду PHP
>
```

2. Скорочений стиль є найбільш простим, однак, він не рекомендується, тому що вступає в протиріччя зі стандартами документів XML та налаштуваннями в «php.ini».

   Скорочений стиль

   ```
   <?Блок
   // коду PHP
   >
   ```

3. Цей стиль використовує найдовшу запис і схожий на стиль тегів, що застосовуються для включення кодів JavaScript. Цей стиль є кращим при використанні редактора HTML, який не розпізнає інші стилі тегів. Так як більшість нових редакторів XHTML розпізнають стиль тегів XML, то використання цього стилю не рекомендується.

   Стиль сценарію

   ```
   <script language="php">Блок
   // коду PHP
   </ SCRIPT>
   ```

PHP містить два основних оператора для виведення тексту в браузері Web: «echo» і «print». Обидва оператора розміщуються між відкритим і закритим тегами блоку коду PHP і можуть перебувати в будь-якому місці в документах XHTML. Оператори «echo» і «print» використовують наступний формат: «echo» - використовується для виведення одного або кількох рядків.

```
echo "виведений текст";
```

«print» - використовується для виведення рядка. В деяких випадках оператор «print» пропонує більшу функціональність, ніж оператор «echo».

```
print "виведений текст";
```

Наступні приклади демонструють використання і розміщення команд «echo» і «print» в документі XHTML.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD/XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml11-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <title>Страница Web</title>
</head>
<body>
<?php
echo "Друк рядка оператором echo"."<br>";
print "Друк рядка оператором print";
?>
</body>
</html>
```

2.2 Передача даних з HTML-форми PHP-сценарію

Обробка форм є дуже важливою властивістю PHP. За допомогою форм користувачі взаємодіють зі сторінками Web, і з їхньою ж допомогою можна збирати інформацію для персоналізованих сторінок відвідувачів. У більш широкому сенсі інформаційної обробки, форми призначені для введення даних в системи обробки. Вони є первинним механізмом отримання

даних, які обробляють сценарії для породження нової інформації, поновлення файлів і баз даних, а також для відповіді на запити користувачів для отримання інформації.

Передача даних з форм в PHP-сценарій відбувається методами GET або POST протоколу HTTP. Обидва методи однаково ефективні при використанні в формах з невеликою кількістю полів. Метод POST рекомендований при передачі великих обсягів тексту і файлів, обмеження за замовчуванням встановлено в 8 Мбайт і налаштовується у файлі php.ini.

Контейнер форми виглядає наступним чином:

```
<form name="form1" action="script.php" method="post">
</form>
```

де «name» - назва HTML-об'єкта, «action» - відносний або абсолютний шлях до сценарію, якому передаються дані, «method» - назва методу, яким передаються дані.

У контейнер форми поміщаються додаткові елементи управління, якими управляє користувач:

INPUT і його варіації

Елемент <input> є найбільш вживаним тегом HTML-форм. За допомогою цього тега реалізуються основні функції форми. Він дозволяє створювати всередині форми поля введення рядка тексту, імені файлу, пароля і т.д. Також варто згадати про варіацію тега, що реалізує можливість завантаження файлів на сервер. Повний опис можливостей даного тега дано у додатку A.1.1

TEXTAREA

Елемент <textarea> відповідає за передачу багаторядкового тексту. Важливо пам'ятати, що об'єм тексту, що передається обмежений параметрами методу, який використовується для передачі. Повний опис можливостей даного тега дано у додатку A.1.2

Списки вибору SELECT

Досить часто існує необхідність представити які-небудь дані у вигляді списку і передбачити можливість вибору в цьому списку. У HTML списки реалізуються за допомогою тега <select>. Списки можуть давати можливість одиночного або множинного вибору. Повний опис можливостей даного тега дано у додатку A.1.3

2.3 Змінні, константи та функції в PHP

Змінні

Змінні є тимчасовим місцем зберігання, використовуваним для представлення значень в сценарії PHP. У PHP є два основні типи змінних: скалярні і масиви. Скалярні змінні містять тільки одне значення в даний момент часу, а змінні масиви - список значень. Змінні масиви обговорюються в наступному розділі. Скалярні змінні PHP містять значення типів описаних у лабораторній роботі № 3.2.

Імена змінних PHP всіх типів починаються зі знака «$». Імена змінних можуть містити літери, числа, і символ підкреслення (_); вони не можуть, проте, починатися з цифри. У PHP імена змінних розрізняють регістр символів.

Інтерполяція змінних

PHP підтримує також процес, званий інтерполяцією - заміну змінної в рядку її вмістом. Замість з'єднання змінних і літералів, їх можна об'єднувати всередині подвійних лапок (). Змінні і літерали не можна об'єднати всередині одиночних лапок. При використанні подвійних лапок значення змінної виводиться разом з літералів.

```
<?php
```

```
$fname = "John";
$lname = "Doe";

echo "The user's name is $fname $lname";

?>
```

Константи

Константи, як і змінні, є тимчасовим сховищем значень у пам'яті. На відміну від змінних значення константи ніколи не змінюється. При оголошенні константи використовується функція define(), яка вимагає задати ім'я константи і значення цієї константи.

Константам можна присвоювати такі типи даних.

Цілі — цілі числа або числа без десяткової крапки (1, 999, 325 812 841).

Числа з плаваючою точкою — числа, що містять десяткову крапку (1.11, 2.5, .44).

Рядки — текстова або числова інформація. Строкові дані завжди полягають в лапки ("Hello World "478-477-5555").

Імена констант PHP на відміну від змінних не починаються зі знака "$". Імена констант звичайно записують у верхньому регістрі. Імена констант можуть містити літери, цифри та символ підкреслення (_); вони не можуть, проте, починатися з цифри. Оголошення констант показано нижче.

```
define ("STRING_CONSTANT", "This is my string.");
define ("NUMERIC_CONSTANT", 5);
```

Вивід констант подібний до виводу змінних.

Функції

Функції використовуються для розбиття великих блоків коду на менші, більш керовані одиниці. Код, що міститься всередині функції, виконує певне завдання і повертає значення. PHP містить два типи функцій — визна-

чені користувачем (або створені програмістом) і внутрішні (вбудовані функції), які є частиною визначення мови PHP.

Виклик вбудованої функції відбувається за допомогою її імені. Наприклад, функція, що виводить інформацію про PHP та Apache:

```
<?php
phpinfo();
?>
```

У даному випадку функція викликається без параметрів. Наступна функція використовує ряд аргументів і повертає значення (у даному випадку дескриптор відкритого файлу:

```
<?php
f=fopen("d:\www\index.php","w+");
?>
```

2.4 Робота з масивами в PHP

Змінну масиву можна використовувати для зберігання множини або послідовності значень. Система PHP підтримує масиви з числовими індексами і асоціативні масиви. Масив в PHP є фактично впорядкованим відображенням. Відображення є типом, який відображає значення в ключі. Змінні масивів складаються з двох частин — індексу та елемента. Індекс масиву, іноді званий ключем масиву, є значенням, застосовуваним для ідентифікації або доступу до елементів масиву. Індекс масиву поміщається в квадратні дужки. Більшість масивів використовують числові індекси, які зазвичай починаються з 0 або 1. У PHP асоціативні масиви можуть використовувати рядкові індекси. Обидва типи масивів створюються за допомогою конструкції array()

Масиви з числовими індексами

```
$my_array = array('red', 'green', 'blue');
```

Цей код створює масив з числовим індексом з ім'ям $my_array. Масиву присвоюється три елементи — «red», «green» і «blue». Кожен елемент ідентифікується числовим індексом.

```
$my_array [0] = 'red' // індекс 0 відповідає елементу red
$my_array [1] = 'green' // індекс 1 відповідає елементу green
$my_array [2] = 'blue' // індекс 2 відповідає елементу blue
```

Щоб отримати доступ до вмісту масиву, використовується ім'я масиву та індекс.

Асоціативні масиви

Асоціативні масиви дозволяють використовувати більш корисні значення індексу. Для масивів з числовими індексами значення індексу створюються автоматично, починаючи з 0. Асоціативні масиви допускають застосування числових і строкових значень індексу.

```
$members = array('FName' => 'John', 'LName' => 'Smith', 'Age' => 50)
```

У цьому прикладі члени масиву містять три елементи, однак використовуються рядкові індекси — FName, LName і Age.

```
$members ['FName'] = 'John' // індекс FName відповідає елементу John
$members ['LName'] = 'Smith' // індекс LName відповідає елементу Smith
$members ['Age'] = '50 '// індекс Age відповідає елементу 50
```

Для доступу до вмісту масиву використовується ім'я масиву та індекс.

2.5 Функції, визначені користувачем

Визначені користувачем функції створюються за допомогою ключового слова function. Вони особливо корисні у великих програмах PHP, так як можуть містити блоки коду, які можуть використовуватися в програмі, що дозволяє уникнути повторного переписування коду. Далі представлений приклад простої визначеної користувачем функції PHP:

```
function AddNumbers ($num1, $num2)
{
echo "Це приклад функції PHP.
Вона обчислює суму двох чисел і повертає
результат програмі, що її викликала";
return $num1 + $num2;
}
```

Визначені користувачем функції можуть викликатися в будь-якому місці програми на PHP. У PHP функція виконується при використанні в коді її імені. Після виклику функція отримує всі передані їй значення у формі параметрів, виконує певні завдання і повертає значення програмі.

2.6 Змінні всередині функції

Глобальні змінні — це змінні, які доступні всій програмі, включаючи підпрограми (користувальницькі функції).

Локальні змінні — змінні, визначені всередині підпрограми (користувацької функції). Вони доступні тільки всередині функції, в якій вони визначені.

Для PHP всі оголошені і використовувані у функції змінні за замовчуванням локальні для функції. Тобто, за умовчанням немає можливості змінити значення глобальної змінної в тілі функції.

існує спеціальна інструкція global, що дозволяє користувальницької функції працювати з глобальними змінними. Розглянемо даний принцип на конкретному прикладі:

```
<?php
$a = 1 ;
$b = 2 ;

function sum()
{
```

```
global $a, $b;
$b = $a + $b;
}
?>
```

2.7 Функції-змінні

PHP підтримує концепцію змінних функцій. Це означає, що якщо до імені змінної приєднані круглі дужки, PHP шукає функцію з тим же ім'ям, що і результат обчислення змінної, і намагається її виконати. Цю можливість можна використовувати для реалізації зворотних викликів, таблиць функцій і безлічі інших речей.

Приклад використання функції-змінної наведено в додатку A.2

2.8 Індивідуальне завдання

Завдання до лабораторної роботи

1. Вивчити теоретичний матеріал
2. Відповісти на контрольні запитання
3. Скласти алгоритм (блок-схему) програми
4. Виконати практичне завдання
5. Скласти звіт
6. Захистити роботу

Контрольні запитання

1. Що таке змінні, константи та функції?
2. Що таке інтерполяція змінних?

3. Що таке масиви?

4. Як отримати доступ до індексного масиву?

5. До ассоціативного?

6. Як створити користувацьку функцію?

7. Що таке локальні змінні?

8. Що таке глобальні змінні? Як ними користуватись у тілі функції?

9. Що таке функції-змінні?

10. Яким чином здійснюється виклик функції-змінної?

Практичні завдання

Написати HTML-сторінку з формою, що складається з:

1. однорядкового поля вводу, поля вводу пароля та кнопки відправлення форми.

2. однорядкового поля вводу, прихованого текстового поля та кнопки відправлення форми.

3. однорядкового поля вводу, багаторядкового поля вводу та кнопки відправлення форми.

4. багаторядкового поля вводу, списку з одиночним вибором з п'яти елементів та кнопки відправлення форми.

5. прихованого текстового поля, списку з одиночним вибором з п'яти елементів та кнопки відправлення форми.

6. багаторядкового поля вводу, списку з множинним вибором з п'яти елементів та кнопки відправлення форми.

7. списку з одиночним вибором з п'яти елементів, поля вводу пароля та кнопки відправлення форми.

8. списку з множинним вибором з п'яти елементів, багаторядкового поля вводу та кнопки відправлення форми.

   Отримати дані форми і вивести їх за допомогою іншого сценарію. Написання програми в одному PHP-файлі схвалюється.

9. створити асоціативний масив з днями тижня та вивести його на сторінку

10. створити індексний массив з назвами місяців і вивести його на екран

    Написати HTML-сторінку з формою, що складається з однорядкового поля вводу та кнопки відправлення. Записати в поле число та обробити наступним чином:

11. створити константу та помножити на отримане з форми число

12. створити змінну, занести в неї результат з форми, помножити змінну саму на себе

13. створити константу та обчислити вираз $x * const + 2 * x$

14. створити константу та обчислити вираз $\frac{x}{const} + x^2$

15. створити змінну та константу, в змінну занести константу і додати до числа з форми.

    Результат вивести на сторінку

    Створити форму з трьома однорядковими полями і кнопкою відправлення форми

16. отримані з форми дані занести в асоціативний масив

17. отримані дані занести в індексний масив

    зміст масиву роздрукувати функцією print_r();

18. створити функцію, що друкує свій параметр

19. створити функцію, що перемножує два свої параметри і повертає результат. Результат роздрукувати

20. створити функцію, що перемножує свої параметри і друкує результат

21. створити функцію, що змінює глобальну змінну

22. створити функцію, що перемножує глобальну змінну і вхідний параметр, рузультат друкує

23. створити функцію, що домножує глобальну змінну на вхідний параметр і повертає результат

24. створити функцію, що формує індексний масив з двох локальних змінних і друкує його функцією print_r();

25. створити функцію, що формує асоціативний масив з параметрів і друкує його функцією print_r();

Лабораторная работа № 3

Взаємодія PHP та web-сервера. Синтаксис PHP

3.1 Змінні оточення web-сервера Apache, суперглобальні масиви PHP

Змінні оточення web-сервера Apache

Змінні оточення - дуже важливий механізм взаємодії веб-сервера з предобработчікамі запитів. При отриманні HTTP-запиту веб-сервер формує змінні оточення, заносячи в них різну інформацію: IP-адреса клієнта, запитуваний документ, параметри запиту і т.п. При передачі управління якомусь предобработчіку останній має доступ до змінних оточення веб-сервера, отже, йому доступна вище перерахована інформація.

Безпосередньо перед запуском сценарію сервер передає йому якісь змінні оточення з інформацією. Змінні оточення в мові PHP можна використовувати як звичайнісінькі змінні. Змінні оточення діляться на чотири великі групи:

1. Формовані сервером змінні;

2. Спеціальні змінні сервера Apache;

3. Змінні HTTP-полів запиту;

4. Змінні SSL-з'єднання (захищеного з'єднання).

Суперглобальні масиви PHP

Суперглобальними масивами (superglobal arrays) в PHP називаються зумовлені масиви, які видно в будь-якому місці вихідного коду без використання ключового слова **global**.

Доступ до змінних оточення здійснюється через суперглобальний масив $_SERVER у вигляді $_SERVER['змінна_CGI']; Наприклад код

```
print $_SERVER['SERVER_SOFTWARE'];
```

роздрукує в браузері строку виду:

```
Apache/2.2.4 (Win32) mod_ssl/2.2.4 OpenSSL/0.9.8k PHP/5.3.3
```

Повний перелік змінних оточення web-сервера Apache та суперглабальних масивів PHP дивіться у додатках B.1 та B.2.

3.2 Оператори PHP

Синтаксис PHP дуже нагадує синтаксис мови C і багато в чому запозичений з таких мов як Java і Perl. Інструкції поділяються також як і в C або Perl — кожен вираз закінчується крапкою з комою. Закриваючий тег «?»> також має на увазі кінець інструкції.

Коментарі застосовуються в PHP для запису власних зауважень під час процесу розробки коду. Такі коментарі можуть визначати призначення сегмента коду або їх можна використовувати для виключення блоків коду під час тестування і налагодження сценаріїв.

Синтаксичний аналізатор PHP ігнорує коментарі. Коментарі в PHP можна визначити одним з наступних способів:

1. `//` - Простий коментар PHP;
2. `#` - Альтернативний простий коментар PHP;
3. `/* ... */` - Багаторядкові блоки коментарів.

Імена змінних позначаються знаком $.

```
<?php
$message = "Привет, я - скрипт PHP!";
echo $message;
?>
```

PHP підтримує вісім простих типів даних :

1. Чотири скалярних типи:
   - ◇ Boolean (двійкові дані)
   - ◇ Integer (цілі числа)
   - ◇ Float (числа з плаваючою точкою або 'double')
   - ◇ String (рядки)
2. Два змішаних типи:
   - ◇ Array (масиви)
   - ◇ Object (об'єкти)
3. Два спеціальних типи:
   - ◇ resource (ресурси)
   - ◇ NULL («порожні»)
4. Існують також кілька псевдотіпов:
   - ◇ Mixed (змішані)
   - ◇ Number (числа)
   - ◇ Callback (зворотний визов)

Основними конструкціями мови PHP є:

1. Умовні оператори (if, else);
2. Цикли (while, do-while, for, foreach, break, continue);
3. Конструкції вибору (switch);
4. Конструкції оголошення (declare);
5. Конструкції повернення значень (return);
6. Конструкції включень (require, include).

Для здійснення операцій зі змінними існують різні групи операторів. Оператором називається дещо, що складається з одного або більше значень (виразів), яке можна обчислити як нове значення (таким чином, вся конструкція може розглядатися як вираз).

Оператори бувають трьох видів.

- ◇ Унарні оператори, які працюють тільки з одним аргументом, наприклад, «!» (оператор заперечення) або «++» (інкрімент).
- ◇ Бінарні оператори: до них належать більшість підтримуваних в PHP операторів
- ◇ тернарних оператор «. . . ?. . . :. . . ». Він використовується для умовного вибору між двома операторами, в залежності від результату обчислення третього оператора.

Пріоритет виконання операторів

Пріоритет операторів визначає, в якому порядку будуть обчислюватися два і більше вирази. Наприклад, вираз $1+5*3$ обчислюється як 16, а не 18, оскільки операція множення «*» має більш високий пріоритет, ніж операція додавання «+». У разі, якщо оператори мають однаковий пріоритет, вони

будуть виконуватися зліва направо. Круглі дужки можуть використовуватися для примусового вказівки необхідного порядку виконання операторів. Наприклад, вираз $(1 + 5) * 3$ обчислюється як 18.

В таблиці B.1 додатку надано пріоритети виконання операторів PHP.

Ліва асоціативність вказує на те, що вираз обчислюється зліва направо, права асоціативність відповідно має увазі протилежний порядок.

3.3 Конструкції вибору

До конструкцій вибору відносять: умовний оператор (if. .. else) і перемикач (switch).

if . . . else

Синтаксис умовного оператора:

```
if (condition) statement 1 else statement 2
```

Умова condition може бути будь-яким виразом. Якщо condition істинне, то виконується оператор statement 1. В іншому випадку виконується оператор statement 2. Допустима скорочена форма запису умовного оператора, в якій відсутні else і оператор statement 2.

У свою чергу, оператори statement 1 і statement 2 можуть бути умовними, що дозволяє організовувати послідовність перевірок будь-якої глибини вкладеності. І в цих послідовностях кожен умовний оператор може бути як повним, так і скороченим. У зв'язку з цим можливі помилки неоднозначного зіставлення if і else.

Зауважимо, що перевірка додаткових умов можлива за допомогою оператора «elseif». Оператор «if» може включати скільки завгодно блоків «elseif», але else в кожному if може бути тільки один. Як правило, в конструкціях «if . . . elseif . . . else» оператор else визначає, що потрібно робити, якщо ніякі інші умови не є true.

PHP надає також можливість альтернативного синтаксису умовного оператора - без фігурних дужок, а із застосуванням оператора «endif».

Перемикач «switch»

Перемикач «switch» є найбільш зручним засобом для організації множинного розгалуження. Синтаксис перемикача такий:

```
switch (expression)
{
case value1: statements; break;
case value2: statements; break;
default:
statements;
}
```

Керуюча структура «switch» передає керування тому з помічених «case» операторів, для якого значення константного виразу збігається зі значенням expression. Якщо значення expression не збігається ні з одним з константних виразів, то виконується перехід до оператора, поміченого міткою «default». У кожному перемикачі може бути не більше однієї мітки «default», однак вона може бути відсутньою взагалі.

3.4 Конструкції циклів

На другому місці за частотою використання, після конструкцій умов (умовних операторів), знаходяться цикли.

Цикли дозволяють повторювати певну (і навіть невизначений — коли робота циклу залежить від умови) кількість разів різні послідовності операторів. Дані оператори називаються тілом циклу. Прохід циклу називається ітерацією.

PHP підтримує чотири види циклів:

- ◇ Цикл з передумовою (while)

◇ Цикл з постусловіем (do-while)

◇ Цикл з лічильником (for)

◇ Спеціальний цикл перебору масивів (foreach)

При використанні циклів є можливість використання операторів «break» і «continue». Перший з них перериває роботу всього циклу, а другий — тільки поточної ітерації.

Цикл з передумовою while

Цикл з передумовою while працює за такими принципами:

1. Обчислюється значення логічного виразу.
2. Якщо значення істинно, виконується тіло циклу, в іншому випадку — переходимо на наступний за циклом оператор.

Синтаксис циклу з передумовою:

```
while (логічний_вираз) інструкція;
```

В даному випадку тілом циклу є інструкція. Зазвичай тіло циклу складається з великої кількості операторів. Наведемо приклад циклу з передумовою while:

```
<? Php
$x = 0;
while ($x++<10) echo $x;
/ / Виводить 12345678910
?>
```

Зверніть увагу на послідовність виконання операцій умови $x++<10. Спочатку перевіряється умова, а тільки потім збільшується значення змінної. Якщо ми поставимо операцію інкремента перед змінної (++$x<10), то спочатку буде виконано збільшення змінної, а тільки потім — порівняння. У результаті ми отримаємо рядок «123456789».

Подібно конструкції умовного оператора «if», можна групувати оператори всередині тіла циклу while, використовуючи наступний альтернативний синтаксис:

```
while (логічний_вираз):
інструкція;
...
endwhile;
```

Цикл з постумовою do while

На відміну від циклу «while», цей цикл перевіряє значення виразу не до, а після кожного проходу (ітерації). Таким чином, тіло циклу виконується хоча б один раз. Синтаксис циклу з постусловіем такий:

```
do
{
тело_цікла;
}
while (логічний_вираз);
```

Після чергової ітерації перевіряється, чи істинний логічний_вираз, і, якщо це так, управління передається знову на початок циклу, в іншому випадку цикл обривається.

Цикл з лічильником for (регулярний цикл)

Цикл з лічильником використовується для виконання тіла циклу певне число разів.

Синтаксис циклу for такий:

```
for (ініціалізуючі_команди; умова_роботи; команди_після_ітерації) {тіло_
```

Цикл for починає свою роботу з виконання ініціалізуючіх_команди. Дані команди виконуються тільки один раз. Після цього перевіряється умова_роботи, якщо воно істинно (true), то виконується тіло_цикла. Пі-

сля того, як буде виконаний останній оператор тіла, виконуються команди_після_ітерації.

Для циклу for є і альтернативний синтаксис:

```
for (ініціалізуючі_команди; умова_роботи; команди_після_ітерації):
операторы;
endfor;
```

Цикл перебору масивів foreach

В PHP4 з'явився ще один спеціальний тип циклу — «foreach». Даний цикл призначений спеціально для перебору масивів.

Синтаксис циклу foreach виглядає наступним чином:

```
foreach (масив as $ключ => $значення)
команди;
```

Тут команди циклічно виконуються для кожного елемента масиву, при цьому чергова пара «ключ => значення» виявляється в змінних $ключ і $значення.

У циклу foreach є й інша форма запису, яку слід застосовувати, коли нас не цікавить значення ключа чергового елемента. Виглядає вона так:

```
foreach (масив as $значення)
команди;
```

У цьому випадку доступно лише значення чергового елементу масиву, але не його ключ. Це може бути корисно, наприклад, для роботи з масивами-списками:

Увага: Цикл foreach оперує не вихідним масивом, а його копією. Це означає, що будь-які зміни, які вносяться в масив, не можуть бути «видимі» з тіла циклу. Що дозволяє, наприклад, в якості масиву використовувати не тільки змінну, але і результат роботи якої-небудь функції, що повертає масив (у цьому випадку функція буде викликана всього один раз — до початку циклу, а потім робота буде проводитися з копією повернутого значення).

Конструкція break

Дуже часто для того, щоб спростити логіку якого-небудь складного циклу, зручно мати можливість його перервати в ході чергової ітерації (наприклад, при виконанні якого-небудь особливого умови). Для цього і існує конструкція break, яка здійснює негайний вихід з циклу. Вона може задаватися з одним необов'язковим параметром — числом, яке вказує, з якого вкладеного циклу має бути здійснений вихід. За замовчуванням використовується 1, тобто вихід з поточного циклу, але іноді застосовуються й інші значення. Синтаксис конструкції break:

```
break; / / За замовчуванням
break (номер_цикла); / / Для вкладених циклів
(вказується номер циклу, що переривається)
```

Конструкція continue

Конструкція continue так само, як і break, працює тільки «в парі» з циклічними конструкціями. Вона негайно завершує поточну ітерацію циклу та переходить до нової (звичайно, якщо виконується умова циклу для циклу з передумовою). Точно так само, як і для break, для continue можна вказати рівень вкладеності циклу, який буде продовжений з повернення управління.

3.5 Конструкції включення

Конструкції включень дозволяють збирати PHP програму (скрипт) з декількох окремих файлів.

У PHP існують дві основні конструкції включень: require і include.

Конструкція включень require

Конструкція require дозволяє включати файли в сценарій PHP до виконання сценарію PHP. Загальний синтаксис require такий:

```
require им'я_файлу;
або
require (им'я_файлу);
```

При запуску (саме при запуску, а не при виконанні!) Програми інтерпретатор просто замінить інструкцію на вміст файлу им'я_файлу (цей файл може також містити сценарій на PHP, обрамлений, як зазвичай, тегами <? Та?>). Причому зробить він це безпосередньо перед запуском програми (на відміну від include , який розглядається нижче). Це буває досить зручно для включення в висновок сценарію різних шаблонних сторінок HTML-кодом.

Конструкція continue

Конструкція include також призначена для включення файлів в код сценарію PHP.

На відміну від конструкції require конструкція include дозволяє включати файли в код PHP скрипта під час виконання сценарію. Синтаксис конструкції include виглядає наступним чином:

```
include им'я_файлу;
або
include (им'я_файлу);
```

Принципова різниця між цими двома операторами в тому, що include дозволяє включати файли «на літу», і робити це декілька разів, наприклад у циклах.

Конструкції одноразового включення require_once і include_once

У великих PHP сценаріях інструкції include і require застосовуються дуже часто. Тому стає досить складно контролювати, як би випадково не включити один і той же файл декілька разів, що найчастіше призводить до помилки, яку складно виявити.

У PHP передбачено вирішення даної проблеми. Використовуючи конструкції одноразового включення require_once і include_once, можна бути впевненим, що один файл не буде включено двічі. Працюють конструкції одноразового включення require_once і include_once так само, як і require і include відповідно. Різниця в їх роботі лише в тому, що перед включенням файлу інтерпрететор перевіряє, чи включений вказаний файл раніше чи ні. Якщо так, то файл не буде включено знову.

Включення віддалених файлів

Конструкції однократних включень require_once і include_ince також дозволяють включати віддалені файли, якщо така можливість включена в конфігураційному файлі PHP.

Якщо URL fopen увімкнено в PHP (як у конфігурації за замовчуванням), ви можете специфікувати файл, що підключається з використанням URL (через HTTP), замість локального шляху.

Для того, щоб віддалене включення файлів було доступно, необхідно в конфігураційному файлі (php.ini) встановити allow_url_fopen = 1.

3.6 Індивідуальне завдання

Завдання до лабораторної роботи

1. Вивчити теоретичний матеріал
2. Відповісти на контрольні запитання
3. Скласти алгоритм (блок-схему) програми
4. Виконати практичне завдання
5. Скласти звіт
6. Захистити роботу

Контрольні запитання

1. Які оператори виводу тексту ви знаєте?
2. Які групи змінних оточення ви знаєте?
3. Що таке суперглобальні масиви, як ними користуватись?
4. Які варіації тега <input> ви знаєте?
5. Як організувати список вибору?
6. Які конструкції вибору ви знаєте?
7. Які конструкції циклів ви знаєте?
8. У якому випадку цикл виконується хоча-б один раз?
9. Якими чином можна включити інший текстовий файл у сценарій?
10. Яка особливість операторів include_once та require_once?

Практичні завдання

Створити окремий PHP-файл, підключити його у головну сторінку. Файл повинен:

1. У регулярному циклі виводити числа від 1 до 20
2. У циклі з передумовою виводити числа від 20 до 50
3. У циклі з постумовою вивести числа від -50 до -10
4. Використовуючи регулярний цикл вивести елементи арифметичної прогресії з 20 елементів починаючи з 1 та доданком 2
5. Використовуючи з передумовою вивести елементи геометричної прогресії з 20 елементів починаючи з 1 та доданком 2
6. Розрахувати факторіал 10 за допомогою регулярного циклу

7. Вивести латинський алфавіт, використовуючи регулярний цикл
8. Вивести латинський алфавіт, використовуючи цикл з передумовою
9. Вивести Цифри від 20 до 0 у зворотньому порядку, використовуючи цикл з постумовою.

   Написати HTML-сторінку з формою, що складається з однорядкового поля вводу та кнопки відправлення форми. В поле вводу ввести число та обчислити вираз:

10. якщо $x > 0$ то $5 * x^2 + 2 * x - 10$, інакше $5 * x^2 + 2 * x + 10$
11. якщо $0 < x \leq 10$ то $10 * x^3 - 2 * x - 10$, інакше $x^2 + 20 * x$
12. якщо $0 < x \leq 20$ то $\frac{x}{x+1} + 2 * x - 10$, інакше $\frac{x+1}{x} + 10$
13. якщо $-10 < x \leq 0$ то $5 * x^2 + 2 * x - 10$, $x > 0$ $5 * x^2 + 2 * x + 10$, інакше $\frac{x+1}{x} + 10$
14. якщо $x \leq 10$ то $x^3 + 2 * x^2$, інакше $2 * x + 10$
15. якщо $x > 10$ то $5 * x^2 + \frac{x}{2}$, інакше $\frac{x+10}{x+1}$

   Написати HTML-сторінку з формою, що складається з однорядкового поля вводу та кнопки відправлення форми. В поле вводу ввести число та використовуючи конструкцію «switch-case»:

16. за номером дня тижня вивести його назву.
17. за номером місяця року вивести його назву.
18. від 0 до 10 вивести назви цифр
19. відповідно номера вивести букву латинського алфавіту

   Опрацювати змінні оточення

20. вивести шлях до каталогу з веб-документами
21. вивести підпис веб-сервера

22. вивести інформацію про браузер користувача

23. вивести адресу електронної пошти адміністратора

24. за допомогою конструкції перебору масивів вивести зміст $_SERVER

25. за допомогою конструкції перебору масивів вивести зміст $_ENV

Лабораторная работа № 4

Робота з рядками. Регулярні вирази

4.1 Рядки. Функції роботи з рядками

Рядки є послідовностями символів. У PHP символ відповідає байту, тобто існує точно 256 можливих різних символів. Рядки можуть бути дуже великими. У PHP не існує практичного обмеження на розмір рядків, тому взагалі немає причин турбуватися про їх довжині. Строкові значення можуть використовуватися буквально або присвоюватися змінним.

У PHP строковий літерал можна представляти трьома способами.

- ◇ рядки в одиночних лапках
- ◇ рядки в подвійних лапках
- ◇ рядки в синтаксисі heredoc

Рядки в одиночних лапках

Одиночні лапки надають найпростіший метод для роботи з рядками. При використанні цього методу рядки укладаються в одиночні лапки (”).

Якщо одиночні лапки потрібні як частина рядка, вони повинні бути екрановані символом зворотної косої межі (\). Хоча одиночні лапки надають простий спосіб роботи з рядками, одиночні лапки не підтримують застосування інтерполяції.

Рядки в подвійних лапках

Рядки PHP можна виводити також за допомогою подвійних лапок (). Якщо рядки PHP поміщаються в подвійні лапки, то можна застосовувати інтерполяцію. Для рядків у подвійних лапках PHP підтримує також більшість екранованих символів. Ці символи представлені в таблиці 4.1 нижче.

Табл. 4.1. Екрановані символи у подвійних лапках

Символ	Опис
\N	перенесення рядка
\R	повернення каретки
\T	горизонтальна табуляція
\\	зворотна коса риска
\$	знак долара
\"	подвійні лапки

Рядки в синтаксисі heredoc

Heredoc-синтаксис — спосіб визначення строкових змінних у вихідному коді програм.

При визначенні строкових змінних їх вміст, зазвичай, полягає в одинарні або подвійні лапки, у зв'язку з чим символи лапок, які повинні бути частиною даних, доводиться екранувати за допомогою escape-послідовностей

. Heredoc-синтаксис дозволяє визначити рядок, не укладаючи її в лапки, у зв'язку з чим необхідність екранування цих символів відпадає.

Приклад використання такого синтаксису наведено нижче:

```
$S = << EOL;
Лапки бувають 'одинарними' і "подвійними".
EOL
```

Функції роботи з рядками

У базовому наборі PHP існує величезна кількість функцій для обробки рядків. Як правило їх достатньо для написання програм, іноді необхідно комбінувати ці функції між собою для отримання необхідного результату. Повний перелік функцій дан у таблиці C.1 додатків.

4.2 Рядки, що включають HTML-код

Досить часто ми працюємо з рядками, що містять html-теги. Якщо відобразити такий рядок в браузері за допомогою звичайних функцій відображення даних `echo()` або `print()`, то ми не побачимо самих html-тегів, а отримаємо відформатовану у відповідності з цими тегами сторінку. Браузер обробляє всі html-теги у відповідності зі стандартом мови HTML. Іноді нам потрібно бачити безпосередньо рядок, без обробки його браузером. Щоб цього досягти, потрібно перед тим, як виводити, застосувати до рядка функцію `htmlspecialchars()`.

4.3 Синтаксис регулярних виразів

Регулярний вираз (regular expression) — це технологія, яка дозволяє задати шаблон і здійснити пошук даних, відповідних цьому шаблону, в заданому тексті, представленому у вигляді рядка. Одне з поширених застосувань регулярних виразів — це перевірка рядка на відповідність будь-яким правилам.

Приклад регулярного виразу:

```
/^\w+([\.\w]+)*\w@\w((\.\w)*\w+)*\.\w{2,3}$/
```

Основна перевага РВ полягає в тому, що вони дозволяють організувати більш гнучкий пошук, тобто знайти те, про що немає точного знання, але є приблизне уявлення. Наприклад, потрібно знайти всі семизначні номери телефонів, що зустрічаються в тексті. Ми не шукаємо якийсь заздалегідь відомий нам номер телефону, ми знаємо тільки, що шуканий номер складається з семи цифр. Для цього можна скористатися наступним РВ:

```
/\d{3}-\d{2}-\d{2}/m
```

Основні функції для роботи з Perl-сумісними регулярними виразами: `preg_match (pattern, string, [result, flags])` і `preg_match_all (pattern, string, result, [flags])`, де:

`pattern` — шаблон регулярного виразу;

`string` — рядок, в якій проводиться пошук;

`result` — містить масив результатів (нульовий елемент масиву містить відповідність всьому шаблоном, перший - першому "захопленому"подшаблону і т.д.);

`flags` — необов'язковий параметр, що визначає те, як впорядковані результати пошуку.

Ці функції здійснюють пошук за шаблоном і повертають інформацію про те, скільки разів відбулося збіг. Для `preg_match()` це 0 (немає збігів) або 1, оскільки пошук припиняється, як тільки знайдено перший збіг. Функція `preg\_match\_all()` робить пошук до кінця рядка і тому знаходить всі збіги. Всі точні збіги містяться в першому елементі масиву result у кожної з цих функцій (для `preg\_match\_all()` цей елемент — теж масив).

4.4 Функції PHP для роботи з регулярними виразами

4.5 Регулярні вирази в SQL-запитах

4.6 Індивідуальне завдання

Завдання до лабораторної роботи

1. Вивчити теоретичний матеріал
2. Відповісти на контрольні запитання
3. Скласти алгоритм (блок-схему) програми
4. Виконати практичне завдання
5. Скласти звіт
6. Захистити роботу

Контрольні запитання

1. Що таке змінні, константи та функції?
2. Що таке інтерполяція змінних?
3. Що таке масиви?
4. Як отримати доступ до індексного масиву?
5. До ассоціативного?
6. Як створити користувацьку функцію?
7. Що таке локальні змінні?
8. Що таке глобальні змінні? Як ними користуватись у тілі функції?
9. Що таке функції-змінні?
10. Яким чином здійснюється виклик функції-змінної?

Практичні завдання

Написати HTML-сторінку з формою, що складається з:

1. однорядкового поля вводу, поля вводу пароля та кнопки відправлення форми.
2.

Лабораторная работа № 5

Работа с файлами

5.1 Создание, открытие и закрытие файла

5.2 Блочные чтение и запись

5.3 Построчные чтение и запись

5.4 Перемещение указателя по файлу

5.5 Копирование, перемещение, блокировка, удаление файла

5.6 Загрузка файла на сервер

Лабораторная работа № 6

Сессии в PHP. Отправка e-mail

6.1 Механизм сессии. Настройки сессий

6.2 Создание сессий, регистрация переменных сессий, удаление переменных сессий

6.3 Работа с cookies

6.4 Отправка e-mail

Лабораторная работа № 7

Взаимодействие PHP с СУБД MySQL

7.1 Функции для работы с MySQL

7.2 Установка соединения. Выбор таблицы

7.3 Выборка из таблицы, разбор результата выборки

7.4 Добавление записей, обновление записей

7.5 Очистка и удаление таблицы

Лабораторная работа № 8

ООП. Работа с XML

8.1 Классы и объекты

8.2 Наследование

8.3 Конструкторы

8.4 Операторы «::» и «parent»

8.5 Объектная модель XML-документа

8.6 Расширения SAX и DOM для работы с XML

Лабораторная работа № 9

Работа с сокетами. Пересылка данных JSON

9.1 Структура сокета. Открытие и закрытие сокета

9.2 Запись в сокет и чтение из сокета

9.3 Синтаксис JSON, его структуры

9.4 Кодирование и декодирование JSON в PHP

9.5 Обмен данными с JavaScript-приложением

Лабораторная работа № 10

Работа с изображениями

10.1 Создание изображения

10.2 Рисование простых геометрических фигур

10.3 Рисование текста на изображении

10.4 Изменение размера изображения

10.5 Функции библиотеки GD

Додаток А

А.1 Елементи форм HTML

А.1.1 INPUT і його методи

Однорядкові поля введення даних

```
<input type=text name=им'я_параметра [value=значення]
[size=розмір_поля] [maxlen=довжина_поля]>
```

Даний тег створює поле вводу з максимально допустимою довжиною тексту «maxlen» і розміром в size «знакомест». Якщо вказаний атрибут «value», то в поле буде спочатку відображатися значення даного атрибуту. У квадратних дужках [] позначені необов'язкові атрибути.

Поля вводу паролів

```
<input type=password name=им'я_параметра [value=значення]
[size=розмір_поля] [maxlen=довжина_поля]>
```

Структура даного тегу така сама як і у <input type=text>. Різниця лише у відображенні даних, що вводить користувач.

Приховане текстове поле

```
<input type=hidden name=им'я_параметра [value=значення]>
```

Такі поля передають дані серверу, але не відображаються на сторінці. Значення атрибуту «value» встановлюється при формуванні сторінки, або JavaScript-сценарієм.

Незалежні перемикачі

```
<input type=checkbox name=им'я_параметра [value=значення]
[checked]>
```

Перемикач виду «прапорець». У разі встановлення прапорця при відправлені форми серверу будуть передані параметри «им'я_параметра=значення». Якщо прапорець не встановлено серверу взагалі нічого не буде відправлено.

Перемикач за замовчуванням вимкнутий, щоб зробити його увімкнутим за замовчуванням треба встановити атрибут «checked».

Стан перемикача не залежить від стану інших перемикачів цього типу.

Залежні перемикачі

Залежний перемикач, так само як і незалежний перемикач, може заходитись у двох станах, в залежності від атрибуту «checked». При цьому на формі може бути увімкнений лише один перемикач серед групи перемикачів з однаковим атрибутом «name».

```
<form action="http://localhost/script.php" method="GET">
<input type=radio name=answer value=yes checked>Да
<input type=radio name=answer value=no>Нет
<input type=submit value=Отправить>
</form>
```

Кнопки відправлення та очищення параметрів форми

Кнопка відправки служить для передачі серверу змісту форми на сторінці. Атрибут «value» визначає текст, ща відображається на кнопці. Під

час відправлення форми серверу будуть передані дані кнопки у вигляді «им'я_параметра=значення».

```
<input type=submit [name=им'я_параметра] value=значення>
```

У разі використання кнопки із зображенням сереру передадуться координати кліку відносно зображення.

```
<input type=submit [name=им'я_параметра] src=зображення>
```

Кнопка очищення форми знищує всі зміни внесені користувачем сайту у дану форму.

```
<input type=reset [name=им'я_параметра] value=значення>
```

Поле вибору файлу

Тег <input> також дозволяє активувати діалогове вікно вибору файлу та завантажувати його на сервер при відправленні форми.

```
<input type=file name=имя [value=имя_файла]>
```

A.1.2 Багаторядкове текстове поле

Синтаксис багаторядкового поля виглядає наступним чином:

```
<textarea name=имя [cols=ширина_в_символах]
[rows=высота_в_символах] wrap=тип_переноса>
текст за замовчуванням
</textarea>
```

Хоча висота і ширина поля необов'язкові параметри їх бажано вказувати. Атрибут «wrap» відповідає за перенос і може приймати наступні значення:

1. Virtual — справа від тексту з'являється полоса прокрутки, а текст розбивається на рядки.
2. Physical — залежить від браузера і виглядає по-різному
3. None — текст залишається у тому вигляді, в якому користувач його ввів, з'являються горизонтальна і вертикальна полоси прокрутки.

A.1.3 Списки з одиночним вибором

Списки з одиночним вибором реалізуються за допомогою наступної конструкції:

```
<select name=day size=1>
<option value=1>Понедельник</option>
<option value=2>Вторник</option>
<option value=3 selected>Среда</option>
<option value=4>Четверг</option>
<option value=5>Пятница</option>
<option value=6>Суббота</option>
<option value=7>Воскресенье</option>
</select>
```

При відправленні форми сервер отримає дані виду «им'я_параметра=значення». За замовчуванням може бути обраний пункт списку серед атрибутів якого є «selected».

Списки з множинним вибором

Список з множинним вибором відрізняється лише атрибутом «multiple» в середині тега <select>.

```
<select name=day size=7 multiple>
<option value=1>Понедельник</option>
<option value=1>Вторник</option>
<option value=1>Среда</option>
<option value=1>Четверг</option>
<option value=1>Пятница</option>
<option value=1>Суббота</option>
```

```
<option value=1>Воскресенье</option>
</select>
```

A.2 Функції-змінні

Приклад використання змінної функції:

```
<?php
?php
function foo() {
    echo "In foo()<br />\n";
}

function bar($arg = '')
{
    echo "In bar(); argument was '$arg'.<br />\n";
}

// Функция-обертка для echo
function echoit($string)
{
    echo $string;
}

$func = 'foo';
$func();        // Вызывает функцию foo()

$func = 'bar';
$func('test');  // Вызывает функцию bar()
```

```
$func = 'echoit';
$func('test');  // Вызывает функцию echoit()
```

Додаток В

В.1 Змінні оточення web-сервера Apache

Формовані сервером змінні

AUTH_TYPE Використовується схема аутентифікації. Зазвичай BASIC

CONTENT_LENGTH Довжина вмісту, наприклад, текст / HTML

CONTENT_TYPE MIME-тип вмісту, наприклад, текст / HTML

GETAWAY_INTERFACE Версія CGI, наприклад CGI/1.1

PATH_info HTTP-шлях до сценарію

PATH_TRANSLATED Повний шлях до сценарію

REMOTE_ADDR IP-адреса запитуваного комп'ютера-клієнта

REMOTE_HOST Доменне ім'я запитувача комп'ютера (якщо доступно). Доменне ім'я визначається веб-сервером за допомогою служби DNS. Директива HostnameLookups сервера Apache дозволяє (або забороняє) перетворення IP-адреси в доменне ім'я.

REMOTE_PORT Порт, закріплений за браузером для отримання відповіді від сервера

REMOTE_USER Ім'я користувача, що пройшов аутентифікацію

QUERY_STRING Рядок переданих серверу параметрів

SERVER_ADDR IP-адресу сервера

SERVER_NAME Доменне ім'я сервера. Визначається директивою ServerName файлу конфігурації

SERVER_PORT TCP-порт Web-сервера. Зазвичай 80

SERVER_PROTOCOL Версія протоколу HTTP. Наприклад, HTTP/1.1

SERVER_SOFTWARE Програмне забезпечення сервера

SCRIPT_NAME HTTP-шлях до сценарію

SCRIPT_FILENAME Файл сценарію в файловій системі сервера (фізичний шлях). Наприклад, /VAR/WWW/CGI-BIN/script.cgi

Спеціальні змінні сервера Apache

DOCUMENT_ROOT Фізичний шлях до кореневого WWW-каталогу сервера. Наприклад, /var/www.html/

SERVER_ADMIN Адреса електронної пошти адміністратора сервера

SERVER_SIGNATURE Підпис сервера. Наприклад, "Apache/1.3.3 сервера на www.somefirm.com порт 80"

Змінні HTTP-полів запиту

HTTP_HOST Ім'я віртуального хоста, якому адресовано запит

HTTP_USER_AGENT Програмне забезпечення віддаленого користувача. Зазвичай ця змінна оточення містить назву і версію браузера

HTTP_ACCEPT Список підтримуваних клієнтом типів інформації.

HTTP_ACCEPT_LANGUAGE Список підтримуваних мов в порядку переваги, наприклад, RU, EN

HTTP_ACCEPT_ENCODING Список підтримуваних методів стиснення

HTTP_ACCEPT_CHARSET Список підтримуваних кодувань

HTTP_CONNECTION Тип з'єднання. Можливі два варіанти: •

- Keep-Alive - якщо після відповіді на запит не потрібно розривати з'єднання;
- Close - якщо потрібно закрити з'єднання відразу після відповіді на запит.

HTTP_REFERER Значення поля REFERER. У цьому полі браузер передає URL ресурсу, який посилається на наш сервер. Наприклад, якщо користувач перейшов на сайт зі сторінки http://www.somehost.com/page.php, то значення поля REFERER буде http://www.somehost.com/page.php.

HTTP_X_FORWARDED_FOR Якщо користувач працює через проксі-сервер, то в цьому полі буде IP-адреса комп'ютера, який звернувся до проксі-сервера. Якщо це поле вже містить значення, то нове значення буде додано через кому.

B.2 Суперглобальні масиви PHP

$GLOBALS - масив всіх глобальних змінних (у тому числі і для користувача).

$_SERVER - містить безліч інформації про поточний запит і сервер.

$_ENV - поточні змінні середовища. Їх набір специфічний для кожної конкретної платформи, на якій виконується сценарій.

$_GET - асоціативний масив з параметрами GET-запиту. У початковому вигляді ці параметри доступні в $_SERVER ['QUERY_STRING'] і в $_SERVER ['REQUEST_URI'] в складі URI.

$_POST - асоціативний масив значень полів HTML-форми при відправки методом POST.

$_FILES - асоціативний масив з відомостями про надіслані методом POST файлах. Кожен елемент має індекс ідентичний значенню атрибута «name» у формі і, в свою чергу, також є масивом з наступними елементами:

1. $_FILES['name'] - вихідне ім'я файлу на комп'ютері користувача.
2. $_FILES['type'] - зазначений агентом користувача MIME - тип файлу.

3. $_FILES['size'] - розмір файлу в байтах.

4. $_FILES['tmp_name'] - повний шлях до файлу в тимчасовій папці.

5. $_FILES['error'] - код помилки.

$_COOKIE - асоціативний масив з переданими агентом користувача значеннями cookie.

$_REQUEST - загальний масив ввідних даних запиту користувача як в масивах $_GET, $_POST, $_COOKIE. Починаючи з версії PHP 4.1 включається і вміст $_FILES.

B.3 Пріоритети виконання операторів

Пріоритети виконання операторів та їхня асоціативність показана на таблиці B.1.

Табл. B.1. Пріоритети виконання операторів

Асоціативність	Оператор
неасоціативна	`new`
права	`[`
неасоціативна	`++ --`
неасоціативна	`! ~ -(int) (float) (string) (array) (object) @`
ліва	`* / %`
ліва	`+ - .`
ліва	`<< >>`
неассоціативна	`< <= > >=`
неассоціативна	`== != === !==`
ліва	`&`
ліва	`^`
ліва	`|`
ліва	`&&`

ліва	`||`
ліва	`? :`
права	`= += -= *= /= .= %= &= |= ^= <<= >>=`
ліва	`and`
ліва	`xor`
ліва	`or`
ліва	`,`

Додаток С

С.1 Функції роботи з рядками

Табл. С.1. Повний список функцій роботи з рядками

Функція	Опис
addcslashes	екрануючі спецсимволи в стилі мови С
addslashes	екрануючі спецсимволи в рядку
bin2hex	Перетворює бінарні дані у шістнадцятірічне подання
chr	Повертає символ за його кодом
chunk_split	Розбиває рядок на фрагменти
convert_cyr_string	Перетворює рядок з одного кириличної кодування в інше
count_chars	Повертає інформацію про символи, що входять в рядок
crc32	Обчислює CRC32 для рядка
crypt	Необоротне шифрування (хешування)
echo	Виводить одну чи більше рядків
explode	Розбиває рядок на підрядки
fprintf	Записує отформатированную рядок у потік
get_html_translation_table	Повертає таблицю перетворень

hebrev	Перетворює текст на івриті з логічного кодування у візуальне
hebrevc	Перетворює текст на івриті з логічнго кодування у візуальне з перетворенням в переклад
htmlentities	Перетворює символи у відповідні HTML теги
htmlspecialchars	Перетворює спеціальні символи в HTML теги
html_entity_decode	Перетворює HTML теги в відповідні символи
implode	Об'єднує елементи масиву в рядок
localeconv	Повертає інформацію про числові формати
ltrim	Видаляє пробіли з початку рядка
md5	Повертає MD5-хеш рядка
md5_file	Повертає MD5-хеш файлу
metaphone	Повертає ключ metaphone для рядка
nl2br	Вставляє HTML-код розриву рядка перед кожним переведенням рядка
number_format	Форматує число з поділом груп
ord	Повертає ASCII-код символу
parse_str	Розбирає рядок у змінні
print	Виводить рядок
printf	Виводить відформатований рядок
quoted_printable_decode	розкодує рядок, закодовану методом quoted printable
quotemeta	екрануючі спеціальні символи
rtrim	Видаляє пробіли з кінця рядка
sha1	Повертає SHA1-хеш рядка
sha1_file	Повертає SHA1-хеш файлу
similar_text	Обчислює ступінь схожості двох рядків

soundex	Повертає ключ soundex для рядка
sprintf	Повертає відформатований рядок
sscanf	Розбирає рядок у відповідності із заданим форматом
strcasecmp	Порівняння рядків без урахування регістра, безпечне для даних у двійковій формі
strcmp	Порівняння рядків, безпечне для даних у двійковій формі
strcoll	Порівняння рядків з урахуванням поточної локалі
strcspn	Повертає довжину ділянки на початку рядка, не відповідного
масці	
stripcslashes	Видаляє екранування символів, вироблене функцією addcslashes ()
stripos	Повертає позицію першого входження підрядка без урахування регістра
stripslashes	Видаляє екранування символів, вироблене функцією addslashes ()
strip_tags	Видаляє HTML і PHP теги з рядка
stristr	Аналог функції strstr, але незалежний від регістру
strlen	Повертає довжину рядка
strnatcasecmp	Порівняння рядків без урахування регістра з використанням алгоритму
strnatcmp	Порівняння рядків з використанням алгоритму "природнього упорядкування"
strncasecmp	порівняння перших n символів рядків без урахування регістра, безпечне для даних у двійковій формі

strncmp	порівняння перших n символів рядків без урахування регістра, безпечне для даних у двійковій формі
strpos	Знаходить перше входження підрядка в рядок
strrchr	Знаходить останнє входження символу в рядок
strrev	Перевертає рядок
strripos	Повертає позицію останнього входження підрядка без урахування регістра
strrpos	Знаходить останнє входження символу в рядок
strspn	Повертає довжину ділянки на початку рядка, відповідного масці
strstr	Знаходить перше входження підрядка
strtok	Розбиває рядок
strtolower	Перетворює рядок у нижній регістр
strtoupper	Перетворює рядок у верхній регістр
strtr	Перетворює задані символи
str_ireplace	Регістро-незалежний варіант функції str_replace ().
str_pad	Доповнює рядок інший рядком до заданої довжини
str_repeat	Повертає повторювану рядок
str_replace	Замінює рядок пошуку на рядок заміни
str_rot13	Виконує над рядком перетворення ROT13
str_shuffle	перемішує символи в рядку
str_split	Розбиває рядок в масив
str_word_count	Повертає інформацію про слова, що входять в рядок
substr	Функція повертає частину рядка

substr_count	Підраховує кількість входжень підрядка в рядок
substr_replace	Замінює частину рядка
trim	Видаляє пробіли з початку та кінця рядка
ucfirst	Перетворює перший символ рядка в верхній регістр
ucwords	Перетворює у верхній регістр перший символ кожного слова в рядку
vprintf	Виводить відформатований рядок
vsprintf	Повертає відформатований рядок
wordwrap	Виконує перенесення рядка на дану кількість символів з використанням символу розриву рядка