**Робота № 16**

**Тема: РЕГУЛЯРНІ ВИРАЗИ**

**Мета:** Навчитися будувати і застосовувати регулярні вирази

**Завдання:**

1) Вивчити правила формування регулярних виразів.

2) Виконати всі пункти лабораторної роботи.

3) Надати звіт про виконану роботу.

Регулярні вирази (regular expressions) - це досить складна, але цікава функціональна можливість, яка дозволяє зробити основні і найнеобхідніші перевірки змінних або зробити маніпуляції над рядками. Регулярні вирази являють собою шаблон і покликані перевірити, чи відповідає йому рядок. Спільно з перевіркою може здійснюватися вибірка або заміна знайденого підрядка.  
  
Мова PHP підтримує два типи регулярних виразів - стандартні POSIX і Perl. Функції для роботи з такими виразами використовуються різні.  
Той факт, що PHP підтримує два типи шаблонів, є незаперечною перевагою, тому що це перетворює його в дуже потужний інструмент забезпечення безпеки, а безпека не буває надмірною. Деякі мови взагалі не мають такої можливості, і її доводиться реалізовувати самостійно за допомогою функцій для маніпуляцій над рядками. В цьому випадку зростає ймовірність помилки в реалізації перевірок, якою хакер неодмінно скористається.  
  
**Використання регулярних виразів Perl**  
  
Регулярні вирази в стилі Perl надають більше можливостей  
  
Регулярні вирази Perl укладаються між двома слеш. Наприклад, наступний шаблон відповідає слову hacker: "/hacker/". Після шаблону можуть зазначатися модифікатори в наступному вигляді:

/Шаблон/модифікатори

Модифікаторами є літери, які впливають на регулярний вираз. Найбільш популярними модифікаторами Perl є "i" та "x". Розглянемо їх на прикладах.  
  
- **i** - ігнорувати регістр букв. Це означає, що регулярний вираз "/Hacker/i" буде відповідати словами hacker, HACKER, HacKer і т.д.  
  
- **x** - ігнорувати в шаблоні пробіли, переведення рядків і коментарі. Це дозволяє вам використовувати в шаблоні коментарі, щоб код був більш читабельним.  
  
- **m** - за замовчуванням текст розглядається як один рядок, але якщо вказаний цей модифікатор, то можна використовувати як багаторядковий текст.

-**g**-глобальний пошук.  
  
Можна вказувати відразу кілька модифікаторів.  
  
**Регулярні вирази (шаблони)**

У шаблонах використовуються наступні метасимволи (символи позначають групи інших символів) часто звані egrep - стандартом:

|  |  |
| --- | --- |
| **\** | - Вважати наступний метасимвол як звичайний символ. |
| **^** | - Початок рядка |
| **.** | - Один довільний символ. Крім '\n' - кінець рядка. |
| **$** | - Кінець рядка |
| **|** | - Альтернатива (або) |
| **()** | - Групування |
| **[]** | - Клас символів |

Метасимволи мають модифікатори (пишуться після метасимвола):

|  |  |
| --- | --- |
| **\*** | - Повторюється 0 або більше число разів |
| **+** | - - // - 1 або більше число разів |
| **?** | - 1 або 0 раз |
| **{n}** | - Точно n разів |
| **{n,}** | - Щонайменше раз |
| **{n, m}** | - Не менше ніж n, але і не більше m |

У всіх інших випадках фігурні дужки вважаються звичайними (регулярними) символами. Таким чином '\*' еквівалентна {0,}, '+' - {1,} і '?' - {0,1}. n і m не можуть бути більше 65536.

За замовчуванням дія метасимволів "жадібна" (greedy). Збіг поширюється стільки раз, скільки можливо, не враховуючи результат дії наступних метасимволів. Якщо ви хочете "зменшити їх апетит", то використовуйте символ '?'. Це не змінює значення метасимволів, просто зменшує поширення. Таким чином:

|  |  |
| --- | --- |
| **\*?** | - Стане 0 і більше |
| **+?** | - 1 і більше |
| **??** | - 0 або 1 раз |
| **{n}?** | - Точно n разів |
| **{n,}?** | - Не менше ніж n раз |
| **{n, m}?** | - Більше або дорівнює n і менше m разів |

В шаблонах можна використовувати `\` - символи (бекслеш-символи):

|  |  |
| --- | --- |
| **\t** | - Символ табуляції |
| **\n** | - Новий рядок |
| **\r** | - Переведення каретки |
| **\а** | - Переведення формату |
| **\v** | - Вертикальна табуляція |
| **\033** | - Вісімковий запис символу |
| **\x1A** | - Шістнадцятковий запис символу |
| **\c[** | - Control символ |
| ***\l*** | - Нижній регістр наступного символу |
| **\u** | - Верхній регістр -//- |
| **\L** | - Всі символи в нижньому регістрі до \E |
| **\U** | - Всі символи в верхньому регістрі до \E |
| **\E** | - Обмежувач зміни регістру |
| **\Q** | - Скасування дії як метасимвола |

Додатково в Perl додані наступні метасимволи:

|  |  |
| --- | --- |
| **\w** | - Алфавітно-цифровий або '\_' символ |
| **\W** | - Не - // - |
| **\s** | - Один пробіл |
| **\S** | - Один не пробіл |
| **\d** | - Одна цифра |
| **\D** | - Одна не цифра |

Зверніть увагу, що все це "один" символ. Для позначення послідовності застосовуйте модифікатори. так:

|  |  |
| --- | --- |
| **\w+** | - слово |
| **\d+** | - ціле число |
| **[+ -]?\d+** | - Ціле зі знаком |
| **[+ -]?\d+\.?\d\*** | - Число з точкою |

Крім того існують уявні метасимволи. Позначають неіснуючі символи в місці зміни значення. Такі як:

|  |  |
| --- | --- |
| **\b** | - Межа слова |
| **\B** | - Не межа слова |
| **\A** | - Початок рядка |
| **\Z** | - Кінець рядка |
| **\G** | - Кінець дії m // g |

Межа слова (\b) - це уявна точка між символами \w і \W. Метасимволи **\A** і **\Z** - аналогічні '^' і '$', але якщо початок рядка '^' і кінець рядка '$' діють для кожного рядка в багаторядковому рядку, то **\A** і **\Z** позначають початок і кінець всього багаторядкового рядка .

Отже, у регулярних виразах Perl зворотний слеш "\" має спеціальне значення, але якщо необхідно вказати цей символ, не вкладаючи певного сенсу, то вкажіть його двічі.

Якщо всередині шаблону застосовується груповання (круглі дужки), то номер підрядка групи позначається як '\цифра'. Зауважте, що за шаблоном в межах виразу або блоку ці групи позначаються як '$цифра'. Крім цього існують додаткові змінні:

|  |  |
| --- | --- |
| **$+** | - Позначає останній збіг |
| **$&** | - Весь збіг |
| **$`** | - Все до збігу |
| **$'** | - Все після збігу |

*приклад:*

$S = "Один 1 два 2 і три 3";

if ($s = ~ / (\d +) \D + (\d +) /)

{

print "$1 \n"; # Результат '1'

print "$2 \n"; # '2'

print "$+ \n"; # '2'

print "$& \n"; # '1 два 2'

print "$` \n "; # 'Один '

print "$'\n"; # 'і три 3'

}

Perl версії 5 містить додаткові конструкції шаблонів:

|  |  |
| --- | --- |
| **(?#коментар)** | - Коментар в тілі шаблону. |
| **(?:шаблон)** | - Групування як і '()', але без зворотного посилання |
| **(?=шаблон)** | - "Заглядання" вперед. Наприклад / \w + (? = \t) / відповідає слову, за яким йде табуляція, але символ '\t' не включається в результат. |

*приклад:*

$S = "1 + 2-3 \* 4";

if ($s = ~ / (\d) (? = -) /) # Знайти цифру за якою стоїть '-'

{

print "$1 \n"; # Результат '2'

}

else

{

print "помилка пошуку \n";

}

(?! Шаблон) - "заглядання" вперед по заперечення:

*приклад:*

$S = "1 + 2-3 \* 4";

if ($s = ~ / (\d) (?! \+) /) # Знайти цифру за якою не слідує '+'

{

print "$1 \n"; # Результат '2'

}

else

{

print "помилка пошуку \n";

}

*(? i*smx) - "внутрішні" модифікатори. Зручно застосовувати в шаблонах, де наприклад потрібно всередині шаблону вказати модифікатор.

Розглянемо приклад. Припустимо, що нам потрібно вказати три послідовно розміщенні цифри. Для цього можна використовувати наступний регулярний вираз:

/\d\d\d/

Те ж саме можна записати інакше:

/\d{3}/

Тепер подивимося, як можна створити шаблон з цифри, букви і цифри:

/\d\w\d/

Квадратні дужки можна використовувати у регулярних виразах для задання діапазону можливих значень. Наприклад, ви хочете знайти будь-які числа від 0 до 9:[**0123456789] чи[0-9]**.  
  
Щоб регулярний вираз відповідав будь-якому символу латинського алфавіту, слід написати[**a-z]**. Але останній вираз визначає тільки прописні букви. Щоб включити в нього ще й великі, напишемо[**a-zA-Z]**.  
  
У квадратних дужках ми перераховуємо необхідні символи або діапазони, а можна і те й інше одночасно. Наприклад, такий вираз шукає будь-яку букву латинського алфавіту, цифру або символи підкреслення, тире і пробілу:

[0-9a-zA-Z-\_]

Ось регулярний вираз, який відповідає будь-яким цифрам і заголовним буквам: **/[0-9A-Z]/**.

Ускладнимо завдання. Припустимо, що необхідно знайти рядок, в якому спочатку йдуть від 3 до 5 заголовних букв, потім пробіл і далі від 3 до 7 цифр. У вигляді регулярного виразу це буде виглядати наступним чином:

/[A-Z]{3,5}\s\d{3,7}/

А наступний приклад показує, як замінити в рядку символи, що входять до діапазону, на символ "Х" за допомогою функції *preg\_replace()*:

$text = "df2345@sdfsdfsd334df";

$newtext = preg\_replace( "/[0-9A-Z]/", "X", $text);

echo $newtext;

Символ "^" означає заперечення. Якщо необхідно замінити на символ "Х" все, крім цифр 1, 2 і 3, то використовуємо шаблон **/[^123]/**:

$newtext = preg\_replace( "/[^123]/", "X", $text);

У регулярних виразах Perl дуже важливе значення має зворотний слеш.

**Правила регулярного виразу.** **(regex)**

1. Будь-який символ позначає себе самого, якщо це не метасимвол. Якщо вам потрібно скасувати дію метасимвола, то поставте перед ним '\'.
2. Рядок символів позначає рядок цих символів.
3. Множина можливих символів (клас) у квадратних дужках '[]', це означає, що в даному місці може стояти один із зазначених у дужках символів. Якщо перший символ в дужках це '^' - значить жоден із зазначених символів не може стояти в даному місці виразу. Усередині класу можна вживати символ '-', що позначає діапазон символів. Наприклад, a-z - одна з малих букв латинського алфавіту, 0-9 - цифра і т.д.
4. Альтернативні послідовності розділяються символом '|' Зауважте що всередині квадратних дужок це звичайний символ.
5. Усередині регулярного виразу можна вказувати "підшаблони" укладаючи їх в круглі дужки і посилатися на них як '\номер' Перша дужка позначається як '\1'.

**Функції регулярних виразів**   
Познайомимося з функціями для роботи з регулярними виразами.

# preg\_match

(PHP 4, PHP 5, PHP 7)

preg\_match - Виконує перевірку на відповідність регулярному виразу

### опис [¶](http://php.net/manual/ru/function.preg-match.php#refsect1-function.preg-match-description)

**int** preg\_match (string $pattern , string $subject[, array &$matches[, int $flags = 0[, int$offset = 0]]])

Шукає в заданому тексті **subject** збіги з шаблоном **pattern** .

### список параметрів [¶](http://php.net/manual/ru/function.preg-match.php#refsect1-function.preg-match-parameters)

**pattern**

Шуканий шаблон, рядок.

**subject**

Вхідний рядок.

**matches**

У разі, якщо зазначений додатковий параметр **matches** , він буде заповнений результатами пошуку. Елемент $matches[0] буде містити частину рядка, відповідну входженню всього шаблону, $matches[1] - частина рядка, які відповідають першій підмасці, і так далі.

**flags**

**flags** може приймати значення наступного прапора:

**PREG\_OFFSET\_CAPTURE**

У разі, якщо цей прапор вказано, для кожного знайденого підрядка буде вказана її позиція в заданій стрічці. Необхідно пам'ятати, що цей прапор змінює формат повертається масиву **matches** в масив, кожен елемент якого містить масив, що містить в індексі з номером 0 знайдений підрядок, а зміщення цього підрядка в параметрі **subject** - в індексі 1.

**offset**

Зазвичай пошук здійснюється зліва направо, з початку рядка. Можна використовувати додатковий параметр **offset** для вказівки альтернативної початкової позиції для пошуку (в байтах).

**зауваження:**

Використання параметра **offset** не еквівалентне заміні співставляючого рядка виразом substr ($subject, $offset) при виконанні функції **preg\_match (),** оскільки шаблон **pattern** може містити такі умови як ^, $ або (? <= X). Порівняйте:

<?php   
$subject = "abcdef" ;   
$pattern = '/^def/' ;   
preg\_match ( $pattern , $subject , $matches , PREG\_OFFSET\_CAPTURE , 3 );   
print\_r ( $matches );   
?>

Результат виконання даного прикладу:

Array

(

)

У той час як цей приклад

<?php   
$subject = "abcdef" ;   
$pattern = '/^def/' ;   
preg\_match ( $pattern , substr ( $subject , 3 ), $matches , PREG\_OFFSET\_CAPTURE );   
print\_r ( $matches );   
?>

виведе наступне:

Array

(

[0] => Array

(

[0] => def

[1] => 0

)

)

### значення, що повертаються [¶](http://php.net/manual/ru/function.preg-match.php#refsect1-function.preg-match-returnvalues)

**preg\_match ()** повертає 1, якщо параметр **pattern** відповідає переданому параметру **subject** , 0 якщо ні, чи **FALSE** в разі помилки.

**Увага**

Ця функція може повертати як boolean **FALSE** , так і не-boolean значення, яке приводиться до **FALSE** . За більш детальною інформацією зверніться до розділу [Булев тип](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://php.net/manual/ru/language.types.boolean.php&usg=ALkJrhgRcVdE9w9_x2EdlATpjxMXXoQVxQ) . Використовуйте [оператор ===](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://php.net/manual/ru/language.operators.comparison.php&usg=ALkJrhgjWApy1bE0aBkxQD27PC0aJyYabg) для перевірки значення, що повертається цією функцією.

Ця функція дуже зручна для перевірки, чи відповідає рядок певному шаблону. Наприклад, якщо ви хочете перевірити відповідність змінної *$text* шаблоном електронної поштової адреси, то можна виконати наступного код:

$r = preg\_match( "/^([a-zA-Z0-9\.\_\-]+@[a-zA-Z0-9\.\_\-]+(\.[a-zA-Z0-9]+)+)+$/", $text );

if ( !$r ) {

exit( "Помилка формату поштової скриньки");

}

### приклади [¶](http://php.net/manual/ru/function.preg-match.php#refsect1-function.preg-match-examples)

**Приклад # 1 Пошук підрядка "php" в тексті**

<?php   
// Символ "i" после закрывающего ограничителя шаблона означает   
// регистронезависимый поиск.   
if ( preg\_match ( "/php/i" , "PHP is the web scripting language of choice." )) {   
echo "Вхождение найдено." ;   
} else {   
echo "Вхождение не найдено." ;   
}   
?>

**Приклад # 2 Пошук слова "web" в тексті**

<?php   
/\* Специальная последовательность \b в шаблоне означает границу слова,   
следовательно, только изолированное вхождение слова 'web' будет соответствовать маске, в отличие от "webbing" или "cobweb" \*/   
if ( preg\_match ( "/\bweb\b/i" , "PHP is the web scripting language of choice." )) {   
echo "Вхождение найдено." ;   
} else {   
echo "Вхождение не найдено." ;   
}   
  
if ( preg\_match ( "/\bweb\b/i" , "PHP is the website scripting language of choice." )) {   
echo "Вхождение найдено." ;   
} else {   
echo "Вхождение не найдено." ;   
}   
?>

**Приклад # 3 Витяг доменного імені з URL**

<?php   
// Извлекаем имя хоста из URL   
preg\_match ( '@^(?:http://)?([^/]+)@i' ,   
"http://www.php.net/index.html" , $matches );   
$host = $matches[1];   
  
// извлекаем две последние части имени хоста   
preg\_match ( '/[^.]+\.[^.]+$/' , $host , $matches );   
echo "доменное имя: { $matches[0]} \n" ;   
?>

Результат виконання даного прикладу:

доменне ім'я: php.net

**Приклад # 4 Використання іменованих підмасок**

<?php   
  
$str = 'foobar: 2008' ;   
  
preg\_match ( '/(?P<name>\w+): (?P<digit>\d+)/' , $str , $matches );   
  
/\* Это также работает в PHP 5.2.2 (PCRE 7.0) и более поздних версиях,   
\* однако, вышеуказанная форма рекомендуется для обратной совместимости \*/   
// preg\_match('/(?<name>\w+): (?<digit>\d+)/', $str, $matches);   
  
print\_r ( $matches );   
  
?>

Результат виконання даного прикладу:

Array

(

[0] => foobar: 2008

[Name] => foobar

[1] => foobar

[Digit] => 2008

[2] => 2008

)

### Примітки [¶](http://php.net/manual/ru/function.preg-match.php#refsect1-function.preg-match-notes)

**Підказка**

Не використовуйте функцію **preg\_match (),** якщо необхідно перевірити наявність підрядка в заданому рядку. Використовуйте для цього [strpos ()](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://php.net/manual/ru/function.strpos.php&usg=ALkJrhjhu8ro5Sjq0KpnU7oscRHD3F_uEA) оскільки вона виконує це завдання набагато швидше.

# preg\_match\_all

(PHP 4, PHP 5, PHP 7)

preg\_match\_all - Виконує глобальний пошук шаблону в рядку

### опис [¶](http://php.net/manual/ru/function.preg-match-all.php#refsect1-function.preg-match-all-description)

**int** preg\_match\_all (string $pattern , string $subject[, array &$matches[, int $flags =PREG\_PATTERN\_ORDER[, int $offset = 0]]])

Шукає в рядку **subject** всі збіги з шаблоном **pattern** і поміщає результат в масив **matches** в порядку, визначеному комбінацією прапорів **flags** .

Після знаходження першої відповідності наступні пошуки будуть здійснюватися не з початку рядка, а від кінця останнього знайденого входження.

### список параметрів [¶](http://php.net/manual/ru/function.preg-match-all.php#refsect1-function.preg-match-all-parameters)

**pattern**

Шуканий шаблон, рядок.

**subject**

Вхідний рядок.

**matches**

Масив співпавших значень, відсортований відповідно до параметром **flags** .

**flags**

Може бути комбінацією наступних прапорів (необхідно розуміти, що використання **PREG\_PATTERN\_ORDER** одночасно з **PREG\_SET\_ORDER** безглуздо):

**PREG\_PATTERN\_ORDER**

Впорядковує результати так, що елемент $matches[0] містить масив повних входжень шаблону, елемент $matches[1] містить масив входжень першої підмаски, і так далі.

<?php   
preg\_match\_all ( "|<[^>]+>(.\*)</[^>]+>|U" ,   
"<b>пример: </b><div align=left>это тест</div>" ,   
$out , PREG\_PATTERN\_ORDER );   
echo $out[0][0] . ", " . $out[0][1] . "\n" ;   
echo $out[1][0] . ", " . $out[1][1] . "\n" ;   
?>

Результат виконання даного прикладу:

<B> приклад: </ b>, <div align = left> це тест </ div>

приклад:, це тест

Таким чином, $out[0] містить масив повних входжень шаблону, а $out[1] містить масив підрядків, що містяться в тегах.

Якщо шаблон містить іменовані підшаблони, $matches додатково буде містити записи для ключів з іменами цих підшаблонів.

Якщо шаблон містить дубльовані підшаблони, то тільки найправіший підшаблон буде збережений в $matches[NAME].

<?php   
preg\_match\_all (   
'/(?J)(?<match>foo)|(?<match>bar)/' ,   
'foo bar' ,   
$matches ,   
PREG\_PATTERN\_ORDER   
);   
print\_r ( $matches['match']);   
?>

Результат виконання даного прикладу:

Array

(

[0] =>

[1] => bar

)

**PREG\_SET\_ORDER**

Впорядковує результати так, що елемент $matches[0] містить перший набір входжень, елемент $matches[1] містить другий набір входжень, і так далі.

<?php   
preg\_match\_all ( "|<[^>]+>(.\*)</[^>]+>|U" ,   
"<b>пример: </b><div align=\"left\">это тест</div>" ,   
$out , PREG\_SET\_ORDER );   
echo $out[0][0] . ", " . $out[0][1] . "\n" ;   
echo $out[1][0] . ", " . $out[1][1] . "\n" ;   
?>

Результат виконання даного прикладу:

<B> приклад: </ b>, приклад:

<Div align = "left"> це тест </ div>, це тест

**PREG\_OFFSET\_CAPTURE**

У разі, якщо цей прапор вказано, для кожного знайденого підрядка буде вказана його позиція в заданому рядку. Необхідно пам'ятати, що цей прапор змінює формат повернутого масиву **matches** в масив, кожен елемент якого містить масив, що містить в індексі з номером 0 знайдений підрядок, а зміщення цього підрядка в параметрі **subject** - в індексі 1.

У разі якщо прапори не були вказані, за умовчанням використовується **PREG\_PATTERN\_ORDER** .

**offset**

Зазвичай пошук здійснюється зліва направо, з початку рядка. Додатковий параметр **offset** може бути використаний для вказівки альтернативної початкової позиції для пошуку.

.

### значення, що повертаються [¶](http://php.net/manual/ru/function.preg-match-all.php#refsect1-function.preg-match-all-returnvalues)

Повертає кількість знайдених входжень шаблону (яке може бути і нулем) або **FALSE** , якщо під час виконання виникли будь-які помилки.

### список змін [¶](http://php.net/manual/ru/function.preg-match-all.php#refsect1-function.preg-match-all-changelog)

| **версія** | **опис** |
| --- | --- |
| 5.4.0 | Параметр **matches** став необов'язковим. |
| 5.3.6 | Повертає **FALSE** якщо **offset** більше ніж довжина **subject** . |
| 5.2.2 | Іменовані підмаски тепер приймають синтаксис (? <Name>) і (? 'Name'), також як і (? P <name>). Попередні версії брали тільки (? P <name>). |

### приклади [¶](http://php.net/manual/ru/function.preg-match-all.php#refsect1-function.preg-match-all-examples)

**Приклад # 1 Отримання всіх телефонних номерів з тексту.**

<?php   
preg\_match\_all ( "/\(? (\d{3})? \)? (?(1)[\-\s] ) \d{3}-\d{4}/x" ,   
"Звоните 555-1212 или 1-800-555-1212" , $phones );   
?>

**Приклад # 2 Жадібний пошук збігів з HTML-тегами**

<?php   
// Запись \\2 является примером использования ссылок на підмаски.   
// Она означает необходимость соответствия подстроки строке, захваченной   
// второй подмаской, в нашем примере это ([\w]+).   
// Дополнительный обратный слеш необходим, так как используются двойные кавычки.   
$html = "<b>полужирный текст</b><a href=howdy.html>нажми</a>" ;   
  
preg\_match\_all ( "/(<([\w]+)[^>]\*>)(.\*?)(<\/\\2>)/" , $html , $matches , PREG\_SET\_ORDER );   
  
foreach ( $matches as $val ) {   
echo "совпадение: " . $val[0] . "\n" ;   
echo "часть 1: " . $val[1] . "\n" ;   
echo "часть 2: " . $val[2] . "\n" ;   
echo "часть 3: " . $val[3] . "\n" ;   
echo "часть 4: " . $val[4] . "\n\n" ;   
}   
?>

Результат виконання даного прикладу:

збіг: <b> напівжирний текст </ b>

частина 1: <b>

частина 2: b

частина 3: напівжирний текст

частина 4: </ b>

matched: <a href=hody.html> натисни </a>

частина 1: <a href=howdy.html>

частина 2: a

частина 3: натисни

частина 4: </a>

**Приклад # 3 Використання іменованих підмасок**

<?php   
  
$str = <<<FOO   
a: 1   
b: 2   
c: 3   
FOO;   
  
preg\_match\_all ( '/(?P<name>\w+): (?P<digit>\d+)/' , $str , $matches );   
  
/\* Это также работает в PHP 5.2.2 (PCRE 7.0) и более поздних версиях,   
\* однако, вышеуказанная форма рекомендуется для обратной совместимости \*/   
// preg\_match\_all('/(?<name>\w+): (?<digit>\d+)/', $str, $matches);   
  
print\_r ( $matches );   
  
?>

Результат виконання даного прикладу:

Array

(

[0] => Array

(

[0] => a: 1

[1] => b: 2

[2] => c: 3

)

[Name] => Array

(

[0] => a

[1] => b

[2] => c

)

[1] => Array

(

[0] => a

[1] => b

[2] => c

)

[Digit] => Array

(

[0] => 1

[1] => 2

[2] => 3

)

[2] => Array

(

[0] => 1

[1] => 2

[2] => 3

)

)

# preg\_replace

(PHP 4, PHP 5, PHP 7)

preg\_replace - Виконує пошук і заміну по регулярному виразу

### опис [¶](http://php.net/manual/ru/function.preg-replace.php#refsect1-function.preg-replace-description)

[mixed](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://php.net/manual/ru/language.pseudo-types.php&usg=ALkJrhie_lgL-HgaNsLRXVB5jbHBqBYGGw#language.types.mixed) **preg\_replace** ( [mixed](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://php.net/manual/ru/language.pseudo-types.php&usg=ALkJrhie_lgL-HgaNsLRXVB5jbHBqBYGGw#language.types.mixed) $pattern , [mixed](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://php.net/manual/ru/language.pseudo-types.php&usg=ALkJrhie_lgL-HgaNsLRXVB5jbHBqBYGGw#language.types.mixed) $replacement , [mixed](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://php.net/manual/ru/language.pseudo-types.php&usg=ALkJrhie_lgL-HgaNsLRXVB5jbHBqBYGGw#language.types.mixed) $subject[, int $limit = -1[, int &$count]])

Виконує пошук збігів в рядку **subject** з шаблоном **pattern** і замінює їх на **replacement** .

### список параметрів [¶](http://php.net/manual/ru/function.preg-replace.php#refsect1-function.preg-replace-parameters)

**pattern**

Шуканий шаблон. Може бути як рядком, так і масивом рядків.

Також доступні деякі [модифікатори PCRE](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://php.net/manual/ru/reference.pcre.pattern.modifiers.php&usg=ALkJrhhjFi9lz29M2wbHoRWz09SQJwp-Jg) .

**replacement**

Рядок або масив рядків для заміни. Якщо цей параметр є рядком, а **pattern** є масивом, всі шаблони будуть замінені цим рядком. Якщо і **pattern** і **replacement** є масивами, кожен елемент **pattern** буде замінений відповідним елементом з **replacement** . Якщо масив **replacement** містить менше елементів, ніж масив **pattern** , то всі зайві шаблони з **pattern** будуть замінені порожніми рядками.

**replacement** може містити посилання виду \\*n,* або (починаючи з PHP 4.0.4) $*n,* причому останній варіант краще. Кожне таке посилання буде замінене на підрядок, відповідний n -ій підмасці. N може приймати значення від 0 до 99, причому посилання \\0 (або $0) відповідає входженню всього шаблону. Підмаски нумеруються зліва направо, починаючи з одиниці. Для використання зворотного слеша, його необхідно продублювати (рядок PHP "\\\\").

При заміні за шаблоном з використанням посилань на підмасці може виникнути ситуація, коли безпосередньо за маскою слідує цифра (наприклад, установка цифри відразу після збіглася маски). У такому разі не можна використовувати знайому нотацію виду \\1 для посилання на підмасці. Запис, наприклад, \\11, «збентежить» **preg\_replace (),** так як вона не зможе зрозуміти, чи хочете ви використовувати посилання \\1, за якою слідує цифра1 або ж ви хочете просто використовувати посилання \\11, за якою нічого не слід . Це непорозуміння можна усунути, якщо скористатися конструкцією ${1} 1, що використовує ізольоване посилання $1, і наступну за нею цифру 1.

При використанні застарілого модифікатора e ця функція екранує деякі символи (а саме ', ", \і NULL) в рядках, що заміщають зворотні посилання. Це зроблено для посвідчення коректності синтаксису при використанні зворотних посилань всередині одинарних або подвійних лапок (наприклад,' strlen (\'$1 \') + strlen ( "$2") '). Переконайтеся, що ви володієте [синтаксисом обробки рядків](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://php.net/manual/ru/language.types.string.php&usg=ALkJrhgOQsT2EM5oA4Lvn6FxOi9Q6DaTOg) PHP для того, щоб точно усвідомлювати, як буде виглядати інтерпретований рядок.

**subject**

Рядок або масив рядків для пошуку і заміни.

Якщо **subject** є масивом, то пошук з заміною здійснюється для кожного елемента масиву **subject** , а повернуте значення також буде масивом.

**limit**

Максимально можливу кількість замін кожного шаблону для кожного рядка **subject** . За замовчуванням дорівнює-1 (без обмежень).

**count**

Якщо вказана, то ця змінна буде заповнена кількістю проведених замін.

### значення, що повертаються [¶](http://php.net/manual/ru/function.preg-replace.php#refsect1-function.preg-replace-returnvalues)

**preg\_replace ()** повертає масив, якщо параметр **subject** є масивом, інакше повертається рядок.

Якщо знайдені збіги, повертається нова версія **subject** , інакше **subject** повертається недоторканим, в разі помилки повертається **NULL** .

### помилки [¶](http://php.net/manual/ru/function.preg-replace.php#refsect1-function.preg-replace-errors)

З версії PHP 5.5.0, якщо передається модифікатор "\e", викликається помилка рівня **E\_DEPRECATED** . З версії PHP 7.0.0 в цьому випадку видається **E\_WARNING** і сам модифікатор ігнорується.

### список змін [¶](http://php.net/manual/ru/function.preg-replace.php#refsect1-function.preg-replace-changelog)

| **версія** | **опис** |
| --- | --- |
| 7.0.0 | Вилучена підтримка модифікатора / e. Замість нього використовуйте [preg\_replace\_callback ()](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://php.net/manual/ru/function.preg-replace-callback.php&usg=ALkJrhhmN54I7jbyW5egWyGnMOe6F64_rA) . |
| 5.5.0 | Модифікатор / e тепер вважається застарілим. Використовуйте функцію [preg\_replace\_callback ()](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://php.net/manual/ru/function.preg-replace-callback.php&usg=ALkJrhhmN54I7jbyW5egWyGnMOe6F64_rA) . Дивіться документацію [PREG\_REPLACE\_EVAL](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://php.net/manual/ru/reference.pcre.pattern.modifiers.php&usg=ALkJrhhjFi9lz29M2wbHoRWz09SQJwp-Jg#reference.pcre.pattern.modifiers.eval) з додатковою інформацією і описом проблем з безпекою. |
| 5.1.0 | Доданий параметр **count** |

### приклади [¶](http://php.net/manual/ru/function.preg-replace.php#refsect1-function.preg-replace-examples)

**Приклад # 1 Використання підмасок, за якими слідує цифра**

<?php   
$string = 'April 15, 2003' ;   
$pattern = '/(\w+) (\d+), (\d+)/i' ;   
$replacement = '${1}1,$3' ;   
echo preg\_replace ( $pattern , $replacement , $string );   
?>

Результат виконання даного прикладу:

April1,2003

**Приклад # 2 Використання масивів з числовими індексами в якості аргументів функції** **preg\_replace ()**

<?php   
$string = 'The quick brown fox jumps over the lazy dog.' ;   
$patterns = array();   
$patterns[0] = '/quick/' ;   
$patterns[1] = '/brown/' ;   
$patterns[2] = '/fox/' ;   
$replacements = array();   
$replacements[2] = 'bear' ;   
$replacements[1] = 'black' ;   
$replacements[0] = 'slow' ;   
echo preg\_replace ( $patterns , $replacements , $string );   
?>

Результат виконання даного прикладу:

The bear black slow jumps over the lazy dog.

Відсортувавши по ключам шаблони і заміни, отримуємо бажаний результат:

<?php   
ksort ( $patterns );   
ksort ( $replacements );   
echo preg\_replace ( $patterns , $replacements , $string );   
?>

Результат виконання даного прикладу:

The slow black bear jumps over the lazy dog.

**Приклад # 3 Заміна по декільком шаблонами**

<?php   
$patterns = array ( '/(19|20)(\d{2})-(\d{1,2})-(\d{1,2})/' ,   
'/^\s\*{(\w+)}\s\*=/' );   
$replace = array ( '\3/\4/\1\2' , '$\1 =' );   
echo preg\_replace ( $patterns , $replace , '{startDate} = 1999-5-27' );   
?>

Результат виконання даного прикладу:

$StartDate = 5/27/1999

**Приклад # 4 Чистка пробілів**

Цей приклад вичищає зайві **пробіл**и в рядку.

<?php   
$str = 'foo o' ;   
$str = preg\_replace ( '/\s\s+/' , ' ' , $str );   
// Это теперь будет 'foo o'   
echo $str ;   
?>

**Приклад № 5 Використання параметра** **count**

<?php   
$count = 0 ;   
  
echo preg\_replace (array( '/\d/' , '/\s/' ), '\*' , 'xp 4 to' , - 1 , $count );   
echo $count ; //3   
?>

Результат виконання даного прикладу:

xp \*\*\* to

3

### Примітки [¶](http://php.net/manual/ru/function.preg-replace.php#refsect1-function.preg-replace-notes)

**зауваження:**

При використанні масивів в **pattern** і **replacement** , ключі обробляються в тому порядку, в якому вони знаходяться в масиві. Цей порядок не завжди збігається з числовим порядком індексів. Якщо ви використовуєте індекси для зіставлення один з одним потрібного **pattern** і **replacement** , то вам необхідно прогнати через функцію [ksort ()](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://php.net/manual/ru/function.ksort.php&usg=ALkJrhhkC6I-7NTSNMj2d_jh2JZpbF0hWw) обидва масиви перед використанням **preg\_replace ().**

### **Функція preg\_split** Функція *preg\_split()* розбиває рядок на частини відповідно до регулярного виразу і повертає масив рядків. опис [¶](http://php.net/manual/ru/function.preg-split.php#refsect1-function.preg-split-description)

**array** preg\_split (string $pattern , string $subject[, int $limit = -1[, int $flags = 0]])

Розбиває рядок по регулярному виразу.

### список параметрів [¶](http://php.net/manual/ru/function.preg-split.php#refsect1-function.preg-split-parameters)

**pattern**

Рядок, що містить шуканий шаблон.

**subject**

Вхідний рядок.

**limit**

Якщо вказано, функція повертає не більше, ніж **limit** підрядків. Частина рядка, що залишилася буде повернута в останньому підрядку. Спеціальне значення **limit** , рівне -1, 0 або **NULL** , має на увазі відсутність обмеження, і, як фактичного стандарту в PHP, можна використовувати **NULL** для пропуску параметра **flags** .

**flags**

**flags** може бути будь-якою комбінацією наступних прапорів (об'єднаних за допомогою побітового оператора |):

**PREG\_SPLIT\_NO\_EMPTY**

Якщо вказаний цей прапор, функція **preg\_split ()** поверне тільки непусті підрядки.

**PREG\_SPLIT\_DELIM\_CAPTURE**

Якщо вказаний цей прапор, вираз, укладений в круглі дужки розділяючого шаблону, також вилучається із заданого рядка і повертається функцією.

**PREG\_SPLIT\_OFFSET\_CAPTURE**

Якщо вказаний цей прапор, для кожного знайденого підрядка буде вказана його позиція в заданому рядку. Необхідно пам'ятати, що цей прапор змінює формат повернутого масиву: кожен елемент буде містити масив, що містить в індексі з номером 0 знайдений підрядок, а зміщення цього підрядка в параметрі **subject** - в індексі 1.

### значення, що повертаються [¶](http://php.net/manual/ru/function.preg-split.php#refsect1-function.preg-split-returnvalues)

Повертає масив, що складається з підрядків заданого рядка **subject** , яка розбита на межі, відповідним шаблоном **pattern** , або **FALSE** в разі виникнення помилки.

### приклади [¶](http://php.net/manual/ru/function.preg-split.php#refsect1-function.preg-split-examples)

**Приклад # 1** **preg\_split ()** **приклад: Отримання підрядків із заданого тексту**

<?php   
// разбиваем строку по произвольному числу запятых и пробельных символов,   
// которые включают в себя " ", \r, \t, \n и \f   
$keywords = preg\_split ( "/[\s,]+/" , "hypertext language, programming" );   
print\_r ( $keywords );   
?>

Результат виконання даного прикладу:

Array

(

[0] => hypertext

[1] => language

[2] => programming

)

**Приклад # 2 Розбиваємо рядок на складові символи**

<?php   
$str = 'string' ;   
$chars = preg\_split ( '//' , $str , - 1 , PREG\_SPLIT\_NO\_EMPTY );   
print\_r ( $chars );   
?>

Результат виконання даного прикладу:

Array

(

[0] => s

[1] => t

[2] => r

[3] => i

[4] => n

[5] => g

)

**Приклад # 3 Розбиваємо рядок із зазначенням зміщення для кожного із знайдених підрядків**

<?php   
$str = 'hypertext language programming' ;   
$chars = preg\_split ( '/ /' , $str , - 1 , PREG\_SPLIT\_OFFSET\_CAPTURE );   
print\_r ( $chars );   
?>

Результат виконання даного прикладу:

Array

(

[0] => Array

(

[0] => hypertext

[1] => 0

)

[1] => Array

(

[0] => language

[1] => 10

)

[2] => Array

(

[0] => programming

[1] => 19

)

)

### Примітки [¶](http://php.net/manual/ru/function.preg-split.php#refsect1-function.preg-split-notes)

**Підказка**

Якщо вам не потрібна потужність регулярних виразів, ви можете вибрати більш швидкі (хоч і прості) альтернативи на зразок [explode ()](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://php.net/manual/ru/function.explode.php&usg=ALkJrhhG-9egABMlAQkngWGkSdr73tYIyQ) або [str\_split ()](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://php.net/manual/ru/function.str-split.php&usg=ALkJrhhrrSgnpVaY2KeHA7a81OL1S-JiKA) .

**preg\_filter**

(PHP 5> = 5.3.0, PHP 7)

preg\_filter - Здійснює пошук і заміну по регулярному виразу

### опис [¶](http://php.net/manual/ru/function.preg-filter.php#refsect1-function.preg-filter-description)

[mixed](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://php.net/manual/ru/language.pseudo-types.php&usg=ALkJrhie_lgL-HgaNsLRXVB5jbHBqBYGGw#language.types.mixed) **preg\_filter** ( [mixed](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://php.net/manual/ru/language.pseudo-types.php&usg=ALkJrhie_lgL-HgaNsLRXVB5jbHBqBYGGw#language.types.mixed) $pattern , [mixed](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://php.net/manual/ru/language.pseudo-types.php&usg=ALkJrhie_lgL-HgaNsLRXVB5jbHBqBYGGw#language.types.mixed) $replacement , [mixed](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://php.net/manual/ru/language.pseudo-types.php&usg=ALkJrhie_lgL-HgaNsLRXVB5jbHBqBYGGw#language.types.mixed) $subject[, int $limit = -1[,int &$count]])

Функція **preg\_filter ()** ідентична функції [preg\_replace ()](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://php.net/manual/ru/function.preg-replace.php&usg=ALkJrhjbXDACq4M1EQdUDbNiuLIg3mY2Cg) за винятком того, що повертає тільки ті значення (можливо, перетворені), в яких знайдено збіг.

### Значення, що повертаються [¶](http://php.net/manual/ru/function.preg-filter.php#refsect1-function.preg-filter-returnvalues)

Повертає [array](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://php.net/manual/ru/language.types.array.php&usg=ALkJrhje9L1Olf_XiBqK0p9UPoCujImQqA) , якщо аргумент **subject** має тип [array](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://php.net/manual/ru/language.types.array.php&usg=ALkJrhje9L1Olf_XiBqK0p9UPoCujImQqA) , або [string](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://php.net/manual/ru/language.types.string.php&usg=ALkJrhgOQsT2EM5oA4Lvn6FxOi9Q6DaTOg) в іншому випадку.

Якщо збігів не знайдено або виникла помилка, повертається порожній масив [array](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://php.net/manual/ru/language.types.array.php&usg=ALkJrhje9L1Olf_XiBqK0p9UPoCujImQqA), коли **subject** має тип [array](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://php.net/manual/ru/language.types.array.php&usg=ALkJrhje9L1Olf_XiBqK0p9UPoCujImQqA) , або **NULL** в іншому випадку.

### Приклади [¶](http://php.net/manual/ru/function.preg-filter.php#refsect1-function.preg-filter-examples)

**Приклад # 1 Приклад для порівняння функцій** **preg\_filter ()** **і** [preg\_replace ()](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://php.net/manual/ru/function.preg-replace.php&usg=ALkJrhjbXDACq4M1EQdUDbNiuLIg3mY2Cg)

<?php   
$subject = array( '1' , 'а' , '2' , 'б' , '3' , 'А' , 'Б' , '4' );   
$pattern = array( '/\d/' , '/[а-я]/' , '/[1а]/' );   
$replace = array( 'А:$0' , 'Б:$0' , 'В:$0' );   
  
echo "preg\_filter возвращает\n" ;   
print\_r ( preg\_filter ( $pattern , $replace , $subject ));   
  
echo "preg\_replace возвращает\n" ;   
print\_r ( preg\_replace ( $pattern , $replace , $subject ));   
?>

Результат виконання даного прикладу:

preg\_filter повертає

Array

(

[0] => А: В: 1

[1] => Б: В: а

[2] => А: 2

[3] => Б: б

[4] => А: 3

[7] => А: 4

)

preg\_replace повертає

Array

(

[0] => А: В: 1

[1] => Б: В: а

[2] => А: 2

[3] => Б: б

[4] => А: 3

[5] => А

[6] => Б

[7] => А: 4

)

# preg\_grep

(PHP 4, PHP 5, PHP 7)

preg\_grep - Повертає масив входжень, які відповідають шаблону

### опис [¶](http://php.net/manual/ru/function.preg-grep.php#refsect1-function.preg-grep-description)

**array** preg\_grep (string $pattern , array $input[, int $flags = 0])

Повертає масив, що складається з елементів вхідного масиву **input** , які відповідають заданоу шаблону **pattern** .

### список параметрів [¶](http://php.net/manual/ru/function.preg-grep.php#refsect1-function.preg-grep-parameters)

**pattern**

Шуканий шаблон, рядок.

**input**

Вхідний масив.

**flags**

У разі, якщо встановлено в **PREG\_GREP\_INVERT** , функція **preg\_grep ()** повертає ті елементи масиву, які не відповідають заданому шаблону **pattern** .

### значення, що повертаються [¶](http://php.net/manual/ru/function.preg-grep.php#refsect1-function.preg-grep-returnvalues)

Повертає масив, індексований ключами з масиву **input** .

### приклади [¶](http://php.net/manual/ru/function.preg-grep.php#refsect1-function.preg-grep-examples)

**Приклад # 1 Приклад використання** **preg\_grep ()**

<?php   
// Возвращает все элементы массива,   
// содержащие числа с плавающей точкой   
$fl\_array = preg\_grep ( "/^(\d+)?\.\d+$/" , $array );   
?>

# preg\_quote

(PHP 4, PHP 5, PHP 7)

preg\_quote - екранують символи в регулярних виразах

### опис [¶](http://php.net/manual/ru/function.preg-quote.php#refsect1-function.preg-quote-description)

**string** preg\_quote (string $str[, string $delimiter = NULL])

Функція **preg\_quote ()** приймає рядок **str** і додає зворотний слеш перед кожним службовим символом. Це буває корисно, якщо в складанні шаблону беруть участь рядкові змінні, значення яких в процесі роботи скрипта може змінюватися.

У регулярних виразах службовими вважаються наступні символи:. \+ \*?[^] $() {} =! <> | : -

Зверніть увагу, що символ / не є службовим.

**зауваження:**

Зверніть увагу, що **preg\_quote ()** не призначена для застосування до рядків $replacement функції [preg\_replace ()](https://translate.googleusercontent.com/translate_c?depth=1&hl=ru&rurl=translate.google.com.ua&sl=ru&tl=uk&u=http://php.net/manual/ru/function.preg-replace.php&usg=ALkJrhjbXDACq4M1EQdUDbNiuLIg3mY2Cg) і т.д.

### список параметрів [¶](http://php.net/manual/ru/function.preg-quote.php#refsect1-function.preg-quote-parameters)

**str**

Вхідний рядок.

**delimiter**

У разі, якщо зазначений необов'язковий параметр **delimiter** , він буде також екрануватися. Це зручно для екранування роздільника, який використовується в PCRE функціях. Найбільш поширеним роздільником є символ /.

### значення, що повертаються [¶](http://php.net/manual/ru/function.preg-quote.php#refsect1-function.preg-quote-returnvalues)

Повертає екранований рядок.

### список змін [¶](http://php.net/manual/ru/function.preg-quote.php#refsect1-function.preg-quote-changelog)

| **версія** | **опис** |
| --- | --- |
| 5.3.0 | Символ - тепер екранується |

### приклади [¶](http://php.net/manual/ru/function.preg-quote.php#refsect1-function.preg-quote-examples)

**Приклад # 1** **preg\_quote ()** **приклад**

<?php   
$keywords = '$40 за g3/400' ;   
$keywords = preg\_quote ( $keywords , '/' );   
echo $keywords ; // возвращает \$40 за g3\/400   
?>

**Приклад # 2 Виділення курсивом слова в тексті**

<?php   
// В данном примере preg\_quote($word) используется, чтобы   
// избежать трактовки символа '\*' как спец. символа.   
  
$textbody = "Эту книгу \*очень\* тяжело найти." ;   
$word = "\*очень\*" ;   
$textbody = preg\_replace ( "/" . preg\_quote ( $word , '/' ) . "/" ,   
"<i>" . $word . "</i>" ,   
$textbody );   
?>

### Примітки [¶](http://php.net/manual/ru/function.preg-quote.php#refsect1-function.preg-quote-notes)

**Зауваження:** Ця функція безпечна для обробки даних в двійковій формі.

**Хід роботи**

1. Перевірте довільний рядок, чи він є правильною e-mail адресою. Використайте функцію preg\_match().

2. Задано текст ( файл text.txt). Виведіть з нього на веб-сторінку:

1. весь заданий текст;
2. тільки назви відкриваючих тегів;
3. назви відкриваючих тегів разом і кутовими дужками;

*Примітка: теги повинні виводитися як текст і не впливати на форматування веб-сторінки.*

3. Перевірте телефонні номери на відповідність шаблону +38 (097) 65-94-888.

4. Є масив 10 дат у різних форматах (рррр-мм-дд, дд-мм-рррр). Використовуючи оператори циклу і розгалуження, перевірте на відповідністі одному із двох форматів дати і приведіть їх до формату мм.дд.рррр за допомогою функції preg\_replace().

5. У тексті ( файл text.txt), використовуючи регулярні вирази, знайти всі входження слова незалежно від регістру (в тому числі і з усіма закінченнями):

1. тег;
2. HTML;
3. слова, введеного в однорядкове текстове поле в називному відмінку.

Вивести тільки ті речення, в яких дане слово присутнє, в порядку спадання числа знайдених відповідностей шаблону пошуку (першим повинно йти речення, в якому шукане слово зустрічається найчастіше), знайдені слова виділити жирним шрифтом.

**Контрольні запитання**

1. Назвіть відомі Вам формати регулярних виразів.

2. Поясніть використання функцій preg\_grep(), preg\_match(), preg\_replace().

3. Який пошук здійснюватиме вираз:

* “/^\w{6,9}\d {3}/s”;
* “/ \d{3}-\d{2}-\d{2}/m”?

4. Що вказують за допомогою квантифікаторів?

5. Що вказують в параметрі модифікатор?