

Міністерство освіти та науки України  
Тернопільський національний педагогічний університет  
імені Володимира Гнатюка

Кафедра інформатики та методики її викладання

**Балик Н.Р., Олексюк В.П., Мандзюк В.І., Вельгач А.В.**

**РНР.  
Методичні рекомендації  
для проведення практики  
з Web-програмування**



Тернопіль – 2011

ББК 32.973я7

РНР. Методичні рекомендації для проведення практики з Web-програмування.. — Тернопільський національний педагогічний університет імені В.Гнатюка, 2005. — 56 с.

Розробники: Н.Б. Балик, В.П. Олексюк, В.І. Мандзюк.

Рецензенти:

Фриз М. Є., доцент кафедри комп'ютерних наук Тернопільського державного технічного університету ім. І. Пулюя, кандидат технічних наук.

Маланюк П.М., доцент кафедри інформатики та методики викладання інформатики Тернопільського національного педагогічного університету імені В.Гнатюка.

*Затверджено вченою радою Тернопільського національного педагогічного університету імені В.Гнатюка  
Протокол №8 від 31.05.2005*

ББК 32.81я73

© Тернопільський національний педагогічний університет ім. В.Гнатюка, 2005

## Передмова

Однією з вимог професійної підготовки вчителя інформатики є вміння створювати та використовувати у навчальному процесі освітні ресурси.

Найбільш поширеними на практиці є гіпертекстові програмні засоби навчального призначення, які, переважно, містять статичні Web-сторінки, створені засобами мови розмітки гіпертексту HTML.

Разом з цим використання програмних засобів навчального призначення повинне забезпечувати:

- Рациональний розподіл навчальних ресурсів, що передбачає реалізацію технології „клієнт-сервер”.
- Зворотній зв'язок користувача із програмою.
- Централізоване зберігання та опрацювання даних.

Усе це може бути реалізовано за допомогою серверних мов програмування, однією з найбільш поширених з яких є мова PHP.

Завдання комп'ютерної практики полягає у створенні освітнього Web-порталу складовими якого є:

1. Система реєстрації користувачів.
2. Лічильник відвідувачів сайту.
3. Результати голосування.
4. Електронна дошка оголошень.
5. Новини.
6. Календар.
7. Бібліотека ресурсів.

Найбільш раціональним методом виконання поставленого завдання є метод проектів. Кожна група студентів одержує для розробки одну із складових освітнього Web-порталу.

Базовий рівень підготовки студентів передбачає:

- навички процедурного програмування;
- розуміння основ функціонування мережених технологій;
- знання основних команд мови розмітки гіпертексту HTML;

Для виконання завдань практики необхідні програмні засоби:

1. Web- сервер Apache, IIS або подібний до них.
2. Інтерпретатор мови PHP
3. Сервер СКБД MySQL
4. Засоби управління сервером СКБД PHPMyAdmin або MySQL-Admin.
5. Web-редактор із підтримкою синтаксису мови PHP Macro-media DreamViewer
6. Web-браузер Internet Explorer

# Лабораторна робота №1

## Перше знайомство з PHP

### *Теоретичні відомості.*

PHP - це мова програмування, код якої вбудовується безпосередньо в HTML-сторінку. Програму, написану на PHP, називають PHP-скриптом. При запиті користувача **web-сервер** переглядає документ, виконує знайдені в ньому PHP-інструкції (оператори мови PHP), а результат їхнього виконання повертає користувачеві. При цьому статична частина документа, написана мовою HTML, фактично є шаблоном, а змінювана частина формується при виконанні PHP-інструкцій. Для віддаленого користувача подібні документи нічим не відрізняються від звичайних статичних HTML-документів, за винятком того, що в розширенні імені файлу для таких документів може стояти не htm або html, а php (а в старих версіях phtml або php3).


PHP - це система розробки скриптів, що включає в себе CGI - інтерфейс, інтерпретатор мови та набір функцій для доступу до баз даних і різних об'єктів WWW (функції для роботи з електронною поштою, FTP, http, тощо). На сьогодні, PHP є одним із найбільш зручних і водночас достатньо потужним засобом розробки додатків WWW і інтерфейсів до баз даних в мережі Інтернет.

PHP-скрипти знаходяться на сервері і їхній вміст відвідувачеві сайту переглянути неможливо. Файли скриптів мають розширення \*.php. При активації скрипту (запиті користувача) серверна програма виконує всі команди php цього скрипту, не торкаючись статичної частини документа (HTML-код) і результат повертається програмі-браузеру. Після виконання PHP-скрипта користувач бачить звичайну веб-сторінку, яка відрізняється від інших тільки розширенням.

Розглянемо найпростіші інструкції мови PHP. Для цього виконаємо наступні вправи.

### *Вправа 1*

Створимо найпростішу Web-сторінку в якій використаємо PHP-скрипт. Для цього використаємо найпростіший текстовий редактор (наприклад "Блокнот" він же Notepad) та доступ до Web-сервера, який "вміє" обробляти PHP-скрипти.

 *Щоб запустити текстовий редактор блокнот слід клацнути на кнопці Пуск → Программи → Стандартные → Блокнот (В англійській версії Windows Start ->Programs -> Accessories -> Notepad)*


У "Блокноті" напишемо наступний код

```


<html>
<head>
<title> Сторінка з PHP </title>
</head>
<body>
<?php
    echo 'Hello world!!!'
?>
</body>
</html>

```

Далі зберігаємо створений файл в папці на Web-сервері (не забуваємо дати файлу розширення php). Назвемо його first\_prog.php.

 Щоб зберегти файл з розширенням php на Web-сервері слід у меню програми Блокнот вибрати Файл → Сохранить как (File → Save as). Відкриється діалог зберігання файлу, в ньому слід зайти в папку, яка розміщена на Web-сервері, в полі Имя файла (File name) ввести first\_prog.php, а в полі Тип файла (File type) вибрати Все файлы (All files), далі клацнути по кнопці Сохранить (Save)

Для того, щоб переглянути нашу першу Web-сторінку, слід відкрити її в веб-браузері. Для цього запускаємо наприклад Internet Explorer.

 Щоб запустити Internet Explorer слід клацнути на кнопку Пуск → Программы → Internet Explorer (Start → Programs → Internet Explorer)

У полі адреси в браузері пишемо шлях до файлу на сервері:

http://<ім'я серверу>/<шлях до папки з файлом>/first\_prog.php

Для прикладу, якщо ім'я серверу srv, а файл зберегли в папці users/delf то в браузері слід написати http://srv/users/delf/first\_prog.php, натискаємо клавішу Enter.

Якщо все зроблено правильно, то наша сторінка виглядатиме так, як на Рис. 1




**Рис. 1**

Якщо на сторінці напис не Hello world!!!, а текст подібний до

**Parse error: expecting `,' or `;' in c:\inetpub\wwwroot\labs\ first\_prog.php on line 9,**


тоді ви припустилися помилки при написанні php-скрипту, уважно перевірте текст програми.

 У повідомленнях про помилки PHP-інтерпретатор вказує на файл і номер стрічки з помилкою.

Якщо браузер вивів повідомлення, що файл не знайдено (File not found, Страница не найдена, Невозможно отобразить страницу), то ви неправильно набрали шлях до файлу на сервері.

Розглянемо детальніше файл `first_prog.php`. У першій стрічці тег `<html>` означає початок html-документу, а в кінці файлу розміщений відповідний йому закриваючий тег `</html>`, далі описується “голова” документа (між тегами `<head>` і `</head>`), текст що розміщений між тегами `<title>` `</title>` є заголовком сторінки і відображається в заголовку вікна браузера. Весь вміст сторінки розміщується між тегами `<body>` `</body>`. Власне php-скрипт вставлений між тегами `<?php` і `?>`, в запропонованому прикладі скрипт складається лише з одного оператора *echo*, який призначений для виводу однієї або більше стрічок (в даному прикладі одну стрічку ‘Hello Word!!!’). Короткий довідник тегів розмітки html-документу наведений в додатку 1.

Саме текст, що розміщений між тегами, `<?php` і `?>` обробляється на сервері php-інтерпретатором (зайві символи пробілів, табуляції, тощо ігноруються), користувачу замість цих символів і php-скрипту виводиться лише результат роботи цього скрипту. У цьому ви можете переконатись, проглянувши html-код вашої Web-сторінки, яка відкрита в браузері.

 Для того щоб проглянути html-код відкритої Web-сторінки, слід в головному меню *Internet Explorer* слід вибрати Вид – > *Просмотр HTML-кода (View -> View Source)*

Отже, результатом роботи php-скрипту є html-сторінка, яка формується динамічно в процесі виконання скрипту. Саме ця здатність динамічно формувати html-сторінки, в залежності від зовнішніх параметрів переданих скрипту, і є основною перевагою скриптів, що виконуються на сервері, в тому числі і php-скриптів.

## Лабораторна робота № 2

### Змінні в PHP

#### *Теоретичні відомості*

У цій лабораторній роботі ми познайомимось з такими об'єктами мови php, як змінні. Із змінними ви зустрічались вже в інших мовах програмування (наприклад, у Pascal). У php зміст поняття «змінна» не відрізняється від інших мов програмування. Імена змінних в php починаються з символу \$ (наприклад, \$i, \$I,\$count), регістр при найменуванні змінних має важливе значення, тому \$i та \$I це дві різні змінні. Оператором присвоєння є символ = (аналог := у мові Pascal).

PHP підтримує наступні типи змінних:

Ціле число (Integer);

Подвійної точності із плаваючою комою (Double);

Стрічка (String);

Масив (Array);

Об'єкт (Object);

Pdfdoc (тільки, якщо допускається підтримка формату PDF);

Pdfinfo (тільки, якщо допускається підтримка формату PDF);

Опис типу змінної не є обов'язковим і тому програмістом може бути і не визначений. Кожна змінна автоматично перетворюється в кожній з типів й різні функції використовують потрібний тип. Проте, існує декілька функцій, для яких важливий тип змінної. Для ініціалізації (визначення) змінної необхідно їй просто присвоїти значення.

```
<?php
$a = 5; - змушує змінну $a стати змінною типу Integer
$b = 5.0; - змушує змінну $b стати змінною типу Double
$c = "5"; - змушує змінну $c стати змінною типу String
?>
```

При використанні змінних типу string можна використовувати як подвійні так і одинарні лапки. Але між ними є різниця :

```
<?
$opilla='Оболонь';
$s='Пиво $opilla';
echo $s;
//виводить Пиво $opilla
echo '<br>';
$s="Пиво $opilla";
echo $s;
// виводить Пиво Оболонь
?>
```

Змінна розглядається як масив, якщо до її імені додається значення індекса, взятого у квадратні дужки []. Наприклад

```
<?php
$a[0] = 10;
$a[1] = 14;
$a[2] = 8;
//Дістали масив $a з трьох елементів
?>
```

Індексом масиву може бути або ціле число (Integer) або стрічка (string), елементами масиву можуть бути значення будь-якого типу.

```
<?
$a['name']='shoues';
$a['color']='red';
$a['size']=36;
?>
```

Масив, в якому індексами є стрічки називається асоціативним.

### Операції

Операції, що визначені в мові php, описані в таблицях 1 та 2

#### Арифметичні операції

Таблиця 1

Приклад	Назва	Результат
$\$a + \$b$	Додавання	Сума $\$a$ і $\$b$
$\$a - \$b$	Віднімання	Різниця $\$a$ і $\$b$
$\$a * \$b$	Множення	Добуток $\$a$ і $\$b$
$\$a / \$b$	Ділення	Частка від ділення $\$a$ на $\$b$
$\$a \% \$b$	Остача	Остача від цілочисельного ділення $\$a$ на $\$b$

#### Логічні операції

Таблиця 2

Приклад	Назва	Результат
$\$a == \$b$	дорівнює	<b>TRUE</b> , якщо $\$a$ дорівнює $\$b$ .
$\$a === \$b$	ідентично	<b>TRUE</b> , якщо $\$a$ дорівнює $\$b$ і вони одного типу. (тільки в PHP 4)
$\$a != \$b$	не дорівнює	<b>TRUE</b> , якщо $\$a$ не дорівнює $\$b$ .
$\$a <> \$b$	не дорівнює	<b>TRUE</b> , якщо $\$a$ не дорівнює $\$b$ .



<code>\$a !== \$b</code>	не ідентично	<b>TRUE</b> , якщо \$a не дорівнює \$b або вони різних типів. (тільки в PHP 4 і більше)
<code>\$a &lt; \$b</code>	менше	<b>TRUE</b> , якщо \$a строго менше \$b.
<code>\$a &gt; \$b</code>	більше	<b>TRUE</b> , якщо \$a строго більше \$b.
<code>\$a &lt;= \$b</code>	менше або дорівнює	<b>TRUE</b> , якщо \$a менше або дорівнює \$b.
<code>\$a &gt;= \$b</code>	більше або дорівнює	<b>TRUE</b> , якщо \$a більше або дорівнює \$b.
<code>! \$a</code>	Заперечення	<b>TRUE</b> , якщо \$a не <b>TRUE</b> .
<code>\$a &amp;&amp; \$b</code>	Логічне «і»	<b>TRUE</b> , якщо і \$a, і \$b <b>TRUE</b> .
<code>\$a    \$b</code>	Логічне «або»	<b>TRUE</b> , якщо \$a або \$b <b>TRUE</b> .

### **Стрічкові операції**

Основними операціями із стрічками є:

- конкатенація ('.') – повертає об'єднання із правого та лівого аргументів.
- присвоєння ('.=') – приєднує правий аргумент до лівого аргументу.

```
<?
$a = "Hello ";
$b = $a . "World!"; // тепер $b містить "Hello World!"
```

```
$a = "Hello ";
$a .= "World!"; // тепер $a містить "Hello World!"
?>
```

Стрічку можна розглядати і як масив символів

```
<?
$a='Hello';
$c=$a[1];
// c буде мати значення 'e'
?>
```



*Нумерація символів, як і в масивах, починається з нуля.*

Функції для роботи зі стрічками наведено у додатку 4.

### **Операції присвоєння**

Крім базової операції присвоєння (=), існують "комбіновані операції" для всіх бінарних, арифметичних і стрічкових операцій, які дають змогу використати значення у виразі, а потім установити його значення в результат цього виразу. Наприклад:

```
$a = 3;
$a += 5; // установлює в $a 8,
//ніби ми сказали: $a = $a + 5;
$b = "Hello ";
$b .= "There!"; // встановлює в $b "Hello There!";
// аналогічно $b = $b . "There!";
```

**Операції інкременту/декременту**

PHP підтримує операції pre- і post-інкременту й декременту.

**Таблиця 3**

Приклад	Назва	Ефект
++\$a	Pre-increment	Збільшує \$a на 1, потім повертає \$a.
\$a++	Post-	Повертає \$a, потім збільшує \$a на 1.
--\$a	Pre-decrement	Зменшує \$a на 1, потім повертає \$a.
\$a--	Post-	Повертає \$a, потім зменшує \$a на 1.

Наведемо приклад скрипту, який демонструє дію цих операцій:

```
<?php
echo "<h3>Postincrement</h3>";
$a = 5;
echo "Повинне бути 5: " . $a++ . "<br> ";
echo "Повинне бути 6: " . $a . "<br>";

echo "<h3>Preincrement</h3>";
$a = 5;
echo "Повинне бути 6: " . ++$a . "<br>";
echo "Повинне бути 6: " . $a . "<br>";

echo "<h3>Postdecrement</h3>";
$a = 5;
echo "Повинне бути 5: " . $a-- . "<br>";
echo "Повинне бути 4: " . $a . "<br>";

echo "<h3>Predecrement</h3>";
$a = 5;
echo "Повинне бути 4: " . --$a . "<br>";
echo "Повинне бути 4: " . $a . "<br>";
?>
```

**Передача зовнішніх змінних**

Передати зовнішні змінні в PHP-скрипт можна включивши їх в стрічку запиту при виклику скрипта в браузері, тоді вони автоматично будуть

доступні всередині скрипту через глобальний асоціативний масив \$\_GET (див нижче).

### ***Вправа 1***

Наприклад, створимо файл test\_param.php в якому напишемо

```
<html>
<head>
<title> Параметри в PHP </title>
</head>
<body>

<?php
echo $_GET['var1'];
echo "<br>";
echo "-----";
echo "<br>";
echo $_GET['var2'];
?>
</body>
</html>
```

Збережемо створений файл на сервері. У стрічці адреси браузера напишемо

http://srv/users/delf/ test\_param.php?var1=35&var2= RTFM

(Замість srv ви повинні написати ім'я свого Web-сервера, а замість users/delf - ваш шлях до файлу test\_param.php на вашому сервері)

Якщо ви все правильно зробили, то ваша сторінка в браузері виглядатиме так як на Рис. 2

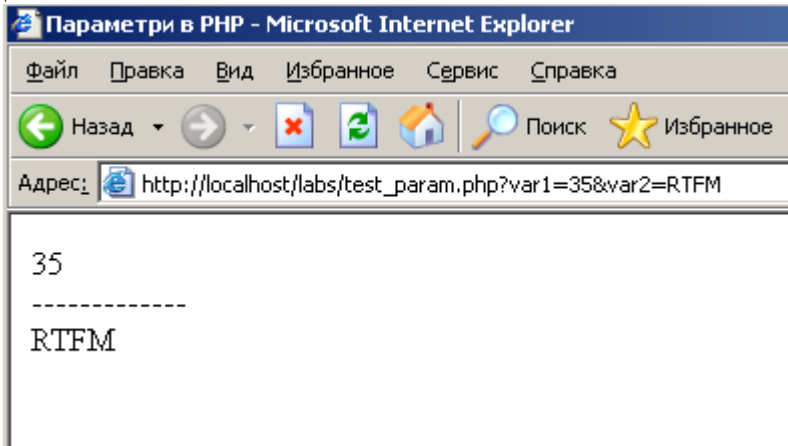


Рис. 2.

У стрічці запиту ми передали php-скрипту дві змінні var1 і var2, присвоївши їм значення відповідно 35 і "RTFM" і доступ до значень цих змінних здійснюється через глобальний масив \$\_GET з ключами var1 і var2.

Зауважимо, що в стрічці адреси, список змінних відділяється від імені файла знаком ?, імена змінних пишуться без знака \$, змінні розділяються між собою знаком & (амперсант), а стрічки пишуться без лапок.

### ***Завдання для виконання.***

1. Написати php-скрипт який видає суму двох зовнішніх параметрів `pagA` і `pagB`.  
Проекспериментуйте із значеннями параметрів що передаються, наприклад надайте `pagA` значення '3 хлопці', а `pagB` значення '6 дівчат'.
2. Яке значення матиме змінна `$a` після виконання наступного коду:  
`$a=2;`  
`$a.=( $b=('Привіт'. $a));`  
`$a+=$b+$b;`  
Свою відповідь перевірте експериментально.
3. Написати php-скрипт що обчислює значення виразів, описаних наступними формулами (математичні функції наведено в додатку 2)
  - a)  $\sqrt[8]{x^8 + 8^x}$
  - b)  $\frac{xyz - 3,3|x + \sqrt[4]{y}|}{10^7 + \sqrt{lg 4y}}$
4. Обчислити відстань між точками з координатами  $(x_1, y_1)$  і  $(x_2, y_2)$

## Лабораторна робота № 3

### Передача параметрів за допомогою форм

У попередній лабораторній роботі ми розглянули змінні в PHP, і розглядали спосіб передачі зовнішніх даних (змінних) в php-скрипт. Цей спосіб полягав у тому, що імена змінних і їхні значення вписувались безпосередньо в стрічку запиту. Як ви змогли переконались, цей спосіб є досить незручним для користувача, тому розглянемо інший спосіб передачі параметрів – за допомогою форми.

Форма – складова HTML-документу, яка містить елементи графічного інтерфейсу користувача (поля введення, списки вибору, кнопки, тощо) і призначена для відправки введених користувачем даних для опрацювання на Web-сервері.

Опис форми розпочинається тегом `<form>` і закінчується тегом `</form>`. Параметри цього тега, та теги, які описують елементи керування (поля введення, списки вибору, кнопки і т.д.), описані в додатку 1.

**Приклад:** створити форму для реєстрації користувача чату, яка містить дані (ім'я, прізвище, адресу електронної пошти, псевдоніму користувача (нік), пароль та поле для повторного вводу паролю).

Створимо файл `register.html` в якому напишемо наступний код

```
<html>
<head>
<title> Реєстрація</title>
</head>
<body>
<!--тут починається опис форми-->
<form action="adduser.php" >

Ім'я <input type="text" name="name" maxlength="20" size="25"><br>
Прізвище <input type="text" name="surname" maxlength="20" size="25"
><br>
Адреса e-mail<input type="text" name="mail" maxlength="20" size="25"
><br>
Псевдонім<input type="text" name="nik" maxlength="20" size="25" ><br>
Пароль<input type="password" name="password" maxlength="20"
size="25" ><br>
Пароль ще раз<in-
put type="password" name="confirm_pass" maxlength="20"
size="25"><br>
<input type="submit" value="Зареєструватися">
</form>
<!--кінець форми-->
</body>
</html>
```

Після відкриття даної сторінки в браузері , отримаємо форму яка зображена на Рис. 3

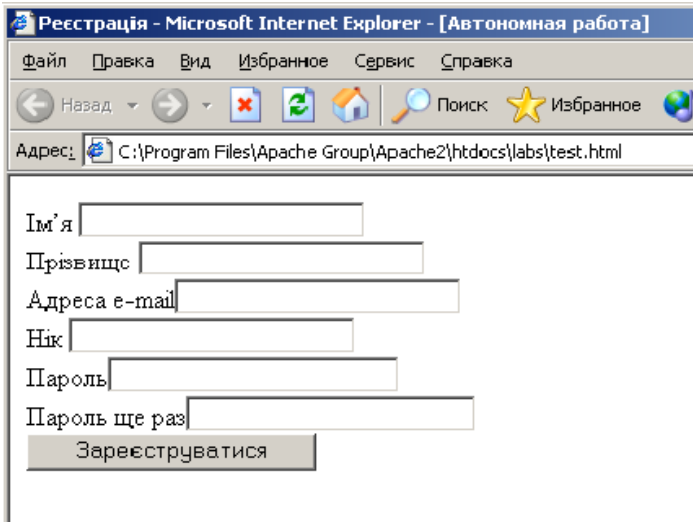


Рис. 3

Оскільки ми ще не написали скрипта `adduser.php`, який обробляє дані, які користувач ввів в форму, то при натискуванні на кнопку «зарєєструватися» нічого не відбудеться.

Нижче наведено один з варіантів скрипта `adduser.php`

```
<?
echo 'Ви ввели такі дані про себе:'.<br>;
echo "<b>Ім'я:</b>".$_GET['name'].<br>";
echo "<b>Прізвище:</b>".$_GET['surname'].<br>";
echo "<b>Адреса е-маїл</b>".$_GET['mail'].<br>";
echo "<b>Нік:</b>".$_GET['nik'].<br>";
?>
```



*Зауважимо, що в даному прикладі файл з формою `register.html` і файл із скриптом `adduser.php` мають лежати на сервері в одному каталозі*

Оскільки ми створили скрипт для обробки даних з форми, то тепер при натискуванні на кнопку «зарєєструватися» виконається скрипт з файлу `adduser.php` (той файл який вказаний в параметрі `action` форми). У цьому файлі дані з форми будуть доступними через глобальний асоціативний масив `$_GET` з ключами, рівними значенням, які вказані в атрибуті `name` при заданні тих чи інших елементів форми. Варто зауважити, що стрічка запиту, в якій

передаються зовнішні змінні, формується автоматично за тими ж правилами, що і при передачі зовнішніх змінних вручну (див. Рис. 4.)



Рис. 4

Зверніть увагу, що при передачі змінних всі поля передаються за допомогою методу GET, в тому числі і поле паролю, передаються в відкритому вигляді. Цього недоліку можна уникнути, змінивши метод передачі даних формою з GET (він використовується за замовчуванням) на POST. Тоді доступ до переданих значень форми буде здійснюватись через глобальний масив `$_POST` з відповідними ключами.

```
<form action="adduser.php" method="post" >
```

Форми можуть використовуватись також для завантаження файлів на сервер.

Наведемо найпростіший приклад форми для завантаження файлів.

```
<form enctype="multipart/form-data" action="savefile.php"
method="post">
<input type="hidden" name="MAX_FILE_SIZE" value="1000">
Send this file: <input name="userfile" type="file">
<input type="submit" value="Send File">
</form>
```

Приховане поле `MAX_FILE_SIZE` має бути описаним перед полем вводу файлу, і його значення відповідає максимальному розміру файлу, який передається. Значення вказується в байтах.

Завантажений файл буде збережено на сервері в тимчасовому каталозі, а в скрипті `savefile.php` для доступу до цього файлу будуть визначені наступні змінні:

`$_FILES['userfile']['tmp_name']` – тимчасове ім'я файлу, під яким завантажений файл був збережений на сервері.

`$_FILES['userfile']['name']` – оригінальне ім'я або шлях до файлу на системі відправника.

`$_FILES['userfile']['size']` – розмір завантаженого файлу в байтах.

`$_FILES['userfile']['type']` – mime-тип файлу, якщо браузер надав цю інформацію. Приклад: `"image/gif"`.

Зауважимо, що вся інформація про завантажений на сервер файл міститься в глобальному масиві `$_FILES`. Цей масив з'явився починаючи з PHP 4.1.0. Ключем цього масиву для доступу до завантаженого файлу на сервер є `'userfile'` – це ім'я відповідного елемента форми, за допомогою якого було зроблене це завантаження.

Наведемо приклад скрипта, який опрацьовує дані отримані з вищенаведеної форми

```
<?
```

```
echo '<b>Файл був збережений на сервері під іменем</b>'.  
$_FILES['userfile']['tmp_name'].'<br>';  
echo '<b>У вас на машині він називався : </b>'. $_FILES['userfile']  
['name'].'<br>';  
echo '<b>Розмір завантаженого файлу '. $_FILES['userfile']['size'].'<br>';  
echo '<b>Mime тип файлу: '. $_FILES['userfile']['type'].'<br>';  
>
```

Для переміщення файлів, що завантажуються на сервер з тимчасового каталога зручно використовувати функцію `move_uploaded_file`. Опис цієї функції наведено в додатку 3

Модифікуємо вищенаведений скрипт, щоб завантажені файли переміщувались з тимчасового каталогу в каталог `images`, який розміщений у тому ж каталозі, що і скрипт. Файл збережемо під таким же іменем як і на комп'ютері користувача.

```
<?  
echo '<b>Файл був збережений на сервері під іменем</b>'.  
$_FILES['userfile']['tmp_name'].'<br>';  
echo '<b>У вас на машині він називався : </b>'. $_FILES['userfile']  
['name'].'<br>';  
echo '<b>Розмір завантаженого файлу '. $_FILES['userfile']['size'].'<br>';  
echo '<b>Mime тип файлу: '. $_FILES['userfile']['type'].'<br>';  
move_uploaded_file($_FILES['userfile']['tmp_name'], "images/"  
$_FILES['userfile']['name']);  
>
```

### ***Завдання для виконання.***

1. Описати форму для завантаження на сервер електронного підручника з наступною обробкою її даних. Для цього використати наступні поля: автор книги, назва книги, видавництво, кількість сторінок, формат файлу, шлях до файлу.
2. Описати форму для реєстрації користувача з наступними даними: псевдонім, пароль, підтвердження пароля, ім'я, прізвище, адреса e-mail, країна (вибір із списку), область, місто(село), вулиця.
3. Описати форму для завантаження на сервер статей. Для цього використати наступні поля: автор статті, опис статті, журнал, у якому була надрукована стаття, кількість сторінок, рік видання, формат файлу, шлях до файлу.



# Лабораторна робота №4

## Структури управління в PHP

### *Теоретичні відомості*

Будь-який PHP-скрипт складається з операторів. Ними можуть бути оператори присвоєння, виклик функції, цикл, умовний оператор або навіть оператор, що нічого не робить (порожній оператор). Оператор завершується крапкою з комою. Крім того, оператори можна групувати за допомогою фігурних дужок `{}`. Група операторів сама також є оператором. Різні типи операторів розглядаються в даній лабораторній.

#### **if**

Конструкція `if` є однією із ключових у багатьох мовах, у тому числі й в PHP. Вона дає змогу виконувати фрагменти коду при виконанні умови. У PHP структура `if` є аналогічною структурі оператора умови мови C:

```
if (expr) statement
```

`expr` обчислюється як булеве значення. Якщо `expr` – TRUE, PHP виконає `statement`, а якщо – FALSE – оператор ігнорується.

Наступний приклад виведе “a is bigger than b”, якщо \$a більше \$b:

```
if ($a > $b)  
echo "a is bigger than b";
```

Часто необхідно виконати не один, а декілька операторів. Зрозуміло, немає необхідності створювати для кожного оператора конструкцію `if`. Замість цього ви можете згрупувати кілька операторів у блок. Наприклад, цей код виведе “a is bigger than b”, якщо \$a більше \$b, а потім присвоїть значення змінної \$a змінній \$b:

```
if ($a > $b) {  
    print "a is bigger than b";  
    $b = $a;  
}
```

Оператори `if` можуть вкладатися один в одного, що дає змогу їх комбінувати при виконанні різних частин програми.

#### **else**

Часто потрібно виконати оператор, якщо дотримано яку-небудь умову, і інший оператор – якщо умова не дотримана. Для цього призначений оператор

else, який розширює оператор if і виконує “свої” оператори, якщо вираз в операторі if обчислюється в FALSE. Наприклад, наступний код виведе “a is bigger than b”, якщо \$a виявиться більше \$b, і “a is NOT bigger than b” – в іншому випадку:

```
if ($a > $b) {
    print "a is bigger than b";
}
else {
    print "a is NOT bigger than b";
}
```

Оператор else виконується тільки в тому випадку, якщо вираз if обчислюється в FALSE

### **switch**

Ще одна конструкція, що дозволяє перевіряти умову і виконувати в залежності від цього різні дії, – це switch. На українську мову назва даного оператора перекладається як «перемикач». І зміст у нього такий же. У залежності від того, яке значення має змінна, він виконує ту чи іншу дію. switch дуже схожий на оператор if...elseif...else або набір операторів if. Структуру switch можна записати наступним чином:

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

На відміну від if, тут значення виразу не зводиться до логічного типу, а просто порівнюється зі значеннями, перерахованими після ключових слів case (значення1, значення 2 і т.д.). Якщо значення виразу співпадає з якимось варіантом, то виконується відповідний блок\_дій – від двокрапки після значення, що співпало, до кінця switch або до першого оператора break, якщо такий знайдеться. Якщо значення виразу не співпало з жодним з варіантів, то виконуються дії по замовчуванні (блок\_дій\_по\_замовчуванні), що знаходяться після ключового слова default. Вираз в switch обчислюється тільки один раз, а в операторі elseif – щоразу, тому, якщо вираз достатньо складний, то switch працює швидше.

```

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

```

Якщо в цьому прикладі опустити оператор `break`, наприклад, у `case "Петро":`, то, якщо змінна виявиться рівною рядковій `"Петро"`, після виведення на екран повідомлення `"Привіт, Петя!"` програма піде далі і виведе також повідомлення `"Привіт, Сеня!"` і тільки потім, зустрівши `break`, продовжить своє виконання за межами `switch`.

## while

Цикли `while` – це цикл з передумовою. Синтаксис оператора `while` має вигляд:

```
while (expr) statement
```

Оператор `while` використовується для неодноразового виконання вкладеного оператора, поки вираз `expr` має логічне значення істини. Значення виразу перевіряється щоразу на початку циклу, тому, якщо це значення змінилося при виконанні вкладеного оператора, виконання не зупиниться до кінця даної ітерації. Якщо вираз `expr` має логічне значення `FALSE` на початку циклу, вкладений оператор не виконується жодного разу.

```
/* приклад 1 */
```

```

$i = 1;
while ($i <= 10) {
    print $i++; /* буде виведено значення
                $i до інкременту
                (пост-інкремент) */
}

```

## for

Цикл з параметром for дещо складніший.

Синтаксис циклу for має вигляд:

```
for (expr1; expr2; expr3) statement
```

Перший вираз (*expr1*) обчислюється один раз і безумовно на початку виконання циклу.

Вираз *expr2* обчислюється на початку кожної ітерації. Якщо він має значення логічної істини, цикл триває й виконується вкладений оператор. Якщо він отримує значення логічної хибності, виконання циклу припиняється.

Наприкінці кожної ітерації обчислюється (виконується) вираз *expr3*.

Кожен з виразів може бути порожнім. Порожній вираз *expr2* означає, що цикл повинен виконуватися безкінченно (PHP неявно припускає, що ця умова TRUE).

Розглянемо наступні приклади. Всі вони виводять числа від 1 до 10:

```
/* приклад 1 */
```

```
for ($i = 1; $i <= 10; $i++) {  
    print $i;  
}
```

```
/* приклад 2 */
```

```
for ($i = 1;; $i++) {  
    if ($i > 10) {  
        break;  
    }  
    print $i;  
}
```

```
/* приклад 3 */
```

```
$i = 1;  
for (;;) {  
    if ($i > 10) {  
        break;  
    }  
    print $i;  
    $i++;  
}
```

```
/* приклад 4 */
```

```
for ($i = 1; $i <= 10; print $i, $i++);
```

## Функції, визначені користувачем (користувацькі)

Функція може бути визначена з використанням такого синтаксису:

```
function foo ($arg_1, $arg_2, ..., $arg_n)
{
    echo "Приклад \n";
    return $retval;
}
```

Всередині функції можна використовувати будь-які оператори мови або інші функції.

Інформація може передаватися у функцію через список аргументів, що є списком розділених комами змінних і/або констант.

Значення з функцій повертаються за допомогою необов'язкового оператора return. Функція може повернути змінну будь-якого типу, у тому числі список і об'єкт. Цей оператор негайно зупиняє виконання функції та передає керування назад на рядок, з якого функція була викликана.

### *Приклад*

Дано число X надрукувати X! (факторіал X).

Для введення значення X опишемо просту форму

```
<html>
<head>
<title> Введення змінної</title>
</head>
<body>
<div align="left">
введіть значення X
<!--тут починається опис форми-->
<form action="factorial.php" method="get" >

X= <input type="text" name="X" maxlength="20" size="25"><br>
<input type="submit" value="Обчислити">
</form>
<!--кінець форми-->
</div>
</body>
</html>
```

Збережемо її в файлі input.html

Власне скрипт опишемо в файлі factorial.php. Нижче наведено один з можливих розв'язків даної задачі.

```
<?
$X=$_GET['X']
if (($X==0)||($X==1))
```

```
{
    $f=1;

}else{
    $f=1;
    for($i=1;$i<=$X;$i++)
    {$f=$f*$i;
    }
};
echo $X.'!='.$f;
?>
```

### *Завдання для виконання.*

1. Задано ціле невід'ємне число  $n$ . Підрахувати в ньому кількість цифр.
2. Для цілого числа  $k$  вивести фразу “На парі викладач поставив студентам  $k$  двійок”, узгодивши закінчення слова “двійка” з числом  $k$

# Лабораторна робота №5

## Робота з файлами

### *Теоретичні відомості*

Для роботи із файловою системою Web-сервера в мові реалізовано наступні процедури та функції:

**file\_exists(string назва\_файлу)** – повертає значення істини, якщо файл із заданим ім'ям існує.

**is\_file (string назва\_файлу)** – перевіряє існування вказаного файлу, а також можливість виконання з ним операцій читання-запису.

**fopen (string назва\_файлу, string режим доступу)** – відкриває вказаний файл у вказаному режимі, “r” – читання, “w” – запису, “a” – додавання даних, а також повертає цілочисельне значення (ідентифікатор відкритого файлу). Зазначений файл може бути розміщений на комп'ютері, на якому виконується Web-сервер, на іншому NNTP чи FTP-сервері.

**fclose (int ідентифікатор відкритого файлу)** – закриває файл із вказаним ідентифікатором.

**fread(int ідентифікатор, int кількість\_байт)** – зчитує із файлу із заданим ідентифікатором вказану кількість байт.

**fgets(int ідентифікатор, int довжина)** – зчитує із файлу стрічку вказаної довжини.

**file(назва\_файлу)** – завантажує файл в індексований масив, кожен елемент якого відповідає одній стрічці файлу.

**fwrite(int ідентифікатор, string стрічка)** – здійснює запис значення рядкової змінної у файл.

Детальніше ці функції описані у додатку 3.

### *Вправа1*

Написати програму – календар подій. Програма повинна зберігати в файлі визначні події і дати їх настання, введені за допомогою форми. При завантаженні сторінки виводити п'ять найближчих до поточної дати подій.

Дані будемо зберігати у текстовому файлі такого формату:

Дата1

Подія1

Дата2

Подія2

Дата3

...

Опишемо форму для додавання події.

```

<html>
<head>
<title>Портал: Календар - нова подія</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-
1251">
</head>
<body>
<center>
  <form action="addevent.php" method="post">
    <table border="0" width="60%">
      <tr>
        <td width="15%" > *Число </td>
        <td width="75%"> <input type="text" name="day" size="2"
maxlength="2">
      </td>
    </tr>
    <tr>
      <td> *Місяць </td>
      <td>
        <select name="month" >
          <option value="01"> Січень</option>
          <option value="02"> Лютий</option>
          <option value="03"> Березень</option>
          <option value="04"> Квітень</option>
          <option value="05"> Травень</option>
          <option value="06"> Червень</option>
          <option value="07"> Липень</option>
          <option value="08"> Серпень</option>
          <option value="09"> Вересень</option>
          <option value="10"> Жовтень</option>
          <option value="11"> Листопад</option>
          <option value="12"> Грудень</option>
        </select>
      </td>
    </tr>
    <tr>
      <td> *Рік </td>
      <td> <input type="text" name="year" size="3" maxlength="4"> </td>
    </tr>
    <tr>
      <td> *Опис події </td>
      <td> <input type="text" name="fulltext" maxlength="255" size="100" >
<div align="right">* обов'язкові поля</div>
    </td>
    </tr>
  </table>
  <input type="submit" value="Відправити">
</form>
</center>

```



```
</body>
</html>
```

Збережемо цю форму у файлі *addevent.html*.

Власне скрипт, який буде обробляти дані з форми та дописувати подію в файл, опишемо в файлі *addevent.php*:

```
<html>
<head>
<title>Реєстрація події</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-
1251">
</head>

<body>
<?
$file_name="events.dat";
//перевіряємо, чи існує файл
if (file_exists($file_name)) {
    //якщо так, то відкриваємо його в режимі доповнення
    $f=fopen($file_name,'a');
}
else {
    //якщо ні, то відкриваємо його в режимі запису (створюємо новий файл)
    $f=fopen($file_name,'w');
}
// перевіряємо чи коректно введено дату
$day = $_POST['day'];
$month = $_POST[' month '];
$year = $_POST['year'];
$fulltext = $_POST['fulltext'];

if (($day=="") || ($month=="") || ($year=="")) {
    echo "Помилка реєстрації події- не правильно введено дату<br>";
    echo "<a href='addevent.html'>Спробувати ще раз...</a>";
    exit;
}
// перевіряємо чи коректно введено текст події
if ($fulltext=="") {
    echo "Помилка реєстрації події- не правильно введено опис події<br>";
    echo "<a href='addevent.php'>Спробувати ще раз...</a>";
    exit;
}
//формуємо дату
$date=mktime(23,59,59,$month,$day,$year);
// записуємо дату у файл
fputs($f,$date."\n\r");
fputs($f,$fulltext."\n\r");
```

```

echo 'Подію успішно додано в календар';
fclose($f);
?>
</body>
</html>

```

А тепер напишемо скрипт, який буде виконувати основну функцію програми: видавати п'ять найближчих до поточної дати подій. Збережемо його у файлі *calendar.php*. Нижче наведено один з можливих варіантів такого скрипта.

```

<html>
<head>
<title>Методичний портал: календар</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
</head>
<body>
<?
$file_name="events.dat";
//Перевіряємо чи існує файл календаря
if (!file_exists($file_name)) {
    echo "Помилка: не існує файлу календаря";
    exit;
}
//відкриваємо файл в режимі читання
$f=fopen($file_name,'r');
//зчитуєм вміст файлу в два масиви: дати і події
$i=0;
while (!feof($f)) {
    $buf=fgets($f, 1000000);
    if (trim($buf)!="")
    {
        $date[$i] = $buf;
    }
    $text[$i] = fgets($f, 1000000);
    $i=$i+1;
}
//закриваєм файл
fclose ($f);

//дата з якою будем порівнювати - поточна дата
$my_date=time();
//масив в який будемо записувати індекси найближчих подій
$near=array();
for ($k=0;$k<5;$k++)
{
    $min=11111111111111111111111111111111;
    //echo $min;

```

```

for ($i=0;($i<count($date));$i++)
{

    if (($date[$i]>=$my_date) and (abs($date[$i]-$my_date)<=$min))
    {
        //якщо події нема в масиві то додаєм її
        if (! in_array($i,$near))
        {
            $min=$date[$i]-$my_date;
            $near[$k]=$i;
        }
    }
}

//видаєм повідомлення якщо масив порожній
if (count($near)==0) {
echo 'Подій після ' .date('d-m-Y',time()).' не введено!';
}
else
{
    // інакше виводимо знайдені події
    echo '<b>Скоро відбудеться:</b> <br>';
}

    for ($i=0;(($i<5) and ($i<count($near)));$i++)
    {
        echo date('d-m-Y',$date[$near[$i]]).' '.$text[$near[$i]].<br>';
    }
?>
</body>
</html>

```

## ***Завдання для виконання.***

1. Розробити сценарій для виконання авторизації користувача. Дані про користувачів зберігати у файлі, що має вигляд:  
назва\_користувача\_1:пароль\_користувача\_1  
  
назва\_користувача\_2:пароль\_користувача\_2  
.....  
назва\_користувача\_N:пароль\_користувача\_N  
Рекомендується використати функцію `explode` для роботи зі стрічками (див. додаток 4).
2. Написати скрипт, який виводить результати голосування на поставлене запитання, де можливими відповідями є: так, ні, утримався.
3. Написати скрипт – лічильник, для підрахунку кількості переглядів сайту. Скрипт має виводити два числа кількість переглядів сьогодні і кількість переглядів взагалі. Функції для роботи із датами наведено у додатку 5.
4. Написати скрипт для роботи електронної дошки оголошень – форуму. Форум складається з трьох сторінок :
  - сторонка з формою для додавання повідомлень в форум.
  - сторонка з скриптом, що вносить дані з форми в файл.
  - скрипт що виводить повідомлення форуму з файлу.
5. Написати програму для вводу та виводу новин. Новини, як і форум, можуть складатися з трьох сторінок:
  - форми вводу,
  - скрипта для запису в файл,
  - скрипта для виводу кількох останніх новин;

## Лабораторна робота № 6

### Робота із СУБД MySQL

Великою перевагою використання мови сценаріїв PHP, є можливість динамічного генерування вмісту. Ми вже використовували вхідні дані користувача, передані в гіперпосиланнях та отримані з файлів. Тепер ми навчимося використовувати реляційні бази даних. База даних є сукупністю даних, організована з метою їх швидкого пошуку і опрацювання. У системі управління реляційної бази даних дані мають вигляд таблиць, стовпці яких називаються полями або атрибутами, а стрічки – записами. З метою пришвидшення пошуку у реляційних базах даних використовують індекси. Індекс – це відсортований список значень деяких полів. Крім значень індекс містить вказівки на адресу значення в таблиці. Інденси, як правило, зберігаються у файлах, окремо від даних таблиць. Індекс називається унікальним, якщо кожне значення його індексного поля є єдиним.

Найчастіше індекси використовуються у вигляді первинного ключа (primary key), який застосовується для унікальної ідентифікації окремого запису таблиці. Жодні два записи таблиці не можуть мати однакових значень первинного ключа. Звичайно первинний ключ потрібен в реляційній базі даних, оскільки дає змогу обробляти дані однозначним способом. Первинні ключі використовуються для виконання цієї вимоги. Первинні ключі можуть складатися з одного або більшої кількості полів таблиці. Первинний ключ вибирають, знайшовши серед даних ті, які унікально визначають запис. Якщо таких записів немає, то їх створюють штучно.

Універсальним інструментарієм для роботи з реляційними базами даних є мова структурованих запитів SQL.

Команди SQL поділяють на дві категорії:

- команди визначення структури БД;
- команди обробки даних.

Для визначення структури баз даних сервера MySQL використовують готові програми оболонки такі як MySQLAdmin або phpMyAdmin.

Детальніше команди мови SQL можна знайти у додатку 6.

Розглянемо детальніше основні конструкції для формування SQL-запитів для обробки даних:

- 1) **insert into назва\_таблиці (поля\_таблиці1, поля\_таблиці2 ...) values('значення1', 'значення2'...)** – запит додає до таблиці **назва\_таблиці** запис, поля якого **поля\_таблиці1**, набувають значень **значення1**.
- 2) **delete from назва\_таблиці where вираз** – Знищує із таблиці записи, для яких є істинним вираз. Параметр вираз є логічним виразом. Наприклад: (id<10) and (age=25)

3) **select \* from where вираз [order by назва\_поля [desc]]** – проводить пошук записів, які відповідають заданому виразу. Якщо отриманих записів декілька, то при заданій конструкції **order by** вони будуть відсортовані за тим полем **назва\_поля** (якщо задано слово **desc**, то сортування буде проведено у зворотному порядку). Символ **\*** вказує на те, що слід отримати дані усіх полів, записи, яких відповідають умові. Тобто замість символу “\*” можна вказувати потрібні поля.

4) **update назва\_таблиці SET (назва\_поля1='значення1', назва\_поля2='значення2', ...) where вираз** – У таблиці для записів, які відповідають умові зазначені поля набувають відповідних значень.

Розглянемо функції мови PHP для роботи із СУБД MySQL.

Перш ніж працювати з базою даних, необхідно встановити з нею мережеве з’єднання, а також провести авторизацію користувача. Для цього використовується функція `mysql_connect()`.

**int mysql\_connect([сервер] [користувач] [пароль]).**

З’єднання з MySQL-сервером буде автоматично закрито після закінчення роботи сценарію, або ж при виклику функції `mysql_close()`.

З метою визначення бази даних, з таблицями якої слід працювати використовують функцію **mysql\_select\_db (назва\_БД).**

Для формування запитів до бази даних застосовують функцію **mysql\_query()**, яка повертає ідентифікатор результуючого набору даних, проте результат відразу не пересилається клієнту. Для того, щоб працювати з ним надалі, і використовується цей ідентифікатор. Існує дуже багато функцій, які приймають його як параметр і повертають ті або інші дані.

Синтаксис функції такий: **int mysql\_query(string \$query [,int \$link\_identifier]).**

Параметр `$query` є запитом-стрічкою, який повинен бути описаний за правилами мови SQL. Необов’язкова змінна `$link_identifier`, яка містить значення ідентифікатора з’єднання, використовується для відновлення втраченого зв’язку із сервером MySQL.

Після виконання запиту і отримання його ідентифікатора потрібно отримати власне дані – результат. Для цього використовується функція **array mysql\_fetch\_row(int \$result)**, яка повертає масив-список із значеннями полів чергового рядка результату `$result`. Якщо вказівник поточної позиції результату був встановлений після останнього запису, то функція повертає значення `false`. Поточна позиція переміщується до наступного запису, тому кожен наступний виклик `mysql_fetch_row()` поверне наступний рядок результату.

Функція **mysql\_fetch\_array(int \$result)** повертає черговий рядок результату у вигляді асоціативного масиву, де кожному полю відповідає елемент з індексом, що співпадає з ім’ям поля.

Розглянемо приклад. Нехай таблиця 10A бази даних pupils має наступний вигляд:

ID	surname	name	bal

Потрібно вивести у таблицю прізвища та імена учнів (поля surname та name) у алфавітному порядку, середній бал яких більший за 4,5 (поле бал).

```
<?php
$db=mysql_connect("localhost","root","");
mysql_select_db("pupils", $db);
$sql="SELECT surname,name,bal FROM 10A WHERE bal>4.5
ORDER BY surname";
$result=mysql_query($sql, $db);
echo "<table align='center' width='100%' border='1'>";
    echo "<tr>";
        echo "<td>";
            echo "№п/п";
        echo "</td>";
        echo "<td>";
            echo "Прізвище";
        echo "</td>";
        echo "<td>";
            echo "Ім'я";
        echo "</td>";
        echo "<td>";
            echo "Середній бал";
        echo "</td>";
    echo "</tr>";
    $i=1;
    while ($row=mysql_fetch_array($result)){
        echo "<tr>";
            echo "<td>";
                echo $i;
            echo "</td>";
            echo "<td>";
                echo $row["surname"];
            echo "</td>";
            echo "<td>";
                echo $row["name"];
            echo "</td>";
            echo "<td>";
                echo $row["bal"];
            echo "</td>";
        echo "</tr>";
        $i=$i+1;
    }
echo "</table>";
mysql_close();
?>
```

Для того, щоб у кожному сценарії не описувати параметри для з'єднання із сервером MySQL (при перенесенні сайту на інший сервер, пароль і ім'я користувача можуть змінитись і доведеться редагувати кожну сторінку), винесемо їх у окремий файл `mysql.inc`, який будемо включати в скрипт командою `include` (див. приклад скрипта для реєстрації користувачів).

```
<?
//приклад файлу mysql.inc
define("mysql_host",'localhost');
define("mysql_user",'root');
define("mysql_password","");
?>
```

Розглянемо як приклад складову частину нашого проекту – сценарії реєстрації користувачів порталу.

Для розв'язання задачі необхідними є:

1. База даних із таблицею, у якій зберігатимуться дані про користувачів.
2. Форма введення даних для реєстрації нового користувача.
3. Сценарій реєстрації нових користувачів.
4. Форма введення даних для авторизації існуючого користувача.
5. Сценарій авторизації існуючих користувачів.

Використовуючи Web-інструментарій PhpMyAdmin у базі даних `portal` створимо таблицю `users` із такими полями:

Назва поля	Опис поля	Тип	Довжина	Додаткові значення
id	ідентифікатор_користувача	Ціле число	10	первинний ключ, автоматичний інкремент
surname	Прізвище	Стрічка	35	
name	Ім'я	Стрічка	35	
nik	Псевдо	Стрічка	35	
password	Пароль	Стрічка	35	
mail	Адреса електронної пошти	Стрічка	35	
country	Країна	Стрічка	35	
city	Місто	Стрічка	35	
address	Адреса	Стрічка	255	

Завантажимо Web-сторінку з адресою `http://web-server/phpadmin/`. У лівому фреймі оберемо базу даних із якою будемо працювати – `portal`. У полі **Назва** введемо назву нової таблиці – `users` та кількість полів у ній (див. рис. 5).



## БД portal на localhost

[Структура](#)[SQL](#)[Експорт](#)[Шукати](#)[Запит згідно прикладу](#)

В БД не виявлено таблиць.

- Створити нову таблицю в БД portal :

Назва :

Поля :

Рис.5

Введемо у завантажену форму назви, типи полів (див. рис. 6):

### БД portal - таблиця users на localhost

Поле	Тип [Документація]	Довжини/Значення <sup>m</sup>	Атрибути	Нуль	По замовчуванню <sup>m</sup>	Додатково
id	INT	10		not null		auto_increment
surname	VARCHAR	35		not null		
name	VARCHAR	35		not null		
nik	VARCHAR	35		not null		
password	VARCHAR	35		not null		
mail	VARCHAR	35		not null		
country	VARCHAR	35		not null		
city	VARCHAR	35		not null		
address	VARCHAR	255		not null		

Коментар до таблиці :

Тип таблиці :

Рис.6

Як первинний ключ задамо поле ідентифікатор користувача (id). З метою забезпечення унікальності цього поля задамо його автоматичне збільшення при додаванні нового запису (додатково – autoincrement).

Створимо файл adduser.html, у якому опишемо форму реєстрації нового користувача:

```
<html>
<head>
<title>Методичний портал - Реєстрація</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-
1251" />
</head>
<body>
<form action="adduser.php" method="post">
<table align="center">
```

```

<tr>
  <td>
    *Псевдонім
  </td>
  <td>
    <input name="nik" type="text" size="35" maxlength="35">
  </td>
</tr>
<tr>
  <td>
    *Пароль
  </td>
  <td>
    input name="password" type="password" size="35"
maxlength="35">
  </td>
</tr>
<tr>
  <td>
    *Пароль ще раз
  </td>
  <td>
    <input name="confirm_password" type="password"
size="35" maxlength="35" >
  </td>
</tr>
<tr>
  <td>
    *Прізвище
  </td>
  <td>
    <input name="surname" type="text" size="35"
maxlength="35">
  </td>
</tr>
<tr>
  <td>
    *Ім'я
  </td>
  <td>
    <input name="name" type="text" size="35"
maxlength="35">
  </td>
</tr>
<tr>
  <td>
    *Ваш e-mail
  </td>
  <td>

```

```

                                <input name="mail" type="text" size="35"
maxlength="35">
                                </td>
                                </tr>
                                <tr>
                                <td>
                                *Країна
                                </td>
                                <td>
                                <select name="country">
                                <option value="other">Інша</option>
                                <option value="ua">Україна</option>
                                <option value="ru">Росія</option>
                                </select>
                                </td>
                                </tr>
                                <tr>
                                <td>
                                *Місто
                                </td>
                                <td>
                                <input name="city" type="text" size="35" maxlength="35">
                                </td>
                                </tr>
                                <tr>
                                <td>
                                Адреса
                                </td>
                                <td>
                                <input name="address" type="text" size="35"
maxlength="35">
                                </td>
                                </tr>
                                </table>
                                <div align="right">* - обов'язкові поля</div>
                                <center> <input type="submit" value="Відправити" > </center>
                                </form>
                                </body>
                                </html>

```

Опишемо php-сценарій adduser.php для проведення реєстрації нових користувачів, дані про яких вводяться за допомогою форми у файлі add-user.html.

Алгоритм роботи сценарію:

1. Перевірити коректність вводу даних користувача:
  - a. значення змінних, введених за допомогою форми не повинні бути порожніми;
  - b. існування зареєстрованого користувача з таким псевдонімом.

С. адреса електронної пошти повинна містити символ «@» .

2. У випадку не виконання п.1 вивести повідомлення про причину помилки та припинити виконання сценарію.
3. Додати дані про користувача до таблиці users.
4. Вивести повідомлення про успішність реєстрації користувача, автоматично додати його до авторизованих користувачів сайту.

<?

```
//Виводимо файл параметрів з'єднання із сервером MySQL
include 'mysql.inc';
echo "<center>";
if (!mysql_connect(mysql_host,mysql_user,mysql_password)){
echo "Не можу з'єднатися із сервером MySQL";
exit;
};
if (!mysql_select_db("portal")){
echo "Не можу з'єднатися із БД";
exit;
}
//Перевіряємо коректність вводу псевдоніму
$nik=$_POST['nik'];
if ($nik==""){
echo "Помилка реєстрації - не введено псевдонім<br>";
echo "<a href='adduser.html'>Спробувати ще раз...</a>";
exit;
}
//Перевіряємо, можливо, такий псевдонім вже зареєстровано
$sql="SELECT * from users WHERE nik='$nik'";
$result=mysql_query($sql);
if (mysql_num_rows($result)>0){
echo "Помилка реєстрації - псевдонім вже зареєстровано<br>";
echo "<a href='adduser.html'>Спробувати ще раз...</a>";
exit;
}
//Перевіряємо чи коректно введено пароль
$password=$_POST['password'];
if ($password==""){
echo "Помилка реєстрації - не введено пароль<br>";
echo "<a href='adduser.html'>Спробувати ще раз...</a>";
exit;
}
//Перевіряємо чи співпадають паролі
if ($password!= $_POST['confirm_password ']){
echo "Помилка реєстрації - паролі не співпадають<br>";
echo "<a href='adduser.html'>Спробувати ще раз...</a>";
exit;
}
//Перевіряємо чи коректно ім'я
$name =$_POST['name'];
```

```

if ($name==""){
    echo "Помилка реєстрації - не введено ім'я<br>";
    echo "<a href='adduser.html'>Спробувати ще раз...</a>";
    exit;
}
//Перевіряємо чи коректно введено прізвище
$surname=$_POST['surname'];
if ($surname==""){
    echo "Помилка реєстрації - не введено прізвище<br>";
    echo "<a href='adduser.html'>Спробувати ще раз...</a>";
    exit;
}

//Перевіряємо коректність вводу e-mail
$mail=$_POST['mail'];
if (!strpos($mail,'@')){
    echo "Помилка реєстрації - не правильно введено e-mail<br>";
    echo "<a href='adduser.html'>Спробувати ще раз...</a>";
    exit;
}
//Перевіряємо коректність вводу адреси
$country=$_POST['country'];
$city=$_POST['city'];
if (($country=="") || ($city=="")){
    echo "Помилка реєстрації - не введено адресу<br>";
    echo "<a href='adduser.html'>Спробувати ще раз...</a>";
    exit;
}
$address=$_POST['address'];
//Додаємо дані про користувача у базу даних
if (isset($nik)){
    $sql="INSERT into users (nik, password, name, surname, mail, country,
city, address) VALUES ('$nik', '$password' , '$name', '$surname', '$mail', '$country',
'$city', '$address')";
    mysql_query($sql);
}
//Створюємо сесію користувача
session_start();
$_SESSION[' nik ']=$nik;
$_SESSION[' password ']=$password;
echo "Користувача $nik зареєстровано!";
//У фреймі leftFrame міняємо гіперпосилання "Реєстрація" на "Вихід"
echo '<script>';
echo 'parent.leftFrame.regist.href="exit.php"';
echo 'parent.leftFrame.regist.innerText="Вихід"';
echo '</script>';

}
echo "</center>";
?>

```

Слід звернути увагу на формування стрічки запиту \$sql: значення полів, які слід опрацювати потрібно брати в лапки. Оскільки запит сформовано у подвійних лапках, то значення полів введено в одинарних лапках. Нехай стрічку “користувача \$nik зареєстровано” взято у подвійні лапки, тоді під час виводу стрічки за допомогою echo замість змінної \$nik буде підставлено її значення. Якщо ж стрічку взято у одинарні лапки, то під час виводу значення змінної підставлено не буде.

У файлі register.html опишемо форму введення даних для авторизації існуючого користувача:

```
<html>
<head>
<title>Документ без имени</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-
1251">
</head>
<body>
<center>
<form action="register.php" method="post">
  Псевдо<input name="nik" type="text" size="20"> <br>
  Пароль<input name="password" type="password" size="20"> <br>
  <input type="submit" value="Гаразд">
</form>
</center>
<div align="right"><a href="adduser.html">Реєстрація нових користу-
вачів...</a></div>
</body>
</html>
```

Опишемо php-сценарій register.php для проведення входу на портал зареєстрованих користувачів, дані про яких вводяться за допомогою форми файлу register.html.

Алгоритм роботи сценарію:

1. Перевірити відповідність вводу даних користувача із тими, які знаходяться у базі даних
  - a. псевдонім;
  - b. пароль.
2. У випадку не виконання п.1 вивести повідомлення про причину помилки та припинити виконання сценарію.
3. У випадку виконання п.1 створити сесію для користувача.

```
<?
include 'mysql.inc';
echo "<center>";
if (! mysql_connect(mysql_host,mysql_user,mysql_password)){
echo "Не могу з'єднатися із сервером MySQL";
exit;
};
```

```

if (!mysql_select_db("portal")){
echo "Не можу з'єднатися із БД";
exit;
}
//Отримуємо дані про користувача з бази даних
$nik=$_POST['nik'];
$password=$_POST['password'];

$sql="SELECT password from users WHERE nik='$nik'";
$result=mysql_query($sql);
if (!$result){
    echo "Помилка реєстрації - не вірно введено псевдонім або
пароль<br>";
    echo "<a href='register.html'>Спробувати ще раз...</a>";
    exit;
}
if (mysql_num_rows($result)==0){
    echo "Помилка реєстрації - не вірно введено псевдонім або
пароль<br>";
    echo "<a href='register.html'>Спробувати ще раз...</a>";
    exit;
}
else {
    $res=mysql_fetch_array($result);
    if ($res['password']==$password){
        session_start();
        $_SESSION[' nik ']=$nik;
        $_SESSION[' password ']=$password;
        echo '<script>';
        echo 'parent.leftFrame.regist.href="exit.php";';
        echo 'parent.leftFrame.regist.innerText="Вихід";';
        echo '</script>';
        echo 'Дякуємо за авторизацію';
    }
    else {
        echo "Помилка авторизації - не вірно введено псевдонім або
пароль<br>";
        echo "<a href='register.html'>Спробувати ще раз...</a>";
        exit;
    }
}
echo "</center>";
?>

```

З метою збереження даних про авторизованих користувачів використовується механізм так званих сесій роботи клієнта, про який детальніше можна прочитати у додатку 7.

## Завдання для виконання.

1. Написати php-сценарій `userinfo.php` для виводу із таблиці `users` даних про користувача (ім'я, прізвище, адреса e-mail, адреса) за його відомим псевдонімом. Псевдонім передається за допомогою форми із елемента – випадаючий список, елементами якого є всі існуючі псевдоніми з таблиці `users`.
2. Використовуючи Web-інструментарій `PhpMyAdmin`, у базі даних `portal` створити таблицю `library` із такими полями:

Назва поля	Опис поля	Тип	Додаткові значення
Id	ідентифікатор_книги	Ціле число	Первинний ключ, автоматичний інкремент
Title	Назва книги	стрічка	
Autor	Автор	стрічка	
publishing	Видавництво	стрічка	
Year	Рік видання	ціле число	
n_pages	кількість сторінок	ціле число	
Format	Формат файлу	стрічка	
path	Адреса знаходження файлу на диску сервера	стрічка	

3. Написати php-сценарій `addbook.php` для додавання книг у таблицю `library` електронної бібліотеки, із використанням форми, створеної у вправі 1 лабораторної роботи №3.

Алгоритм роботи сценарію:

- a) Для перевірки того, чи на дану сторінку увійшов авторизований користувач, на початку роботи сценарію за допомогою функції `include(назва_файлу)` слід приєднати модуль перевірки авторизації `header_login.inc`.
- b) Перевірити коректність вводу даних користувача – усі поля повинні бути заповнені.
- c) У випадку не виконання п.1 вивести повідомлення про причину помилки та припинити виконання сценарію.
- d) Скопіювати файл, завантажений із комп'ютера клієнта у папку `library`.
- e) Додати дані про книгу до таблиці `library`.



4. Написати php-сценарій library.php для виводу книг, дані про які знаходяться у таблиці library.  
Алгоритм роботи сценарію:
- Приєднати модуль перевірки авторизації користувача header\_login.inc.
  - На Web-сторінці головного фрейму вивести дані про книги та посилання на відповідні їм файли.
  - Вивести посилання на сторінку додавання нових ресурсів.
5. Використовуючи Web-інструментарій PhpMyAdmin, у базі даних portal створити таблицю articles для збереження наукових публікацій користувачів порталу із такими полями:

Назва поля	Опис поля	Тип	Додаткові значення
id	Ідентифікатор_статті	ціле число	Первинний ключ, автоматичний інкремент
title	Назва статті	Стрічка	
autor	Ідентифікатор_користувача із таблиці users	Число	
description	Опис статті	Текст	
magazine	Журнал, у якому була надрукована стаття	Стрічка	
year	Рік видання	ціле число	
n_pages	кількість сторінок	ціле число	
format	Формат файлу	Стрічка	
path	Адреса знаходження файлу на диску сервера	Стрічка	

6. Написати php-сценарій addarticle.php для додавання статей у таблицю articles електронної бібліотеки, із використанням форми, створеної у завданні 3 лабораторної роботи №3.  
Алгоритм роботи сценарію:
- Приєднати модуль перевірки реєстрації користувача header\_login.inc .
  - Перевірити коректність вводу даних користувача – усі поля повинні бути заповнені.
  - У випадку не виконання п.1 вивести повідомлення про причину помилки та припинити виконання сценарію.

- d) Скопіювати файл завантажений із комп'ютера клієнта у папку articles.
  - e) Додати дані про книгу до таблиці articles.
7. Написати php-сценарій articles.php для виводу статей, дані про які знаходяться у таблиці articles.  
Алгоритм роботи сценарію:
- a) Приєднати модуль перевірки реєстрації користувача header\_login.inc.
  - b) На веб-сторінці головного фрейму вивести посилання на статті.
  - c) Вивести посилання на сторінку додавання нових ресурсів.
8. Модифікувавши сценарії addbook.php та library.php, створити сторінки addprograms.php та program.php для збереження програм користувачів порталу. Усі дані нових сценаріїв зберігати в окремій таблиці.
9. Модифікувавши сценарії addarticles.php та articles.php, створити сторінки add\_method\_developing.php та method\_developing.php для збереження методичних розробок користувачів порталу. Усі дані нових сценаріїв зберігати в окремій таблиці.

**Додаток 1**  
**Короткий довідник найбільш уживаних тегів html-розмітки**

<i>Тег</i>	<i>Опис тегу</i>	<i>Параметр</i>	<i>Призначення параметра</i>
<b>&lt;body&gt;</b> <b>&lt;/body&gt;</b>	Обрамляє весь вміст документу	<i>text</i>	задає колір тексту всього документа (по замовчуванню) колір задається в форматі #RRGGBB де RR-червона складова кольору виражена двоцифровим шіснадцятковим числом GG – зелена, BB – синя
		<i>bgcolor</i>	задає колір тла документа
		<i>background</i>	задає фонову картинку
<b>&lt;font&gt;</b> <b>&lt;/font&gt;</b>	Задає шрифт тексту	<i>color</i>	колір
		<i>size</i>	розмір, можна задати або фіксований розмір або наскільки збільшити(зменшити) розмір відносно розміру шрифту по замовчуванню
		<i>face</i>	тип шрифту (Arial, Courier і т.п.)
<b>&lt;i&gt;</b> <b>&lt;/i&gt;</b>	Виділяє текст курсивом		
<b>&lt;b&gt;</b> <b>&lt;/b&gt;</b>	Жирний шрифт		
<b>&lt;u&gt;</b> <b>&lt;/u&gt;</b>	Підкреслений текст		
<b>&lt;a&gt;</b> <b>&lt;/a&gt;</b>	Гіперпосилання	<i>href</i>	Задає шлях документу, який відкриється коли користувач натисне на дане гіперпосилання
		<i>target</i>	Задає вікно в якому відкриється документ Може приймати значення _self _parent _top blank
<b>&lt;img&gt;</b>	Вставляє малюнок	<i>src</i>	Шлях до файлу з малюнком
<b>&lt; table &gt;</b> <b>&lt;/ table &gt;</b>	Таблиця	<i>height</i>	Задає висоту таблиці в пікселях або в процентах відносно вікна документа

		<i>width</i>	Ширина таблиці
		<i>border</i>	Товщина рамки таблиці
<b>&lt;tr&gt;</b> <b>&lt;/tr&gt;</b>	Задає рядок таблиці		
<b>&lt;td&gt;</b> <b>&lt;td&gt;</b>	Задає клітинку таблиці		
<b>&lt;form&gt;</b> <b>&lt;/form&gt;</b>	Форма	<i>action</i>	Щлях до сторінки з скриптом, який виконуватиме обробку переданих даних.
		<i>enctype</i>	Формат даних, що передаються. За замовчуванням параметр набуває значення <code>application/x-www-form-urlencoded</code> , що відповідає передачі даних у текстовому форматі. У випадку завантаження файлів на сервер <code>enctype</code> повинен бути вказаний як <code>multipart/form-data</code> .
		<i>method</i>	Метод, що використовується для запиту даних. Може приймати значення <i>Get</i> чи <i>Post</i>
<b>&lt;input type="text"&gt;</b>	Текстове поле	<i>name</i>	ім'я змінної
		<i>size</i>	розмір (в символах)
		<i>maxlength</i>	максимальна кількість символів що можна ввести
		<i>value</i>	текст за замовчуванням
<b>&lt;input type="password"&gt;</b>	Поле введення паролю	<i>Параметри ті самі що і в текстового поля</i>	

<code>&lt;input type="hidden"&gt;</code>	Приховане текстове поле	<i>Параметри ті самі що і в текстового поля</i>	
<code>&lt;textarea&gt; &lt;/textarea&gt;</code>	Область тексту	<i>name</i>	ім'я змінної
		<i>rows</i>	Кількість рядків
		<i>cols</i>	Кількість символів в рядку
<code>&lt;input type="checkbox"&gt;</code>	Незалежний перемикач	<i>name</i>	ім'я змінної
		<i>checked</i>	Якщо присутній то перемикач увімкнутий по замовчуванню
		<i>value</i>	Значення яке набуде змінна коли перемикач увімкнутий
<code>&lt;input type="radio"&gt;</code>	Залежний перемикач	<i>name</i>	ім'я змінної
		<i>checked</i>	Якщо присутній то перемикач увімкнутий по замовчуванню
		<i>value</i>	Значення яке набуде змінна коли перемикач увімкнутий
<code>&lt;select &gt; &lt;/select&gt;</code>	Список вибору. Елементи списку задаються тегом <code>&lt;option&gt;</code>	<i>name</i>	ім'я змінної
		<i>size</i>	кількість видимих рядків (по замовчуванню 1)
		<i>multiple</i>	дозволяє користувачу вибрати одночасно кілька елементів (з допомогою клавіш ctrl і shift)
<code>&lt;option&gt; &lt;/option&gt;</code>	Елемент списку вибору. Вставляється між тегами <code>&lt;select &gt; і &lt;/select&gt;</code>	<i>value</i>	значення яке набуде змінна коли даний елемент вибрано
		<i>selected</i>	задає вибір по замовчуванню
<code>&lt;input type="submit"&gt;</code>	Кнопка відправки форми	<i>name</i>	ім'я змінної
		<i>value</i>	напис на кнопці
<code>&lt;input type="reset"&gt;</code>	Кнопка очищення форми	<i>name</i>	ім'я змінної
		<i>value</i>	напис на кнопці

## Додаток 2

### Список деяких математичних функцій php

**abs** - абсолютне значення

**acos** - арккосинус

**asin** - арксинус

**atan** - арктангенс

**cos** - косинус

**cosh** - гіперболічний косинус

**deg2rad** - конвертує число в градусах в еквівалент у радіанах

**exp** -  $e$  в степені ...

**log10** - логарифм із основою 10

**log** - натуральний логарифм

**max(\$arg1,\$arg2,...)** - повертає найбільше значення з значень параметрів \$arg1, \$arg2 ...

**pi** - значення  $\pi$

**pow(base,exp)** – Повертає  $base$ , в степені  $exp$ .

**rad2deg** - конвертує число в радіанах в еквівалент у градусах

**rand** - генерує випадкове число

**round** - округляє число із плаваючою крапкою/float

**sin** - синус

**sinh** - гіперболічний синус

**sqrt** - квадратний корінь

**tan** - тангенс

**tanh** - гіперболічний тангенс

### Додаток 3

## Список деяких функцій для роботи з файлами

bool **file\_exists** (string filename)

Повертає **TRUE**, якщо файл, визначений параметром *filename*, існує; **FALSE** в іншому випадку.

bool **is\_file** (string filename)

Перевіряє існування вказаного файлу, а також можливість виконання з ним операцій читання-запису.

int **fopen** (string filename, string mode)

Відкриває вказаний файл у вказаному режимі, “r” – читання, “w” – запису, “a” – додання даних, а також повертає цілочисельне значення (ідентифікатор відкритого файлу). Зазначений файл може бути розміщений на комп’ютері, на якому виконується Web-сервер, на іншому NNTP чи FTP-сервері.

bool **fclose** (int fp)

Закривається файл, на який вказує *fp*. Повертає **TRUE** при успіху, **FALSE** при невдачі.

string **fread** (int fp, int length)

**fread()** читає *length* байт із вказівника файлу, на який засилається *fp*. Читання зупиняється, якщо прочитано *length* байт або якщо досягнуто EOF (кінець файлу)

string **fgets** (int fp [, int length])

Повертає стрічку довжиною *length* байт, прочитану з файлу, на який вказує *fp*. Читання закінчується при досягненні точки *length* байт, символу newline (який включається в return-значення) або EOF (дивлячись що буде знайдено першим). Якщо *length* не визначений, за замовчуванням *length* буде 1k, або 1024 байт.

array **file** (string filename [, int use\_include\_path])

Повертає файл у вигляді масиву. Кожен елемент масиву відповідає рядку файлу із символом newline.

**Примітка:** кожний рядок у результуючому масиві буде містити символ кінця рядка, тому вам потрібно буде використати [trim\(\)](#), якщо символи кінця рядка вам не потрібні.

int **fwrite** (int fp, string string [, int length])

**fwrite()** записує вміст рядка *string* у потік файлу, специфікованого вказівником *fp*. Якщо аргумент *length* заданий, запис буде зупинена після запису *length* кількості байтів або досягнення кінця *string*. **fwrite()** повертає кількість записаних байт, або -1 при помилці.

bool **move\_uploaded\_file** (string filename, string destination)

Ця функція перевіряє, чи є файл *filename* правильно завантаженим файлом (що він був завантажений через PHP-механізм HTTP POST). Якщо файл є правильним, він буде перейменований(переміщений) у файл *destination*.

Якщо *filename* не є правильно завантаженим файлом, нічого не відбудеться, а **move\_uploaded\_file()** поверне **FALSE**.

Якщо *filename* є правильно завантаженим файлом, але не може бути переміщений з якихось причин, нічого не відбудеться, а **move\_uploaded\_file()** поверне **FALSE**. Крім того, буде видане попередження.



## Додаток 4

### Список деяких функцій для роботи зі стрічками

array **explode** (string separator, string string [, int limit])

Повертає масив стрічок, кожна з яких є підстрічкою стрічки *string* і сформована шляхом поділу стрічки роздільником утвореним сепаратором стрічки *separator*. Якщо встановлено *limit*, то масив що повертається буде містити максимум елементів *limit* з останнім елементом, що містить залишок *string*.

int **strlen** (string str)

Повертає довжину рядка *string*.

int **strpos** (string haystack, string needle)

Повертає числову позицію першого входження *needle* у рядку *haystack*. Якщо *needle* не знайдений, повертає **FALSE**.

string **strtolower** (string str)

Повертає *string* з усіма алфавітними символами, конвертованими в нижній регістр.

string **strtoupper** (string string)

Повертає *string* із алфавітними символами, конвертованими у верхній регістр.

int **substr\_count** (string haystack, string needle)

**substr\_count()** повертає кількість появ підстрічки *needle* у рядку *haystack*.

string **trim** (string str [, string charlist])

**Примітка:** необов'язковий параметр *charlist* був уведений в PHP 4.1. 0.

Ця функція повертає рядок з вирізаними на початку і вкінці рядка *str* пробілами. Без другого параметра **trim()** вирізує:

- " " (ASCII 32 (0x20)), звичайний space/пробіл.
- "\t" (ASCII 9 (0x09)), символ tab.
- "\n" (ASCII 10 (0x0A)), символ нового рядка (line feed).
- "\r" (ASCII 13 (0x0D)), символ carriage return/повернення каретки.
- "\0" (ASCII 0 (0x00)), NUL-байт.
- "\x0B" (ASCII 11 (0x0B)), символ vertical tab/вертикальної табуляції.

Ви можете також вказати символи які вирізати у параметрі *charlist*. Просто перелічите всі символи, які ви хочете вирізати. За допомогою .. ви можете вказати діапазон символів.

## Додаток 5

### Список деяких функцій для роботи з датою і часом

string **date** (string format [, int timestamp])

Повертає рядок, відформатований у відповідності з рядком формату з використанням заданого цілочисельного *timestamp* або поточного локального часу, якщо *timestamp*/"штамп часу" не заданий.

У рядку формату розпізнаються наступні символи:

- a - "am" або "pm"
- A - "AM" або "PM"
- d - день (число) місяця, 2 цифри із першим нулем, якщо необхідно; тобто від "01" до "31"
- D - день тижня, буквений, 3 букви; наприклад, "Fri"
- F - місяць, буквений, long; наприклад, "January"
- g - година, 12-годинний формат без перших нулів; тобто від "1" до "12"
- G - година, 24-годинний формат без перших нулів; тобто від "0" до "23"
- h - година, 12-годинний формат; тобто від "01" до "12"
- H - година, 24-годинний формат; тобто від "00" до "23"
- i - хвилини; тобто від "00" до "59"
- j - день (число) місяця без перших нулів; тобто від "1" до "31"
- l ('L' у нижньому регістрі) - день тижня, буквений, long; наприклад, "Friday"
- L - boolean, що вказує, чи високосний рік; тобто "0" або "1"
- m - місяць; тобто від "01" до "12"
- M - місяць, буквений, 3 букви; наприклад, "Jan"
- n - місяць без провідних нулів; тобто від "1" до "12"
- O - Різниця із часом за Гринвічем, у годинах; наприклад, "+0200"
- r - RFC 822 формат дати; наприклад, "Thu, 21 Dec 2000 16:01:07 +0200" (введений в PHP 4.0.4)
- s - секунди; тобто від "00" до "59"
- t - кількість днів у даному місяці; тобто від "28" до "31"
- U - секунди епохи Unix Epoch (починаючи з January 1 1970 00:00:00 GMT)
- w - день тижня, числовий, тобто від "0" (Sunday) до "6" (Saturday)
- W - ISO-8601 номер тижня в році, тижні починаються з понеділка/Monday (уведене в PHP 4.1.0)
- Y - рік, 4 цифри; наприклад, "1999"
- y - рік, 2 цифри; наприклад, "99"
- z - день року; тобто від "0" до "365"
- Z - зсув часового пояса, у секундах (тобто від "-43200" до "43200"). Зсув часових поясів на захід від UTC завжди від'ємний, а для поясів у сході від UTC - завжди додатній.

int **mktime** (int hour, int minute, int second, int month, int day, int year)

Повертає Unix timestamp, що відповідає заданим аргументам. Цей timestamp є довгим цілим/long integer числом, що вказує кількість секунд між початком Unix Epoch (January 1 1970) і вказаним часом.

int **time** (void) - Повертає поточний UNIX timestamp

Повертає поточний час, виміряний в секундах з початку Unix Epoch (January 1 1970 00:00:00 GMT)

## Додаток 6 Команди мови SQL

*Деякі команди мови SQL для визначення структури баз даних та таблиць:*

**CREATE DATABASE 'назва\_БД'** – створює базу даних

Для створення бази даних у мові PHP визначено функцію **mysql\_create\_db('назва\_БД')**

**DROP DATABASE 'назва\_БД'** – знищує базу даних

Для знищення бази даних у мові PHP визначено функцію **mysql\_drop\_db('назва\_БД')**

**CREATE TABLE 'назва\_таблиці' ('поле1' тип(довжина),'поле2' тип(довжина),... INDEX ( 'поле\_індексування' ))** – створює таблицю із вказаними полями та індексним полем.

**DROP TABLE 'назва\_таблиці'** – знищує таблицю

**ALTER TABLE 'назва\_таблиці' ADD 'нове\_поле ' тип(довжина)** – додає поле до таблиці

**ALTER TABLE 'назва\_таблиці' DROP 'поле '** – знищує поле в таблиці

*Деякі функції мови PHP для отримання результатів запитів:*

**mysql\_query(стрічка\_запиту\_на\_мові\_SQL)** – відправляє запит до БД.

**mysql\_num\_fields (ідентифікатор\_запиту)** – повертає кількість полів отриманих в результаті запиту. Ідентифікатором запиту є змінна, яка повертається функцією **mysql\_query**.

**mysql\_num\_rows(ідентифікатор\_запиту)** – повертає кількість записів отриманих в результаті запиту.

**mysql\_fetch\_row(ідентифікатор\_запиту)** – повертає масив як результат запиту.

**mysql\_fetch\_array(ідентифікатор\_запиту)** – повертає асоціативний масив як результат запиту, індексами якого є назви полів.

**mysql\_result(ідентифікатор\_запиту,індекс\_рядка\_елемента,індекс\_рядка\_елемента)** – повертає окреме значення запиту.

## Додаток 7 Сесії

Сеанс є групою змінних, які зберігаються з метою подальшого доступу до них, навіть після завершення виконання сценарію PHP. Кожна змінна сеансу зберігається при переході з однією сторінки на іншу без фізичної передачі цієї інформації. Змінні сеансу мають глобальну область дії, тобто є доступними із будь-якого іншого сценарію.

Для створення сесії використовується функція **session\_start()**, яка повертає значення логічної істини у випадку успішного створення сесії. Для реєстрації змінних сесії використовується функція **session\_register(змінна)**, яка повертає значення логічної істини у випадку успішної реєстрації змінної.

Приклад:

```
<?php
session_start(); // створюємо сесію
$user = "Peter";
// реєструємо змінну сесії
if (session_register("user")) {
echo("Змінну зареєстровано із значенням $user.");
}
?>
```

Примітка. Вищеописаний приклад буде справедливий для випадку встановленого параметру `register_globals = yes`. У іншому випадку звернення до змінних сесії слід здійснювати за допомогою асоціативних масивів `$_SESSION['user']` або `$HTTP_SESSION_VARS['user']`.

У розглянутих прикладах зустрічаються такі функції для роботи із сесіями:

**boolean session\_destroy()** видаляє всі дані, асоційовані з поточною сесією.

**session\_is\_registered(string name)** повертає логічне значення істини, якщо змінна з ім'ям *name*, зареєстрована в поточній сесії

## Література

1. Аргерих Л. и др. Профессиональное PHP программирование, 2-е издание. - Пер. с англ. - СПб:Символ-Плюс, 2003. - 1048 с.
2. Аткинсон, Леон. MySQL. Библиотека профессионала.: Пер. с англ. — М.: Издательский дом "Вильямс", 2002. — 624 с.
3. Гилмор В. PHP4. Учебный курс.— СПб.:Питер, 2001,— 352с.
4. Гончаров А. Самоучитель HTML. — СПб.: Питер, 2002. — 240 с.
5. Котеров Д.В. Самоучитель PHP 4. – СПб: БХВ-Петербург, 2003. – 567с.
6. Мазуркевич А. PHP: настольная книга программиста /Александр Мазуркевич, Дмитрий Еловой. — Мн.: Новое знание, 2003. — 480с.
7. Матросов А. В., Сергеев А. О., Чаунин М. П. HTML 4.0. - СПб.: БХВ-Петербург, 2003. - 672 с.
8. Морзе Н. В. Методика навчання інформатики. Частина 3. Методика навчання основних послуг глобальної мережі Інтернет. – К.: Навчальна книга, 2003. –196 с.
9. Петюшкин А. В. HTML. Экспресс-курс. — СПб.: БХВ-Петербург, 2003. — 256 с.
10. Полонская Е.Л. Язык HTML. Самоучитель. : — М. : Издательский дом "Вильямс", 2003.— 320 с.
11. Рамський Ю.С., Іваськів І.С., Ніколаєнко О.Ю. Вивчення Web-програмування в школі: Навчальний посібник. – Тернопіль: Навчальна книга – Богдан, 2004. – 200 с.
12. Спейнауэр С., Экштейн Р. Справочник вебмастера. – Пер. с англ. – СПб: Символ-Плюс, 2001.–608с.
13. Томсон Лаура Разработка Web-приложений на PHP и MySQL: Пер. с англ./Лаура Томсон, Люк Веллинг. — 2-е изд., испр. — СПб: ООО «ДиаСофт», 2003. — 672 с.
14. Ульман Л. Основы программирования на PHP: Пер. с англ. -М.: ДМК Пресс, 2001. -288 с.
15. Харрис Э. PHP/MySQL для начинающих/ Пер. с. англ. – М. КУДИЦ-ОБРАЗ, 2005– 384с.
16. Matt Zandstra SAMS Teach Yourself PHP4 in 24 Hours Sams Publishing, 2000–478с.

### Деякі освітні ресурси мережі Інтернет

1. <http://www.school.msu.ru/computer.php?id=2>
2. <http://www.abiturcenter.ru>
3. <http://sura.ru/dikov/lists.htm>
4. <http://ito.osu.ru/resour/links/>

5. <http://www.informika.ru>
6. <http://www.edu.ru>
7. <http://www.ege.ru>