Лабораторна робота №2

Тема: «Оператори умовного переходу. Оператори циклів»

Мета: Набуття навичок в використанні конструкцій розгалуження в php-скриптах.

Теоретичні відомості

До операторів умовного переходу відносять: оператор **(if...else)** і перемикач **(switch)**. Синтаксис умовного оператора:

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

Умова **condition** може бути будь-яким виразом. Якщо вона є істиною, то виконується оператор **statement 1**. Інакше виконується оператор **statement 2.** Допускається скорочена форма запису умовного оператора, в якій відсутні оператор **else** і оператор **statement 2.**

У свою чергу, оператори **statement 1** і **statement 2** можуть бути умовними, що дозволяє організовувати ланцюжки перевірок будь-якої глибини вкладеності. І в цих ланцюжках кожен умовний оператор може бути як повним, так і скороченим.

Синтаксис мови припускає, що при вкладених умовних операторах кожне **else** відповідає найближчому **if**. У зв'язку з цим можливі помилки неоднозначного зіставлення **if** і **else**.

Простим правильним рішенням цієї задачі є застосування фігурних дужок, тобто нам потрібно фігурними дужками обмежити область дії внутрішнього умовного оператора, зробивши його неповним. Тим самим зовнішній оператор перетворюється на повний умовний:

```
<?
    $x = 1;
    $y = 1;
    if($x==1)
    {
        if($y==1)echo("x=1 and y=1");
    }
    else echo("x!=1");
?>
```

Додаткові умови можливо перевірити за допомогою оператора **elseif.** Оператор **if** може включати скільки завгодно блоків **elseif**, але **else** в кожному **if** може бути тільки один. Як правило, в конструкціях **if**…**elseif**…**else** оператор **else** визначає, що потрібно робити, якщо ніякі інші умови не виконуються.

Наприклад:

Магазин надає знижки при замовленні великої кількості автопокришок. Схема знижок виглядає таким чином:

• Придбання менше 10 автопокришок - без знижки
• Придбання 10-49 автопокришок - знижка 5%
• Придбання 50-99 автопокришок - знижка 10%
• Придбання 100 і більш за автопокришки - знижка 15%

Можна створити код для обчислення знижок з використанням умов і операторів **if** і **elseif**. Для об'єднання двох умов в одне застосовується операція і (**&&**). Значення змінної **$tireqty** вводиться за допомогою форми (розробити самостійно)

```
<?php

$z="<b>Знижка дорівнює</b>";
$n="<b>Без знижки </b>";

If  ( $tireqty >= 10 && $tireqty <= 49 ) {
$discount = 5;
echo "$z  $discount<br> ";}

elseif ( $tireqty >= 50 && $tireqty <= 99 ) {
$discount = 10;
echo "$z  $discount<br> ";}

elseif( $tireqty >= 100 ){
$discount = 15;
echo "$z  $discount<br> ";  }

else
   echo "$n";

?>
```

Зверніть увагу, що можна застосовувати як **elseif**, так і **else if** - обидва варіанти правильні.

При використанні каскадних наборів операторів elseif слід пам'ятати, що буде виконуватися тільки один з блоків операторів. У даному прикладі це не важливо оскільки всі умови є взаємовиключними - в кожен момент часу може виконуватися тільки один з них. Якби умови були записані так, що одночасно могло б виконуватися декілька умов, виконувався б тільки блок, або оператор, наступний за першою дійсною умовою.

Проте, використання оператора **elseif** досить сильно погіршує читабельність коду, і краще в цьому випадку користуватись перемикачем **switch**, який призначений для аналізу множинних умов.

PHP надає також можливість альтернативного синтаксису умовного оператора – без фігурних дужок - із застосуванням оператора **endif.**

Наприклад :

if(вираз): блок_виконання **endif**;

Має сенс, якщо умова, записана в круглих дужках оператора **if**, виявилася істиною, виконуватиметься ваш код, від двокрапки «**:**» до команди **endif**;. Використання такого синтаксису корисне при вбудовуванні php в html-код.

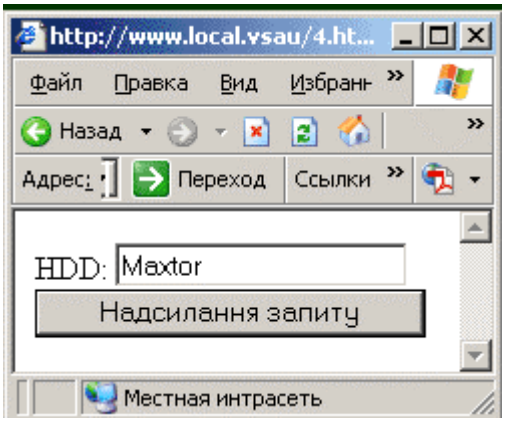
```
<?php
$names = array("Іван","Петр","Семен");
if ($names[0]=="Іван"):
?>
Привіт, Ваня!
<?php endif ?>
```

У наступному прикладі скрипт аналізує зміст змінної **$HDD.** Якщо в ній буде стрічка «Maxtor», то буде генеруватись WEB-сторінка з таблицею і заголовком «**Maxtor**». В протилежному випадку – це буде таблиця із заголовком «**Seagate».** Наявність оператора **endif** в цьому випадку обов'язкова, оскільки фігурна дужка, що позначає кінець блоку **if**, відсутня:

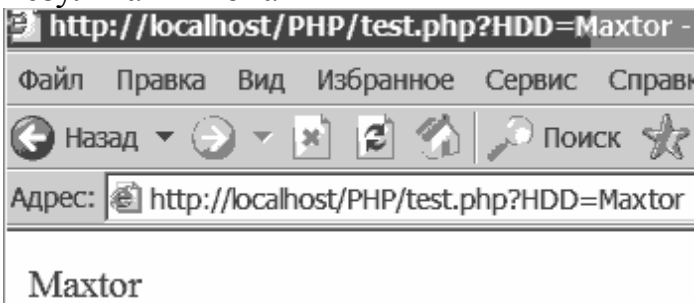
```
<?
   if($HDD == «Maxtor»):
?>
<table>
    <caption> Maxtor </caption>
</table>
<?
   elseif($HDD == «Seagate»):
?>
<table>
   <caption> Seagate </caption>
</table>
<?
   endif;
?>
```

Результат виконання скрипта

Форма запиту



Результат виконання



PHP надає можливість замінювати блоки **if…else** умовною операцією. У зображенні умовної операції присутні два розміщених не підряд символи ‘**?**’ і ‘**:**’ і три операнди виразу:

```
вираз_1 ? вираз_2 : вираз_3
```

Першим обчислюється значення **виразу_1.** Якщо воно є істинним (тобто не дорівнює нулю), то обчислюється значення **вираз**у_2, яке і стає результатом. Якщо при обчисленні значення **вираз_1** буде дорівнювати нулю (хибність), то як результат буде дорівнювати **виразу**_3.

Таким чином, наприклад, код

```
<?
    if($HDD == "Maxtor")
    {
        $CDROM = "Teac";
    }
    else
    {
        $CDROM = "Nec";
    }
?>
```

можна записати таким чином:

```
<?
   $CDROM = ($HDD == "Maxtor") ? "Teac" : "Nec";
?>
```

Перемикач switch

Перемикач **switch** є найзручнішим засобом для організації розгалуження із множинними варіантами умов. Синтаксис перемикача такий:

```
switch(expression) // перемикаючий вираз
{
   case value1: // константний вираз 1
      statements; // блок операторів
   break;
   case value2: // константний вираз 2
      statements;
   break;
   default:     // інакше ...
      statements;
}
```

Управляюча структура **switch** передає управління тому з помічених **case** операторів, для якого значення константного виразу співпадає із значенням перемикаючого виразу. Якщо значення перемикаючого виразу не співпадає ні з одним з константних виразів, то виконується перехід до оператора, поміченого міткою **default**. У кожному перемикачі може бути не більше однієї мітки **default** (проте вона може бути і взагалі відсутньою).

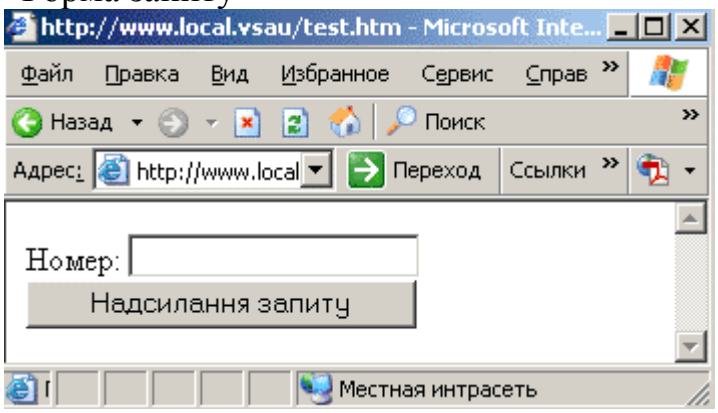
Наведемо приклад програми з перемикачем. У цій програмі виводяться назви непарних цілих десяткових цифр від 1 до 9 не менше заданого, залежно від числа, вказаного у формі test.html. Форма **test.html** не відрізняється від тієї, що ми вже використовували:

```
<?
switch($number)
{
     case 1:
       echo ("one ");
     case 2: case 3:
       echo ("free");
     case 4: case 5:
       echo ("five");
     case 6: case 7:
       echo ("seven");
```

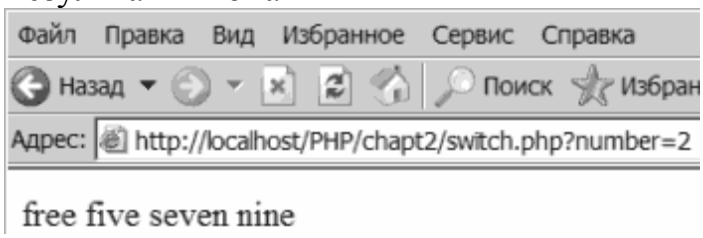
```
    case 8: case 9:
      echo ("nine");
    break;
    default:
      echo ("Ці номери не потрапляють у заданий
діапазон > 9 or < 1");
}
?>
```

Результат виконання скрипту

Форма запиту



Результат виконання



Для перемикачів допустимі будь-які ступені вкладеності, проте зловживати цим без особливої на те необхідності також не слід.

Приведена програма демонструє дію оператора **break**, за допомогою якого відбувається вихід з перемикача. Якщо помістити оператори **break** після виводу кожної з цифр, то у вікні браузера ми побачимо назву тільки однієї непарної цифри.

Оператори циклу

- цикл з передумовою:

```
while(condition) {statements;}
```

- цикл з постумовою:

```
do{statements;} while(condition);
```

- ітераційний цикл:
  ```
  for(expression1;expression2;expression3)
  {statements;}
  ```
- ітераційний цикл foreach:
  ```
  foreach (array as [$key =>] $value)
  {statements;}
  ```

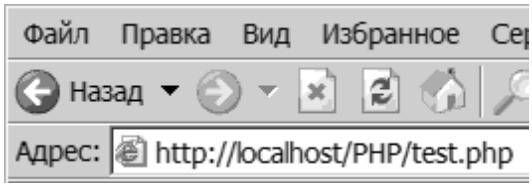
Оператор циклу/ While

Оператор **while** називається оператором циклу з передумовою. При вході в цикл обчислюється умова, і, якщо його значення відмінне від нуля, виконується тіло циклу. Потім обчислення умови і операторів тіла циклу виконується до тих пір, поки значення виразу умови не стане рівним нулю. Оператором **while** зручно користуватися для проглядання всіляких послідовностей, якщо в кінці їх знаходиться наперед відомий символ.

Приклад простого циклу **while**:

```
<?
    $var = 5;
    $i = 0;
    while(++$i <= $var)
    {
        echo($i); echo('<br>');
    }
?>
```

Цей код видає у вікні браузера цифри від одного до п'яти:



1
2
3
4
5

Для виходу з циклу застосовується оператор **break.** При виявленні цього оператора поточна ітерація циклу припиняється, і подальші ітерації не

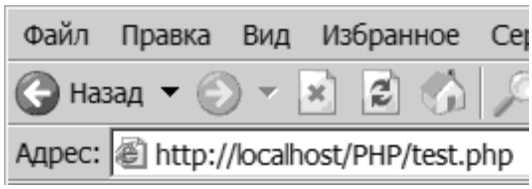
відбуваються. При виконанні наступного прикладу, не зважаючи на те, що змінна $var = 7, у вікні браузера з'являться цифри від 1 до 5.

```
<?
    $var = 7;
    $i = 0;
    while(++$i <= $var)
    {
        echo($i);
        echo('<br>');
        if($i==3)break;
    }
?>
```

Іноді буває потрібно перервати тільки поточну ітерацію, і перейти відразу до наступної. Для цього застосовується оператор **continue:**

```
<?
    $var = 7;
    $i = 0;
    while(++$i <= $var)
    {
        if($i==5)
        {
            continue;
        }
        echo($i);
        echo('<br>');
    }
?>
```

В даному прикладі виводяться цифри від 1 до 7, окрім цифри 5:



1
2
3
4
6
7

Помітимо, що якщо Ви умовного оператора поставите після операторів **echo**, код буде помилковим, і виведуться всі цифри від 1 до 20,

оскільки перевірка умови виходу з циклу на даній ітерації, відбуватиметься вже після виконання цієї ітерації.

Нескінченний цикл реалізується за допомогою оператора **while** таким чином:

```
while(1)
    {
        ...
    }
```

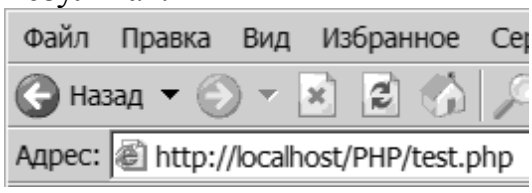
Це теж саме, що і запис **while(**true).

Оператори циклу/ Do…while

Цей оператор називається оператором циклу з постумовою. При вході в цикл у будь-якому випадку виконується тіло циклу (тобто цикл завжди буде виконаний хоча б один раз), потім обчислюється умова, і якщо вона не рівна 0, знов виконується тіло циклу. У нижченаведеному прикладі нуль завжди буде доданий в список, незалежно від умови (++**$i &**lt;= $var):

```
<?
    $var = 5;
    $i = 0;
    do
    {
       echo($i); echo('<br>');
    }
    while(++$i <= $var)
?>
```

Результат:



1
2
3
4
5

Цикл з післяумовою буває корисний при обробці деяких послідовностей, коли обробку потрібно закінчувати не до, а після появи кінцевої ознаки.

Нескінченний цикл реалізується так:

```
do; while(1);
```

Оператори циклу/ For

Ітераційний цикл має наступний формат:

```
for(expression1;expression2;expression3)
{
      statements;
}
```

Тут **expression1 (**ініціалізація циклу) – послідовність визначень і виразів, що розділяється комами. Всі вирази, що входять в ініціалізацію, обчислюються тільки один раз при вході в цикл. Як правило, тут встановлюються початкові значення лічильників і параметрів циклу. Значення виразу-умови (**expression2**) такої ж як і у циклів з пред- і постумовами. За відсутності виразу-умови передбачається, що його значення завжди істинне. Вирази **expression3** обчислюються в кінці кожної ітерації після виконання тіла циклу.

У наступному скрипті, ми за традицією виведемо числа від 0 до 5:

```
<?
    $var = 5;
    $i = 0;
    for ($i = 0; $i <= $var; $i++)
    {
        echo($i);
        echo('<br>');
    }
?>
```

Результат аналогічний, показаному на попередньому малюнку.

Нескінченний цикл можна організувати таким чином:

```
for(;;);
   або
for(;1;);
```

Функція `isset()`

```
isset(variable);
```

Функція `isset()` – корисне доповнення до оператору **`if`**, яка дозволяє визначити, чи встановлено значення змінної. Наприклад, перевірка введення користувачем інформації у формі. Також можна написати простий оператор **`if`** для контролю за тим, чи натиснув користувач кнопку `Submit`. Якщо змінна `$Submit` була встановлена, то можна приступити до обробки даних з форми. Якщо ж змінна `$Submit` не була встановлена, це означає, що

користувач ще не предоставив інформацію у формі і ви повинні показати цю форму в браузері, що відвідувач вузла міг ввести дані.

```
If (isset ($submit)):
//Виконати якісь дії
Else:
//Вивести форму
Endif;
```

З функцією `isset()` можливо також використовувати оператор «НІ» (!) для перевірки того, чи заповнені всі поля форми. Наприклад:

```
If (isset ($phone_nambe)):
      Echo "Ви не ввели свій номер телефону!\n";
Endif;
```

Створення користувацьких функцій.

Коли ми здійснюємо дії, в яких простежується залежність від якихось даних, і при цьому, можливо, нам знадобиться виконувати такі ж дії, але з іншими початковими даними, зручно використовувати механізм функцій - оформити блок дій у вигляді тіла функції, а змінні дані - як її параметри.

Щоб створити свою функцію достатньо надати ім'я функції, перерахувати параметри, які їй передаються та записати оператори, з яких вона буде створена. Зазвичай параметрами функції є дані, які їй треба обробити. Також потрібно оператори функції взяти у фігурні дужки.

```
Function name (arg1, arg2, arg3, …) {
Оператори;
}
```

Приклад об'явлення функції:

```
Function name_print ($name) {
Echo "<p>Ім'я:<b> $name</b>";
}
```

Їй передається один параметр – змінна `$name`, значення якої і виводиться в браузер.

Щоб викликати цю функцію, достатньо вказати її ім'я та передати ту ж кількість параметрів, яку була означено при її створенні. Вдпнному прикладі передається тільки один параметр.

Значимість користувацьких функцій у тому, що економиться час при створенні сценаріїв. Замість того, щоб кожний раз, коли треба ввести ім'я у початковому тексті замість оператора

```
Echo "<p>Ім'я:<b>$name</b>";
```

достатньо ввести `name_print ($name);`

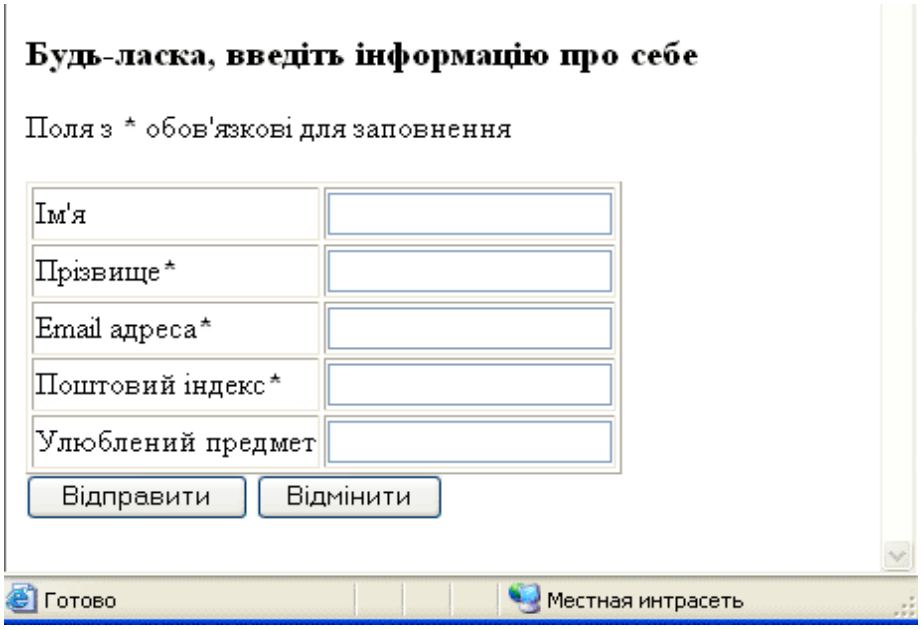
Хід виконання:

Завдання №1

Відтворити приклади наведенні у лабораторній роботі.

Завдання №2

Розробити сценарій, який буде перевіряти чи заповненні обов'язкові поля форми. Форма буде збирати інформацію про користувача, а саме: ім'я, прізвище, адресу електронної скриньки, поштовий код та назву улюбленого предмету (рис. 1).



Для цього потрібно зробити наступне:

1. Створити три функції. Перша - `print_form` просто виводить форму на екран. Якщо деякі поля вже були заповнені, функція автоматично вставляє в них належні значення. Це зручно, так як користувачу не потрібно знову і знову вводити одну й ту ж саму інформацію тільки через те, що він забув заповнити одне з обов'язкових полів.
2. Друга функція - `check_form` перевіряє заповнення обов'язкових полів. Якщо якесь з них не заповнено, функція повідомляє про це користувачу, а потім викликає `print_form` для повторного відображення форми (рис. 2).

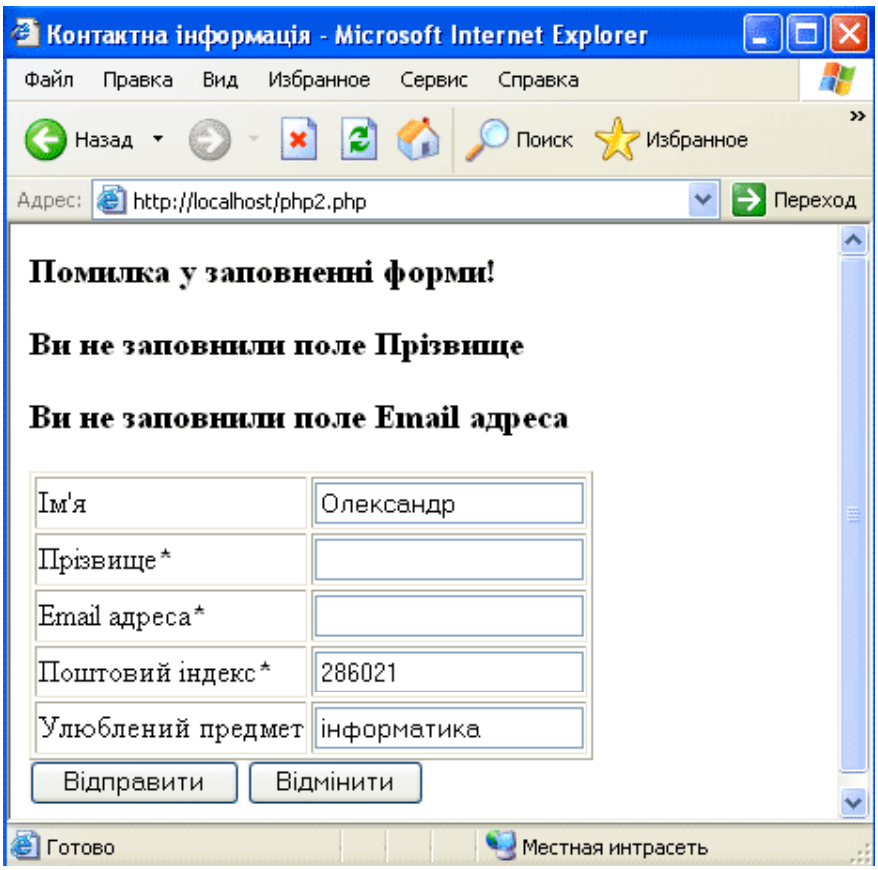


Рис. 2. Результат роботи сценарію, якщо у формі були знайдені помилки

3. Третя функція `confirm_form`, просто відображає введену користувачем інформацію (рис. 3).

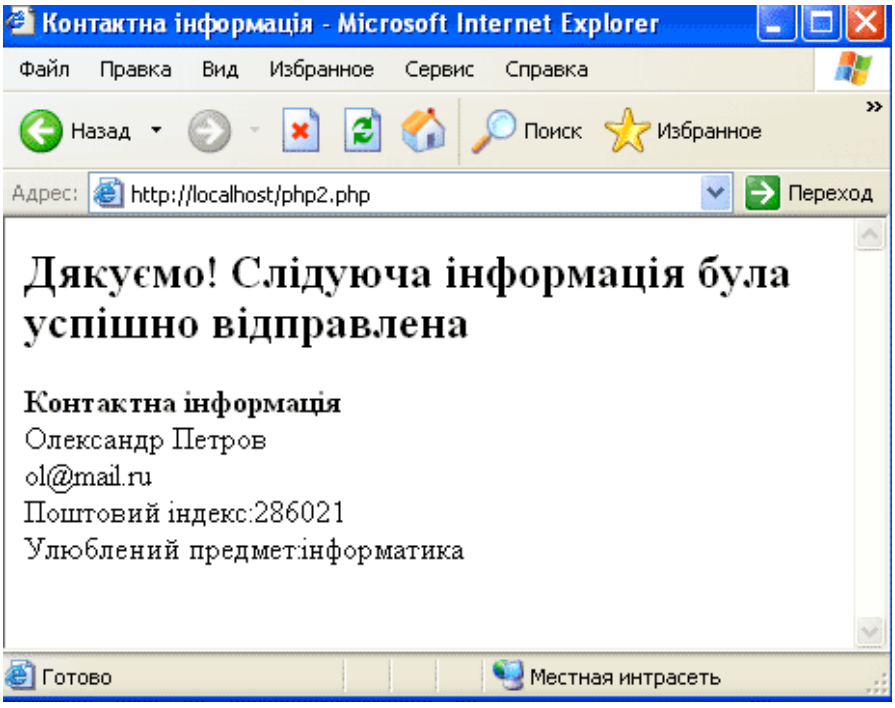


Рис. 3. Результат роботи сценарію, якщо інформація представлена правильно

А це текст самого сценарію:

```
<html>
<head>
<title>Контактна інформація</title>
</head>
<body>
<?php

/* Об'являємо деякі функції*/

function print_form ($f_name, $l_name, $email, $zip,
$object){
     ?>
     <form  action="form_checker.php" method="post">
     <table cellspasing="2" cellpedding="2" border="1">
     <tr>
          <td>Ім'я</td><td><input name="f_name" type="text"
          value="<?php echo $f_name ?>"></td>
     </tr>
     <tr>
          <td>Прізвище<b>*</b></td><td><input name="l_name"
          type="text" value="<?php print $l_name ?>"></td>
     </tr>
     <tr>
          <td>Email адреса<b>*</b></td><td><input
name="email" type="text" value="<?php print $email ?>">
          </td>
     </tr>

     <tr>
          <td>Поштовий індекс<b>*</b></td>
          <td><input name="zip"
               type="text" value="
               <?php print  $zip ?>">
          </td>
     </tr>

     <tr>
          <td>Улюблений предмет</td>
          <td><input name="object" type="text" value="
               <?php print  $object ?>">
          </td>
     </tr>
     </table>

     <input name="submit" type="submit" value="Надіслати">
     <input type="reset" value="Відмінити">
     </form>
<?
}
```

```
function check_form ($f_name, $l_name, $email, $zip,
$object){
     if (!$l_name||!$email||!$zip):echo "<h3>Помилка у
заповненні форми!</h3>";
     if (!$l_name){
          echo "<h3>Ви не заповнили поле
<b>Прізвище</b></h3>";
     }

     if (!$email){
          echo "<h3>Ви не заповнили поле <b>Email
адреса</b></h3>";
     }

     if (!$zip){
          echo "<h3>Ви не заповнили поле <b>Поштовий
індекс</b></h3>";
     }

     print_form ($f_name, $l_name, $email, $zip, $object);
     else:
     confirm_form ($f_name, $l_name, $email, $zip, $object);
     endif;
}

function confirm_form ($f_name, $l_name, $email, $zip,
$object){
     ?>
     <h2>Дякуємо! Слідуюча інформація була успішно надіслана
     </h2>
     <b>Контактна інформація</b>
     <?
     echo "<br>$f_name $l_name<br>$email<br>Поштовий
     індекс:$zip<br>Улюблений предмет:$object\n";
}
/* Початок основної програми*/

if (!$submit):
?>
<h3>Будь-ласка, введіть інформацію про себе</h3>
Поля з <b>*</b> обов'язкові для заповнення<p>
<?php
print_form("","","","","","");
else:
check_form($f_name, $l_name, $email, $zip, $object);
endif;
?>

</body>

</html>
```

Контрольні запитання

1. Які оператори умовного переходу ви знаєте?
2. Як працює оператор **(if...else)**?
3. Для чого використовують **`elseif`**?
4. Яка операція застосовується для об'єднання двох умов в одне?
5. Як можна замінювати блоки **if…else** умовною операцією?
6. Для чого використовують перемикач **(switch)**?
7. Які оператори циклу ви знаєте?
8. Як працює цикл з передумовою?
9. Як працює цикл з постумовою?
10. Як працює ітераційний цикл?
11. Як працює функція **isset**?
12. Що таке користувацька функція? Наведіть приклад.
13. Як об’явити користувацька функцію?
14. Які користувацькі функції ви використовуєте в 2 завданні?
15. Для чого створена функція **`function print_form`**?
16. Для чого створена функція **`function check_form`**?
17. Для чого створена функція **`function confirm_form`?**