Лабораторна робота №6.
Тема: Робота з рядками символів.
Мета: застосування функцій обробки рядків символів

ТЕОРЕТИЧНІ ВІДОМОСТІ

Розглянемо практичні приклади основних операцій з рядками в PHP: об'єднання рядків (конкатенація), порівняння рядків, конвертація рядків в ціле число та в масив, заміна рядка.

Конкатенація (об'єднання) рядків PHP

Конкатенація рядків – це метод маніпулювання рядками, коли ви об'єднуєте 2 або більше рядків. У PHP це завдання вирішується досить просто. Для цього ви можете використовувати оператор конкатенації, яким є '.' (крапка). Нижче наведено приклад об'єднання 2 або більше рядків таким чином:

```
<?php
$str1 = 'Це ';
$str2 = 'приклад ';
$str3 = 'кількох рядків';
$konkat = $str1 . $str2 . $str3;
echo $konkat; // Результат: Це приклад кількох рядків
?>
```

Крім цього, ви можете використовувати оператор конкатенації для додавання рядка до вже існуючого рядка:

```
<?php
$konkat  = 'Це початковий рядок';
$konkat .= ', а цей рядок був приєднаний';
echo $konkat; // Результат: Це початковий рядок, а цей
рядок був приєднаний
?>
```

Якщо об'єднуєте число з рядком, тоді число буде перетворено у значення рядка, тож вивід буде таким:

```
<?php
$i = 100;
$str = $i . ' - це число';
echo $str; // Результат: 100 - це число
?>
```

Приклади порівняння рядків у PHP

Час від часу потрібно порівнювати різні рядки і виконувати певні дії, в залежності від результату. Найпростіший спосіб порівняти рядки – це використовувати оператор порівняння (==):

```
<?php
$str1 = "Тест";
$str2 = "Тест";
if($str1 == "Тест") {
 echo "TRUE"; // Результат: TRUE
}
if($str1 == $str2) {
 echo "TRUE"; // Результат: TRUE
```

```
}
?>
```

Ви також можете використовувати вбудовані у PHP функції, такі як strcmp і strcasecmp. Ці функції роблять бінарно-безпечне порівняння рядків, при цьому strcasecmp є нечутливою до регістру. За допомогою цих функцій ви можете виявити, який рядок більший або менший, оскільки вони повертають < 0, якщо РЯДОК1 менше РЯДКА2, > 0, якщо РЯДОК1 більше РЯДКА2 і 0, якщо ці рядки рівні. Таким чином, ви можете використовувати ці функції так:

```
<?php
$str1 = "Тест";
$str2 = "тест";
if(strcmp($str1, "Тест") == 0) echo "TRUE"; //
Результат: TRUE
echo strcasecmp($str1, $str2); // Результат: -1
?>
```

Іноді трапляється, що рядки є рівними, але порівняння каже, що вони різні. Найбільш поширеною проблемою в даному випадку є наявність пробілів до або після відповідного тексту. Це призводить до того, що рядки стають різними. Щоб вирішити цю проблему, має сенс використовувати вбудовану в PHP функцію обробки trim, щоб видалити всі небажані пробіли:

```
<?php
$str1 = "Тест";
$str2 = "  Тест   ";
if($str1 == trim($str2)) {
 echo "TRUE"; // Результат: TRUE
}
if(strcmp($str1, trim($str2)) == 0) {
 echo "TRUE"; // Результат: TRUE
}
?>
```

Конвертування рядка PHP у ціле число (int, integer)

Іноді важливо мати значення змінної у форматі цілого числа (int, integer). Для прикладу, якщо відвідувачі вашого сайту заповнюють форму з полем, яке повинно бути представлене цілим числом. Однак у масиві $_POST ви отримуєте дані цього поля в якості рядка.

Перетворити рядок на ціле число в PHP досить просто. Потрібно лише використовувати тип даних. Для цього додайте (int) перед вашою змінною. Ось приклад, як це зробити:

```
<?php
$str = "100";
$num = (int)$str;
?>
```

Щоб перевірити, чи дійсно код працює, можна скористатися оператором ===. Цей оператор перевіряє не тільки значення, але й типи даних. Код може виглядати так:

```
<?php
$str = "100";
$num = (int)$str;
if ($str === 100) {echo "Рядок";}
```

```
if ($num === 100) {echo "Ціле число - int";}
?>
```

Перетворення рядків у масив в PHP

Щоб розділити рядок на складові частини і зберегти результат в масиві, можна використовувати функцію explode().

Синтаксис функції:

```
explode($delimiter, $string [,int $limit] )
delimiter – роздільник
string – початковий рядок
limit – обмеження символів для елемента (необов'язковий параметр)
```

Розглянемо практичний приклад:

```
<?php
$string = "item1, item2, item3, item4";
$delimiter = ",";
$itemList = explode($delimiter, $string);
var_dump($itemList);
// Результат: array(4) { [0]=> string(5) "item1" [1]=>
string(6) " item2" [2]=> string(6) " item3" [3]=>
string(6) " item4" }
?>
```

Але що станеться, якщо ви захочете передати до масиву кожне слово з такого тексту:

Демонстраційний текст з додатковими пробілами

Якщо ви будете використовувати пробіл в якості роздільника, тоді ваш масив буде містити також небажані додаткові пробіли. Для вирішення цієї проблеми можна попередньо підготувати текст перед обробкою. У цьому випадку потрібно замінити всі додаткові пробіли одним простим пробілом. Використовуючи функцію preg_replace, ви можете видалити небажані пробіли за допомогою наступного коду:

```
<?php
$string = preg_replace('/\s+/', ' ', $string);
?>
Тепер ви можете обробити текст таким чином:
<?php
$string =
'  Демонстраційний  текст  з    додатковими  пробілам
и      ';
$string = trim($string);
$string = preg_replace('/\s+/', ' ', $string);
$delimiter = ' ';
$itemList = explode($delimiter, $string);
var_dump($itemList);
// Результат: array(5) { [0]=> string(30)
"Демонстраційний" [1]=> string(10) "текст" [2]=>
string(2) "з" [3]=> string(22) "додатковими" [4]=>
string(18) "пробілами" }
?>
```

Тепер згадаємо третій, необов'язковий, параметр функції explode(). Цей параметр встановлює обмеження максимального розміру масиву. Якщо ви напишете такий код:

```
<?php
$itemList = explode($delimiter, $string, 3);
?>
```

Тоді розмір масиву буде обмежено 3-ма елементами, а ті елементи, що не вміщаються в ліміт, будуть об'єднані у останньому елементі. Тобто, вийде такий результат:

```
array(3) { [0]=> string(30) "Демонстраційний" [1]=>
string(10) "текст" [2]=> string(44) "з додатковими
пробілами" }
```

Заміна рядків у PHP

Однією з найважливіших і найбільш часто використовуваних маніпуляцій з рядками в PHP є функція заміни рядків. Ви можете замінити всі елементи в рядку або лише деякі з них. Ви можете обмежити заміну лише частиною вихідного рядка. Крім того, ви можете зробити заміну рядка використовуючи регулярні вирази.

PHP має наступні вбудовані функції маніпулювання рядками, використовуючи заміну:

```
str_replace
str_ireplace
preg_replace
strtr
```

Детальніше розглянемо ці функції.

str_replace

Ця функція замінює всі входження рядка пошуку на рядок заміни. Синтаксис функції str_replace такий:

str_replace (mixed $search , mixed $replace , mixed $subject [, int &$count])

search – значення для пошуку. Для декількох значень можна використовувати масив.

replace – рядок заміни, його буде використано для заміни шуканих значень search. Для декількох значень можна використовувати масив.

subject – рядок або масив, в якому проводиться пошук і заміна. Якщо subject є масивом, то пошук з заміною буде здійснюватися над кожним елементом subject, а результатом функції також буде масив.

count – якщо цей параметр переданий, тоді він буде обмежувати кількість проведених замін.

Спочатку ми визначаємо рядок, який потрібно замінити, а в якості другого параметра визначаємо новий рядок. Останнім обов'язковим параметром є сам основний рядок, в якому потрібно виконати заміну. Ось приклад використання функції str_replace:

```
<?php
$originalString = "Мені подобається червоний колір, у
мене є червоний автомобіль та червоний велосипед.";
$newString = str_replace("червоний", "зелений",
$originalString);
echo $originalString . "<br>";
echo $newString . "<br>";
?>
```

Результатом буде наступне:

Мені подобається червоний колір, у мене є червоний автомобіль та червоний велосипед.

Мені подобається зелений колір, у мене є зелений автомобіль та зелений велосипед.

str_ireplace

Ця функція дуже схожа на str_replace, але вона нечутлива до регістру.

preg_replace

Ця функція виконує пошук і заміну з використанням регулярних виразів. Синтаксис функції наступний:

```
preg_replace ( mixed $pattern , mixed $replacement , mixed $subject [, int $limit = -1 [, int &$count ]] )
```

pattern – регулярний вираз. Може бути як рядком, так і масивом рядків.

replacement – рядок або масив рядків для заміни.

subject – рядок або масив рядків для пошуку та заміни.

limit – максимально можлива кількість замін кожного виразу для кожного рядка subject. За замовчуванням дорівнює -1 (без обмежень).

count – якщо параметр вказано, то ця змінна буде заповнена кількістю проведених замін.

Приклад використання функції (видалення зайвих пробілів в рядку):

```
<?php
$str = '  Тест   №1';
$str = preg_replace('/\s\s+/', ' ', $str);
echo $str;
// Результат: Тест №1
?>
```

strtr

Ця функція замінює задані символи або замінює підрядки Синтаксис функції:

```
strtr ( string $str , string $from , string $to )
```

або

strtr (string $str , array $replace_pairs)

str – рядок для заміни (string).

from – рядок (string), який буде замінений на рядок to.

to – рядок (string), що замінює рядок from.

replace_pairs – параметр може бути використаний замість to і from, в цьому випадку він є масивом (array) та має форму array('from' => 'to', ...).

Приклад використання:

```
<?php
$trans = array("hello" => "hi", "hi" => "hello");
echo strtr("hi all, I said hello", $trans);
// Результат: hello all, I said hi
?>
```

ЗАВДАННЯ ДЛЯ САМОСТІЙНОГО ВИКОНАННЯ

Завдання 1. Робота з substr

Для вирішення завдань даного блоку вам знадобляться наступні функції: substr .

Дано рядок '**html css php**' . Виріжте з неї і виведіть на екран слово 'html', слово 'css' і слово 'php'.

Дано рядок. Вирежіте і виведіть на екран останні **3** символу цього рядка.

Дано рядок. Перевірте, що вона починається на '**http:** //' . Якщо це так, виведіть "так", якщо не так - "ні".

Дано рядок. Перевірте, що вона починається на '**http:** //' або на '**https:** //' . Якщо це так, виведіть "так", якщо не так - "ні".

Дано рядок. Перевірте, що вона закінчується на '**.png**' . Якщо це так, виведіть "так", якщо не так - "ні".

Дано рядок. Перевірте, що вона закінчується на '**.png**' або на '**.jpg**' . Якщо це так, виведіть "так", якщо не так - "ні".

Дано рядок. Якщо в цьому рядку більше 5-ти символів - виріжіть з неї перші **5** символів, додайте три крапки в кінець і виведіть на екран. Якщо ж в цьому рядку **5** і менше символів - просто виведіть цей рядок на екран.

Завдання 2. Робота з str_replace

Для вирішення завдань даного блоку вам знадобляться наступні функції: str_replace .

Дано рядок '**31 .12.2013** ' . Замініть всі точки на дефіси.

Дано рядок **$ str** . Замініть в ній всі букви 'a' на цифру 1, літери 'b' - на 2, а літери 'c' - на 3.

Дано рядок з буквами і цифрами, наприклад, ' **1a2b3c4b5d6e7f8g9h0** '. Видаліть з неї всі цифри. Тобто в нашому випадку повинна вийде рядок ' **abcbdefgh** '.

Завдання 3. Робота з strtr

Для вирішення завдань даного блоку вам знадобляться наступні функції: strtr .

Дано рядок **$ str** . Замініть в ній всі букви 'a' на цифру 1, літери 'b' - на 2, а літери 'c' - на 3. Вирішіть задачу двома способами роботи з функцією **strtr** (масив замін і два рядки замін).

Завдання 4. Робота з substr_replace

Для вирішення завдань даного блоку вам знадобляться наступні функції: substr_replace .

Дано рядок **$ str** . Вирежіте з неї подстроку з 3-го символу (відлік з нуля), 5 штук і замість неї вставте '!!!'.

Завдання 5. Робота з strpos, strrpos

Для вирішення завдань даного блоку вам знадобляться наступні функції: strpos , strrpos .

Дано рядок 'abc abc abc'. Визначте позицію першої літери 'b'.

Дано рядок 'abc abc abc'. Визначте позицію останньої літери 'b'.

Дано рядок 'abc abc abc'. Визначте позицію першої знайденої літери 'b', якщо почати пошук не з початку рядка, а з позиції 3.

Дано рядок 'ааа ааа ааа ааа ааа'. Визначте позицію другого пробілу.

Перевірте, що в рядку є дві точки поспіль. Якщо це так - виведіть 'є', якщо не так - "ні".

Перевірте, що рядок починається на '**http: //**' . Якщо це так - виведіть "так", якщо не так - "ні".

Завдання 6. Робота з explode, implode

Для вирішення завдань даного блоку вам знадобляться наступні функції: explode , implode .

Дано рядок '**html css php**' . За допомогою функції explode запишіть кожне слово цього рядка в окремий елемент масиву.

Дан масив з елементами '**html', 'css', 'php**' . За допомогою функції implode створіть рядок з цих елементів, розділених комами.

В змінної **$ date** лежить дата в форматі '**2013-12-31**' . Перетворіть цю дату в формат '**31 .12.2013** ' .

Завдання 7. Робота з str_split

Для вирішення завдань даного блоку вам знадобляться наступні функції: str_split .

Дано рядок '**1234567890**' . Розбийте її на масив з елементами '**12', '34', '56', '78', '90**' .

Дано рядок '**1234567890**' . Розбийте її на масив з елементами '**1', '2', '3', '4', '5', '6', '7', '8', '9', '0**' .

Дано рядок '**1234567890**' . Зробіть з неї рядок '**12 -34-56-78-90** ' не використовуючи цикл.

Завдання 8. Робота з trim, ltrim, rtrim

Для вирішення завдань даного блоку вам знадобляться наступні функції: trim , ltrim , rtrim .

Дано рядок. Очистіть її від кінцевих пробілів.

Дано рядок '**/ php /**' . Зробіть з неї рядок '**php**' , видаливши кінцеві слеші.

Дано рядок **'слова слова слова.'** . В кінці цього рядка може бути крапка, а може і не бути. Зробіть так, щоб в кінці цього рядка гарантовано стояла крапка. Тобто: якщо цієї точки немає - її треба додати, а якщо є - нічого не робити. Завдання вирішите через **rtrim** без всяких іфов.

Завдання 9. Робота з strrev

Для вирішення завдань даного блоку вам знадобляться наступні функції: strrev .

Дано рядок '**12345**' . Зробіть з неї рядок '**54321**' .

Перевірте, чи є слово **паліндромом** (однаково читається у всіх напрямках, приклади таких слів: **madam, otto, kayak, nun, level**).

Завдання 10. Робота з str_shuffle

Для вирішення завдань даного блоку вам знадобляться наступні функції: str_shuffle .

Дано рядок. Перемішайте символи цього рядка у випадковому порядку.

Створіть рядок з 6-ти випадкових маленьких латинських літер так, щоб букви не повторювалися. Потрібно зробити так, щоб в нашій рядку могла бути будь-яка латинська буква, а не обмежений набір.

Завдання 11. Робота з number_format

Для вирішення завдань даного блоку вам знадобляться наступні функції: number_format .

Дано рядок **'12345678'** . Зробіть з неї рядок **'12 345 678 '** .