Лабораторна робота №10

Тема: Робота з циклами та масивами, обробка даних форм в PHP

0. Ознайомитись з теоретичними відомостями на сайті:

Завдання на лабораторну роботу

1. З кожного завдання вибрати та виконати по 2 підзавдання на власний смак.

Завданння 1. Використання циклів в PHP

Завдання 1.1

Хтось кладе в банк 10000 грн. Банк нараховує 10% річних (тобто, кожен рік на рахунку стає на 10% більше, ніж в минулому році). Написати програму, яка обраховує, через скільки років в банку буде мільйон? Скільки років буде цьому хтось? Чи доживе хтось до цього дня, якщо сьогодні йому 18 років?

Завдання 1.2

Дано тризначне число. Знайдіть:

1) суму його цифр.
2) Знайдіть число, отримане виписуванням в зворотному порядку цифр даного тризначного натурального числа.
3) Переставте цифри так, щоб нове число виявилося найбільшим з можливих.

Завдання 1.3

Використовуючи цикли **for, while, do-while,** виведіть у стовпчик непарні числа від 1 до 50 (3 способи)

Завдання 1.4

Створіть таблицю множення у вигляді HTML-таблиці за такими умовами:

1) число стовпців повинно дорівнювати значенню змінної 1
2) число рядків повинно дорівнювати значенню змінної 2
3) комірки на перетині стовпців та рядків повинні містити значення, що є добутком порядкових номерів стовпця і рядка
4) значення в комірках першого рядка і першого стовпця повинні бути полужирним шрифтом і вирівняні по центру
5) фоновий колір комірок першого рядка і першого стовпця повинен відрізнятися від фонового кольору таблиці

Завдання 2. Використання циклів і масивів в PHP

Завдання 2.1

Вивести на екран квадрат, що складається з n x n квадратів різного кольору.

Завдання 2.2

Вивести на чорному тлі n червоних квадратів випадкового розміру в випадковій позиції в браузері.

Завдання 3.1

Вивести елементи масиву, що повторюються.

Завдання 4.1

Виведіть дане число, яке не перевищує 1000000, прописом (наприклад, 2134 - дві тисячі сто тридцять чотири).

Завдання 5.1

Написати генератор випадкового імені для кішки, собаки, брата або сестри.

Для цього можна створити масив зі складами і кілька разів вибрати з нього випадковий елемент. Ці елементи зібрати по шматочках в змінну $ name і в кінці вивести. Можна використати функцію ***array_rand (),*** *яка вибирає випадковий індекс з масиву.*

Завдання 3. Створення функцій та обробка форм в PHP

Завдання

1) Створіть за допомогою власних функцій таблицю обчислень. Дані вводьте за допомогою форми:

x^y	x!	my_tg(x)	sin(x)	cos(x)	tg(x)
64	24	1.15782128235	-0.756802495308	-0.653643620864	1.15782128235

x: 4 y: 3 =

де sin, cos, tg – за допомогою вбудованих функцій,
my_tg(x), x^y, x! – власні функції.

(Для цього створити папку, наприклад, Function з файлом func.php, в якому створити всі 6 функцій.
Потім цей файл підключити за допомогою конструкції **require_once** "Function/func.php";)

2) Додати ще комірки для знаходження суми, різниці, множення, ділення та середнього арифметичного двох чисел.

2. Створіть репозиторій IPLab10. Закомітьте у нього виконані завдання.
3. Надайте доступ до репозиторію своїм викладачам.

Методичні рекомендації:

1. Обробка форм

Теги **<form>** і **</ form>** задають початок і кінець форми. Початківець формує тег <form>, що містить два атрибути: **action** і **method**. Атрибут **action** містить адресу URL сценарію,

який повинен бути викликаний для обробки сценарію. Атрибут **method** вказує браузеру, який вид HTTP запиту необхідно використовувати для відправки форми; можливі значення **POST** і **GET**.

Зауваження:
Головна відмінність методів POST і GET полягає в способі передачі інформації. У методі GET параметри передаються через адресний рядок, тобто по суті в HTTP-заголовку запиту, в той час як в методі POST параметри передаються через тіло HTTP-запиту і ніяк не відображаються на вигляді адресного рядка.

<Form method = "post" action = "../ admin / add_story.php">

</ Form>

1) Прапорець (checkbox)

Прапорці checkbox пропоную користувачеві ряд варіантів, і дозволяє вибір кількох з них.

<Input name = "Ім'я перемикача" type = "Тип" value = "Значення">
Група прапорців складається з елементів <input>, що мають однакові атрибути name і type (checkbox). Якщо ви хочете, щоб елемент був відзначений за замовчуванням необхідно позначити його як checked. Якщо елемент обраний, то сценарієм надійде рядок ім'я = значення, в іншому випадку в обробник форми не прийде нічого, тобто не вибрано прапорці взагалі ніяк не проявляють себе в переданому наборі даних.

приклад:

<Input name = "mycolor" type = "checkbox" value = "red" checked> Червоний (обраний за замовчуванням)
<Input name = "mycolor" type = "checkbox" value = "blue"> Синій
<Input name = "mycolor" type = "checkbox" value = "black"> Чорний
<Input name = "mycolor" type = "checkbox" value = "white"> Білий

2) Перемикач (radio)

Перемикачі radio пропонують користувачеві ряд варіантів, але дозволяє вибрати тільки один з них.

<Input name = "Ім'я перемикача" type = "Тип" value = "Значення">
Перемикач (radio) має атрибути name, type та value. Атрибут name задає ім'я перемикача, type задає тип radio, а атрибут value задає значення. Якщо користувач вибере перемикач, то сценарієм буде передана рядок ім'я = значення. При необхідності можна вказати параметр checked, який вказує на те, що перемикач буде мати фокус (тобто буде відзначений за замовчуванням) при завантаженні сторінки. Перемикачі також можна об'єднувати в групи, для цього вони повинні мати один і той же ім'я.

приклад:

<Input name = "mycolor" type = "radio" value = "white"> Білий

<Input name = "mycolor" type = "radio" value = "green" checked> Зелений (обраний за замовчуванням)
<Input name = "mycolor" type = "radio" value = "blue"> Синій
<Input name = "mycolor" type = "radio" value = "red"> Червоний
<Input name = "mycolor" type = "radio" value = "black"> Чорний

3) Кнопка скидання форми (Reset)

<Input type = "Тип" name = "Ім'я кнопки" value = "Напис на кнопці">
При натисканні на кнопку скидання (reset), всі елементи форми будуть встановлені в той стан, який було задано в атрибутах за замовчуванням, причому відправка форми не проводитися.

приклад:

<Input type = "reset" name = "Reset" value = "Очистити форму">

4) Список, що випадає (select)

Тег <select> представляє собою список, що випадає або розкритий список, при цьому одночасно можуть бути обрані одна або кілька рядків.

Список починається з парних тегів <select> </ select>. Теги <option> </ option> дозволяють визначити вміст списку, а параметр value визначає значення рядка. Якщо в тезі <option> вказано параметр selected, то рядок буде спочатку обраної. Параметр size задає, скільки рядків буде займати список. Якщо size дорівнює 1, то список буде випадає. Якщо зазначений атрибут multiple, то дозволено вибирати кілька елементів зі списку (при size = 1 не має сенсу).

<Select name = "Ім'я списку" size = "Розмір" multiple>
<Option value = "Значення"> Текст, що відображається в списку </ option>
</ Select>
При передачі даних списку сценарієм передається рядок ім'я = значення, а при розкритому списку передається рядок ім'я = значення1 & ім'я = значення2 & ім'я = значення N.

5) Текстове поле (text)

Дозволяє користувачам вводити різну інформацію.

<Input type = "Тип" name = "Ім'я поля" size = "Розмір" maxlength = "Макс. Кількість символів">
При створенні звичайного текстового поля розміром size і максимальної допустимої довжини maxlength символів, атрибут type приймає значення text. Якщо вказано параметр value, то поле буде містити відображати value-текст. При створенні поля не забувайте вказувати ім'я поля, тому що цей атрибут є обов'язковим.

приклад:

<Input type = "text" name = "txtName" size = "10" maxlength = "5" value = "Текст за замовчуванням">

6) Поле для введення пароля (password)

Повністю аналогічний текстовому полю, за винятком того що символи, що набираються користувачем, не відображатимуться на екрані.

приклад:

<Input type = "password" name = "txtName" size = "10" maxlength = "5">

7) багаторядкове поле введення тексту (textarea)

Багаторядкове поле введення тексту дозволяє відправляти не одну рядок, а відразу декілька. За замовчуванням тег створює пусте поле шириною в 20 символів і складається з двох рядків.

<Textarea name = "Ім'я поля" cols = "Ширина поля" rows = "Число рядків"> Текст </ textarea>
Багаторядкове поле введення тексту починається з парних тегів <textarea> </ textarea>. Тег name задає ім'я багаторядкового поля. Також можна вказати ширину поля (cols) і число рядків (rows). При необхідності можна вказати атрибут readonly, який забороняє редагувати, видаляти і змінювати текст, тобто текст буде призначений тільки для читання. Якщо необхідно щоб текст був спочатку відображений в багаторядковому поле введення, то його необхідно помістити між тегами <textarea> </ textarea>.

приклад:

<Textarea name = "txtArea" cols = "15" rows = "10" readonly>
Текст, який спочатку буде відображений в багаторядковому поле введення і який не можна змінювати, тому що вказано атрибут readonly </ textarea>

8) Приховане текстове поле

Дозволяє передавати сценарієм якусь службову інформацію, не відображаючи її на сторінці.

<Input name = "Ім'я" type = "Тип" value = "Значення">
Приховане поле починається з тега <input>, атрибути якого є name, type та value. Атрибут name задає ім'я поля, type визначає тип поля, а атрибут value задає значення поля.

приклад:

<Input name = "email" type = "hidden" value = "spam@nospam.ru">

9) Кнопка відправки форми (submit)

Служить для відправки форми сценарієм.

<Input type = "Тип" name = "Ім'я кнопки" value = "Текст кнопки">
При створенні кнопки для відправки форми необхідно вказати 2 атрибута: type = "submit" і value = "Текст кнопки". Атрибут name необхідний якщо кнопка не одна, а декілька і всі вони створені для різних операцій, наприклад кнопки "Зберегти", "Видалити", "Редагувати" і т.д. Після натискання на кнопку сценарієм передається рядок ім'я = текст кнопки.

10) Кнопка для завантаження файлів (browse)

Служить для реалізації завантаження файлів на сервер. Об'єкт browse начитається з парних тегів <form> </ form>. Початківець тег <form> містить необхідний атрибут encrypt. Атрибут encrypt приймає значення multipart / form-data, який сповіщає сервер про те, що разом зі звичайною інформацією надсилається і файл. При створенні текстового поля також необхідно вказати тип файлу - "file".

<Form enctype = "multipart / form-data" action = "upload.php" method = "post">
Завантажити файл: <input name = "my_file" type = "file">
  <Input type = "submit" value = "Відправити">
</ Form>

11) Рамка (fieldset)

Об'єкт fieldset дозволяє вам намалювати рамку навколо об'єктів. Має закриває тег </ fieldset>. Тема вказується в тегах <legend> </ legend>. Основне призначення об'єкта - задавання різних стилів оформлення.

приклад:

<Fieldset>
<Legend> Програмне забезпечення (заголовок рамки) </ legend>
Текст, який буде поміщений всередині рамки. </ Fieldset>

Всі дані, які ви хочете отримати з HTML-форми в PHP сценарій обробляються за допомогою суперглобальних масивів **$_POST** або **$_GET**, в залежності від зазначеного в атрибуті **method** методу передачі даних.

***Приклади*:**

Завдання: Вам необхідно отримати дані з текстового поля і багаторядкового поля введення і передати їх сценарієм.
Рішення: Необхідно створити HTML форму і PHP - сценарій для обробки форми.

Обговорення:

Створимо два файли: *form.html* і *action.php*. У файлі *form.html* міститиметься html-форма з текстовим полем *mytext* і текстовою областю *msg*:

```
<Form action = "action.php" name = "myform" method = "post">
   <Input type = "text" name = "mytext" size = "50">
  <Textarea name = "msg" cols = "20" rows = "10"> </ textarea>
    <Input name = "Submit" type = "submit value" = "Відправити дані">
</ Form>
```

У цій html-формі нас цікавить 3 атрибута: *action* який вказує шлях до обробника форми, ім'я текстового поля (*mytext*) і ім'я багаторядкового поля вода (*msg*). Також в формі присутня кнопка, при натисканні на яку відбувається передача даних.

Після того як html-форма готова, нам необхідно створити обробник форми *action.php*:

```
<? Php
  $ text = $ _POST [ 'mytext'];
  $ msg = $ _POST [ 'msg'];
  echo $ text; // Виводимо вміст текстового поля
?>
```

Після того як ми введемо будь-які значення в текстові поля і натиснемо на кнопку "Відправити дані", html-форма відправить значення сценарієм *action.php*.
Після цього в змінних *$text* і *$msg* будуть міститися значення текстового поля і багаторядкового поля введення відповідно, значення яких взяті з суперглобальних змінних $_POST.

Якщо ви хочете, щоб в багаторядковому текстовому полі дотримувалося html-форматування, то потрібно скористатися функцією *nl2br ():*

```
<? Php
  $ Text = nl2br ($ _ POST [ 'mytext']);
?>
```

Завдання: Нехай необхідно створити список, що випадає з роками з 2000 по 2050.
Рішення: Необхідно створити HTML форму с елементом SELECT і PHP - сценарій для обробки форми.

Обговорення:

Для початку створимо два файли: *form.html* і *action.php*. У файлі *form.html* міститиметься html-форма з списком, що випадає. Причому значення в списку можна вказати двома способами:

I. Введення даних вручну:

```
<Select class = "input" name = "years">
<Option value = '2000'> 2000 </ option>
```

```
<Option value = '2001'> 2001 </ option>
<Option value = '2002'> 2002 </ option>
.....
<Option value = '2050'> 2050 </ option>
</ Select>
```

II. Введення даних через цикл:

```
<Select class = "input" name = "years">
<? Php
$year = 2000;
for ($i = 0; $i <= 50; $i ++) // Цикл від 0 до 50
{
  $ new_years = $year + $i; // Формуємо нове значення
  echo '<option >'. $new_years. '</ option>'; // Формуємо нову сходинку
}
?>
</ Select>
```

Як видно, другий приклад з циклом, більш компактний. Скрипт обробника даної форми обробляється точно так само як текстове поле, тобто значення списку можна витягти з суперглобального масиву $ _POST.

Завдання: Завантаження файлу на сервер
Рішення: Необхідно створити HTML форму і PHP - сценарій для обробки файлу.

Опис:

Створимо HTML-форму для відправки файлу на сервер.

```
<FORM  ENCTYPE = "multipart / form-data" ACTION = "action.php" METHOD = POST>
<INPUT NAME = "myfile" TYPE = "file">
<INPUT TYPE = "submit" value = "Передати файл">
</ FORM>
```

У даній html-формі присутній елемент *browse*, який відкриває діалогове вікно для вибору файлу для завантаження на сервер. При натисканні на кнопку "Передати файл", файл передається сценарієм-обробника.

Потім необхідно написати сценарій обробник *action.php*. Перед написанням обробника необхідно визначитися в якій каталог ми будемо копіювати файл:

```
<? Php
if (isset ($ _ FILES [ "myfile"])) // Якщо файл існує
{
  $catalog = "../image/"; // Наш каталог
  if (is_dir ($ catalog)) // Якщо такий каталог є
  {
    $myfile = $_FILES [ "myfile"] [ "tmp_name"]; // Тимчасовий файл
    $myfile_name = $_FILES [ "myfile"] [ "name"]; // Ім'я файлу
```

```
   if (!copy ($myfile, $catalog)) echo 'Помилка при копіюванні файлу'. $ myfile_name //
Якщо не вдалось скопіювати файл
  }
  else mkdir ( '../ image /'); // Якщо такого каталогу немає, то ми його створимо
}
?>
```

Зауваження

Якщо ви довіряєте користувачам закачувати на ваш сервер будь-які файли, потрібно бути гранично обережним. Зловмисники можуть впровадити «нехороший» код в картинку або файл і відправити на сервер. У таких випадках потрібно жорстоко контролювати завантаження файлів.

Даний приклад демонструє створення каталогу та копіювання файлу в цей каталог на сервер.

Продемонструємо приклад з елементом *checkbox*. Цей елемент трохи відрізняється від інших елементів тим, що якщо жоден з елементів checkbox не обраний, суперглобальна змінна $ _POST поверне порожнє значення:

```
<FORM  ACTION = "file.php" METHOD = POST>
<Input name = "mycolor" type = "checkbox" value = "blue"> Синій
<Input name = "mycolor" type = "checkbox" value = "black"> Чорний
<Input name = "mycolor" type = "checkbox" value = "white"> Білий
<Input name = "Submit" type = "submit" value = "Вибрати">
</ FORM>
<? Php
  if (!empty ($_POST [ 'mycolor'])) echo $_POST [ 'mycolor']; // Якщо обраний хоч 1
//елемент
  else echo "Виберіть значення";
?>
```

2. Створення функцій в PHP

Оголошення функції починається службовим словом **function**, потім слідує ім'я функції, після імені функції - список аргументів в дужках. Тіло функції полягає в фігурні дужки і може містити будь-яку кількість операторів.

Вимоги, що пред'являються до імен функцій:

- Імена функцій можуть містити російські букви, але давати функцій імена, що складаються з російських букв не рекомендує;
- Імена функцій не повинні містити пробілів;
- Ім'я кожної користувальницької функції повинно бути унікальним. При цьому, необхідно пам'ятати, що регістр при оголошенні функцій і зверненні до них не враховується. Тобто, наприклад, функції funct () і FUNCT () мають однакові імена;

- Функцій можна давати такі ж імена, як і змінним, тільки без знака $ на початку імен.

Типи значень, що повертаються для користувача функціями, можуть бути будь-якими. Для передачі результату роботи для користувача функцій в основну програму (скрипт) використовується конструкція **return**. Якщо функція нічого не повертає, конструкцію **return** не вказують. Конструкція **return** може повертати все, що завгодно, в тому числі і масиви.

Приклад:

```
<? Php

function funct () {
$ Number = 777;
 return $ number;
}
$ A = funct ();
echo $ a;

?>
```

У розглянутому прикладі функція funct повертає за допомогою конструкції return число 777. Повернене функцією значення присвоюється глобальної змінної $a, а потім оператор echo виводить значення змінної $a в браузер. В результаті ми побачимо в браузері число 777.

Приклад елементарної функції, яка просто друкує "Hello World":

```
<? Php
  function printHello () {
    echo "Hello World";
  }
  printHello ();
?>
```

В даному прикладі ми створили функцію з ім'ям "printHello" без параметрів (всередині круглих дужок нічого немає), потім ми в фігурних дужок написали код функції. Тобто в даному випадку, ми виводимо рядок. За межами функції ми її викликаємо.

Однак, частіше все-таки функція вимагає параметра, і давайте з Вами створимо функцію, якої будемо передавати рядок, а потім цей рядок виводити:

```
<? Php
  function printHello ($ str) {
    echo $ str;
  }
  printHello ( "Hello World !!!");
?>
```

Тут ми створили функцію, але вже з параметром. І далі значення цього параметра ми і виводимо через echo. За межами функції printHello () ми її викликаємо і передаємо рядок "Hello World !!!". Як бачите, і тут все просто.

Деякі стандартні функції при роботі зі змінними:

empty - визначає, порожня чи змінна
floatval - Отримує значення з плаваючою точкою змінної
get_defined_vars - Повертає масив всіх визначених змінних
gettype - Повертає тип змінної
intval - Отримує ціле значення змінної
is_array - Визначає, чи є змінна масивом
is_bool - Визначає, чи є змінна двійковим числом
is_double – Псевдонім функції is_float ()
is_float - Визначає, чи є змінна числом з плаваючою точкою
is_int - Визначає, чи є змінна цілим числом
is_integer - Псевдонім функції is_int ()
is_long - Псевдонім функції is_int ()
is_null - Визначає, чи є змінна NULL
is_numeric - Визначає, чи є змінна числом або числовий рядком
is_object - Визначає, чи є змінна об'єктом
is_real - Псевдонім функції is_float ()
is_resource - Визначає, чи є змінна ресурсом
is_scalar - Визначає, чи є змінна скалярною
is_string - Визначає, чи є змінна рядком
isset - Визначає, чи встановлена змінна
print_r - Друкує інформацію про змінну в зрозумілому людині вигляді
serialize - Генерує упаковане представлення значення
settype - Встановлює тип змінної
strval - Отримує строкове значення змінної
unserialize - Створює значення PHP з упакованого уявлення
unset - Знищує створену змінну