МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

СТВОРЕННЯ ПАКЕТУ ПРОГРАМ НА PHP І HTML ДЛЯ УПРАВЛІННЯ ФУНКЦІЯМИ БАЗИ ДАНИХ

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторної роботи № 4
з курсу “Бази даних та знань”
для студентів спеціальності 125 Кіберебзпека та захист інформації

Затверджено
на засіданні кафедри
“Захист інформації”
протокол № від . .2023 р.

Львів – 2015

Створення пакету програм на php і html для управління функціями бази даних: Методичні вказівки до лабораторної роботи №4 з курсу “Бази даних та знань” для студентів спеціальності 125 Кібербезпека та захист інформації // Укл. А.І. Партика, О.І. Гарасимчук. - Львів: НУЛП, 2015. – 23 с.

Укладачі: Партика А.І, канд. техн. наук, асистент
Гарасимчук О.І., канд. техн. наук, доц.

Відповідальний за випуск:
Гарасимчук О.І., к.т.н, доц. каф. ЗІ

Рецензент:
Микитин Г.В., проф., д.т.н, проф. каф. ЗІ

Метою роботи є вивчення механізмів і принципів проектування баз даних за допомогою мов програмування HTML і php, здобуття практичних навиків їх проектування та створення пакету прикладних програм для простої бази даних із забезпеченням необхідного фукціоналу.

1. Теоретична частина

1.1. Базові елементи HTML

Елемент HTML

Для позначення меж HTML-документу використовується подвійний тег <html>. Початковий тег <**html**>, у якого відсутні атрибути, розміщується на початку html-файлу, а кінцевий тег <**/html** > є останнім тегом коду і позначає закінчення всього документу. Елемент html не входить до складу інших елементів, тобто з елементом html не пов'язаний жоден з об'єктів, які відображаються у вікні браузера.

До складу контейнеру html входять два структурні елементи: head (елемент заголовку) та body (основна частина або тіло документу). Таким чином, документ html має вигляд:

```
<html>
...
<! -- Тут розміщуються елементи заголовку і основної частини документу -- >
...
</html>
```

Заголовок документу(елемент HEAD)

Заголовок HTML-документу визначається елементом head. Тег <**head**> не має атрибутів. Елемент head розміщується одразу після тегу <html>, перед основною частиною Web-сторінки:

```
<html>
<head>
<!-- Тут розміщується заголовок документу -->
</head>
...
</html>
```

Елемент head (як і елемент html) не відображається при перегляді Web-сторінки, він надає браузеру загальну інформацію про HTML-файл.

Назва документу(елемент TITLE)

До заголовку документу входить обов'язковий елемент, який представлений контейнером <title>. Все, що знаходиться між парою тегів <**title**> і <**/title**>, інтерпретується браузером як назва Web-сторінки.

Якщо в документі є гіперпосилання, то назва документу, на який вказано посилання, буде з'являтися у вигляді плаваючої підказки при наведенні на посилання вказівника миші. Елемент title по відношенню до елемента head є дочірнім,тобто вкладеним у контейнер <head>:

```
<html>
<head>
```

```
<title> Назва Web-сторінки </title>
  ...
</head>
...
</html>
```

Тіло документу(елемент BODY)

Елемент body є наступним компонентом Web- сторінки. Парні теги <**body**> і <**/body**> вказують на початок і кінець тіла документа. Весь зміст документа міститься в елементі body.

Початкові і кінцеві теги елементу body є необов'язковими в структурі html-документа. Проте контейнер body необхідний для того, щоб задати властивості всієї сторінки. Наявність в html-документі елементу body є формальною ознакою того, що даний документ має звичайну структуру.

Тег <body> розміщується безпосередньо після елементу head, а кінцевий тег </body> є передостаннім тегом документу:

```
<html>
<head>
<title> Назва Web-сторінки </title>
</head>
<body>
<!-- Зміст документу -->
...
</body>
</html>
```

Початковий тег <body> може доповнюватися декількома атрибутами, які визначають зовнішній вигляд документу в цілому.

1.2.Форматування тексту

Вирівнювання тексту та горизонтальна лінія

Для того, щоб вставити розрив рядка, необхідно скористатися тегом <**br**> . Якщо потрібно зробити так, щоб в певному місці текст НЕ МІГ бути розірваним, необхідно написати тег <**nobr**>. Для розбиття тексту на параграфи використовується тег <**p**>текст<**/p**>

Тег <p> має один параметр **align**, який вказує на спосіб вирівнювання тексту в параграфі: текст може бути вирівнений по правій (**right**), лівій (**left**) межах або по центру (**center**). Якщо параметр пропущено, то текст вирівнюється по лівому краю. Щоб розмістити параграф по центру можна також використовувати тег …

Ще одним способом розбиття тексту на частини можна назвати горизонтальну лінію. Вона є елементом мови HTML і вставляється в текст за допомогою тегу <**hr**>, де параметри можуть бути наступними:

width – задає ширину лінії

size – задає товщину лінії

noshade – лінія не матиме тіні.

Коментар.

Коментар міститься між послідовностями < !-- текст коментаря -- >

Заголовок.

Для їх створення використовуються теги <**н1**>...</**h1**>,..., <**h6**>...</**h6**>. Розмір 1 відповідає найбільшому шрифту, а розмір 6 — найменшому.

Шрифти

Основні шрифти, які використовуються в HTML можна подати за допомогою наступної таблиці:

Тег	**Відповідний тегу шрифт**
<**TT**>	Телеграфний курсив
<**I**>	Курсив
<**b**>	Жирний шрифт
<**big**>	Великий шрифт
<**small**>	Маленький шрифт
<**STRIKE**> або <**S**>	Перекреслений шрифт
<**u**>	Підкреслений шрифт

Іншим засобом логічного виділення можна вважати тег <**strong**> ...</**strong**>. Зміст тегу звичайно відображається жирним шрифтом. Змінити розмір і колір шрифтів можна за допомогою тегу <**font** >...</**font**>. Параметри тегу можуть бути наступними:

size =``value або value" – задає абсолютний або відносний розмір шрифту. Відносний розмір задається по відношенню до базового розміру. Діапазон можливих значень від 1 до 7.

size – задає размір шрифта.

color – вказує колір для тексту.

Наведемо тепер таблицю з деякими назвами і кодами кольорів.

Українська назва	**Англійська назва**	**Код**
Аквамарин	Aqua	#00FFFF
Білий	White	#FFFFFF
Жовтий	Yellow	#FFFF00
Зелений	Green	#008000
Золотий	Gold	#FFD700
Індіго	Indigo	#4B0080
Каштановий	Maroon	#800000
Червоний	Red	#FF0000
Оливковий	Olive	#808000
Помаранчевий	Orange	#FFA500
Пурпуровий	Purple	#800080
Світло-зелений	Lime	#00FF00
Сріблястий	Silver	#C0C0C0
Сірий	Gray	#808080
Сизий	Teal	#008080
Синій	Blue	#0000FF
Ультрамарин	Navy	#000080
Фіолетовий	Violet	#EE80EE
Фуксиновий	Fuchsia	#FF00FF
Чорний	Black	#000000

1.3. Таблиці та списки

Таблиці

Таблиці є важливим засобом для створення сторінок. До появи в мові html засобів для створення таблиць, не можна було навіть розмістити будь-який текст у колонках, не кажучи вже про створення якої-небудь таблиці. Але після появи таких можливостей Web- дизайнери почали використовувати її в повній мірі й досягають досить складних ефектів.

Створюється таблиця за допомогою такої конструкції:

```
<table>
<caption> Заголовок </caption>
<tr>
<td>
...
</td>
....
</tr>
</table>
```

Властивості тегу <TABLE>

Тег **table** починає опис таблиці і може мати наступні параметри:

border – визначає товщину рамки таблиці. Якщо вказано нульове значення, то рамка не відображається. Якщо цей параметр не вказаний, його значення вважається нульовим (рамка не відображається).

width, **height** – вказують розміри таблиці, якщо вони повинні бути чітко заданими.

align – визначає, як повинна бути вирівняна таблиця: справа (right), зліва (left) або по центру (center) сторінки.

cellspacing – число точок між окремими комірками в таблиці.

cellpadding – число точок між рамкою і змістом комірки.

За допомогою тегу <**tr**>...</**tr**> оформлюються рядки таблиці. Він може мати наступні параметри:

align – використовується для задання способу горизонтального форматування даних всередині комірок: вони можуть вирівнюватися по правій (right), по лівій (left) межах або по центру (center).

valign – використовується для задання вертикального форматування даних всередині комірок: вони можуть вирівнюватися по верхній (top), нижній (bottom) межам, по центру (middle) або мати загальну базову лінію (baseline).

Конкретні комірки задаються за допомогою тегу <**td**>...</**td**>, де параметри можуть бути наступними:

width – задає ширину комірки;

colspan – визначає, скільки колонок таблиці комірка буде перекривати;

rowspan – визначає, скільки рядків таблиці комірка буде перекривати;

nowrap – якщо цей параметр вказаний, то вміст комірки буде заповнюватись без переносу;

bgcolor – параметр, який підтримується сучасними браузерами: вказує колір фону комірки у вигляді RGB-триплету або символьного імені.

Ще один тег для для оформлення комірок таблиць – тег <**th**>...</**th**> – потрібен для задання комірок-заголовків. Він в усьому співпадає з тегом <td>, але на відміну від нього, зміст виділяється жирним шрифтом і розміщується по центру.

Якщо потрібно задати заголовок УСІЄЇ таблиці, то використовується тег <**caption**>...</**caption**>. Він має бути в середині тегу <table>, але поза описом комірок. У нього є лише один параметр: **align** – вказує положення заголовку: він може бути в верхній (top) або нижній (bottom) частині таблиці.

Списки

Списки в HTML бувають двох видів: впорядковані (пронумеровані) і невпорядковані. Відрізняються вони лише способом оформлення. Перед пунктами невпорядкованих списків звичайно ставлять символи-булети (bullets), наприклад, точки, ромбики і т.п., в той час, як пунктам впорядкованих списків передують їх номера.

Впорядкований (пронумерований) список. В HTML список складається з тегу-контейнеру списку, який визначає його тип і стандартних тегів <**li**>, що передують кожному пункту списку. Коли браузер зустрічає тег впорядкованого списку, він послідовно нумерує пункти списку: 1, 2, 3 і т. д. Впорядкований список відкривається тегом <**ol**>, а кожен його пункт починається стандартним тегом <**li**>. Для створення заголовку списку використовується спеціальний тег <**lh**>. Список закривається тегом </**ol**>. Відкриваючий і закриваючий теги забезпечують переведення рядка до і після списку, відділяючи таким чином список від решти тексту, тому тут немає необхідності використовувати теги абзацу <p>. Впорядковані списки дозволяють вкладення одне в одного.

Атрибути тегу <OL>

Вони дозволяють встановлювати вид маркерів елементів списку, а також задавати початковий маркер списку. Наприклад:

– type=A – встановлює маркер у вигляді прописних літер;

– type =a – встановлює маркер у вигляді малих прописних літер;

– type =I – маркер у вигляді великих римських цифр;

– type =i – маркер у вигляді маленьких римських цифр;

– type =1 – маркер у вигляді арабських цифр;

Невпорядкований (маркований) список. Це список, у якому відношення між пунктами невизначені. Маркований список замість буквеної або числової нумерації передбачає використання різних символів, які називаються маркерами. Список розміщується всередині контейнера <**ul**>. Браузер створює автоматичний доступ для вкладених списків.

Можна встановити вид маркерів в невпорядкованих списках за допомогою атрибута type, який може набувати трьох значень: **disc**, **squre**, **circle**. Тег <ul> має атрибут **compact**, який дозволяє виводити список в більш компактному вигляді.

Меню

Список, який визначається <**menu**> виводиться більшістю браузерів тими ж шрифтами і в тому ж стилі, що і невпорядкований список. Як і попередні типи списків, список-меню автоматично відокремлюється від решти тексту кодами переведення рядка.

Список типу <DIR>

Елемент типу dir схожий на елемент menu і використовується для ідентифікації певної частини документу. Список контейнера, що починається тегом <dir>, виводиться браузером аналогічно невпорядкованим спискам. Довжина кожного пункту цього типу списків обмежена 24 символами. Список потребує зачиняючого тегу </dir>.

Список визначень. Це особливий вид списків HTML. Вони подають текст у вигляді словникової статті, що складається з певного терміну та абзацу, який розкриває його значення. Елемент списку визначень **dl** є контейнером і забезпечує відокремлення списку від іншого тексту порожніми рядками. Всередині контейнеру тегом <dt> помічається термін, що визначається, а тегом <**dd**> – абзац з його визначенням. Теги <dt> и <dd> не є контейнерами. Текст після тегу <dt> повинен міститися в одному рядку. Якщо ця вимога не виконана або якщо рядок виходить за межі вікна браузера, то відбувається переведення рядку, але без відступу. Текст, що стоїть за тегом <dd> виводиться окремим абзацом з відступом вниз на один або два рядки відносно цього терміну.

1.4.Посилання та робота з ними

Посилання складається з двох частин. Перша з них – це те, що ми бачимо WEB-сторінці; вона називається вказівник (**anchor**). Друга частина, яка дає інструкцію браузеру – адресна частина посилання (URL–адрес).

Вказівники бувають 2 типів – *текстові* та *графічні*. Текстовий вказівник – це слово або слова, що підкреслені прямою лінією. Колір вказівника може регулюватися автором або настройками програми перегляду. Текстові вказівники можуть бути безпосередньо частинами тексту, наприклад: `<a href=“vero.html”> Привіт </a>` або бути списком посилань, з яких потрібно зробити вибір. Графічні не підкреслюються і не виділяються кольором, проте навколо них можна зробити рамку. Багато авторів HTML-документів використовують графічні вказівники у вигляді кнопок, характерних для Windows. Такі кнопки можна досить часто побачити на початкових сторінках Web-сайтів. Вони слугують свого роду змістом сайту і ведуть до різних сторінок, наприклад: `<a href=“whatsnew.htm”> <img src=”whatsnew.gif” border=0> </a>`.

Використання зображення у якості посилання. Для використання зображення в якості посилання на інший документ, достатньо включити ім'я файлу зображення в тег <a>:

```
<a href=“whatsnew.htm”> <img src=”image.gif”> </a>
```

Тег BASE. Вказує базову адресу даного документа (URL), який стане відправною точкою для розрахунку відносних адрес всередині документу. Елемент не має кінцевого тегу. Обов'язкова наявність хоча б одного з параметрів. Параметри: **href** – визначає базову адресу (URL) даного документу; **target** – визначає ім'я фрейму, яке буде використовуватися в гіперпосиланнях по замовчуванню. Приклад:

```
<head>
 <!-- Нехай браузер думає, що знаходиться за адресою : -->
 <base href="http:// igf-ua.org/other/index.html">
 <title> Інструкція з експлуатації </title>
</head>
...
<!-- А тепер створимо відносне посилання на документ -->
<!-- http:// igf-ua.org/list.html -->
<a href="../list.html"> Список </A>
...
```

Створення посилань на документи і файли

Потрібно повідомити браузеру, який елемент сторінки є вказівником, і вказати адресу документа, на який робите посилання. Обидві дії виконуються за допомогою тегу <a>. Тег <a> має єдиний атрибут href, в якому розміщується URL-адрес. Тег є контейнером, тобто потребує . Атрибут href вказує на абсолютну адресу сторінки. Вказівником можуть бути слова, які браузер виділяє підкресленням та кольором. Текст поза контейнером не є вказівником і не буде підкресленим. Коли читач клікає мишкою на цих словах, браузер загружає його сторінку.

Тег LINK. Елемент link описує взаємозв'язок документу з іншими документами на сайті, вказуючи його місце в ієрархічній структурі сайту. Елемент не має кінцевого тегу. В заголовку може міститися декілька елементів link. Параметри: **href** – визначає URL об'єкту; **rel** – визначає тип взаємозв'язку даного документу з об'єктом, який визначений параметром href. Можливі значення:

Stylesheet – вказує на файл, який містить таблицю стилів (CSS) для даного документу. Браузер загрузить css-файл з вказаної в параметрі HREF адреси та застосує його до даного документу;

home – вказує на головну сторінку вашого сайту;

toc, contents – вказують на файл, який містить зміст даного документу;

index – вказує на файл, який містить інформацію для індексного пошуку в даному документі;

glossary – вказує на файл, який містить перелік термінів,що відносяться до даного документу;

copyright – вказує на сторінку сайту, де говориться про його авторів, авторських правах і т.д.;

up, parent – вказує на "батьківську" сторінку (документ, що стоїть на сходинку вище у ієрархічній структурі вашого сайту);

child – вказує на "дочірню" сторінку (документ, що стоїть на сходинку нижче у ієрархічній структурі вашого сайту);

next – вказує на наступну сторінку у послідовності документів (напр. наступну сторінку електронного каталогу, документації або словника);
previous – вказує на попередню сторінку в послідовності документів;
last, end – вказує на останню сторінку в послідовності документів;
first – вказує на першу сторінку в послідовності документів;
help – вказує на сторінку з підказкою;
type – визначає MIME-тип для об'єкту, вказаного в параметрі href.

Приклад:

```
<head>
<title> Елемент div </title>
<link rel="home" title="html-довідник" href="index.html">
<link rel="up" title="текстові блоки" href="textblocks.html">
<link rel="previous" title="елемент p" href="p.html">
<link rel="next" title="елемент address" href="address.html">
</head>
```

Додаткові відомості про тег A.

1)Елемент A не може бути вкладеним сам у себе, тобто неможливі конструкції:

```
<a href="link1.html">
Перший лінк
<a href="link2.html">Другий лінк</a>
Продовжуємо перший лінк
</a>
```

2) За допомогою елементу base можна вказати значення target для всіх гіперпосилань в даному документі.

2.1. Основи PHP

На даний момент PHP являється основною мовою програмування, що застосовується для веб розробки. Але спершу розглянемо яка саме різниця між статичними і динамічними веб сторінками і як працюють програмні скрипти. Статичні сайти створюються за допомогою html верстки. Веб сторінки зберігаються в форматі html й їх можна переглянути за допомогою будь-якого браузера на локальному ПК.

Програмні скрипти навідміну від html коду потребують обробки на сервері для того, щоб браузер інтернет користувача зміг коректно відобразити контент сторінки динамічного сайту.

Будь-який скрипт написаний на php береться в скоби:

```
<?php
……
?>
```

По завершенні кожного нового рядка програмного коду завжди ставимо крапку з комою:

```
<?php
echo"Hello!";
?>
```

Кожна нова команда починається завжди з нового рядка.

Для пояснення тої чи іншої дії в php використовують коментарі. Інформація, що знаходиться в них не виконується програмою:

```
<?php
echo"Hello!";//це однорядковий коментар
echo"Good!"; /* так виглядає
багаторядковий коментар*/
?>
```

Оператор виводу echo в PHP.

Для того, щоб вивести результат виконання роботи скрипта на екран браузера в php програмуванні існують спеціальні оператори, а саме **print** і **echo**. В програмному коді програмісти частіше користуються останнім.

Отже оператор **echo** – це оператор, який виводить присвоєні йому данні на екран. Данні завжди беруться в лапки, а в кінці стоїть крапка з комою. В даній команді можна заключати також теги форматування тексту:

```
<?php echo"<p align='center'> Hello people! </p>"; ?>
```

Як бачимо атрибути тегів форматування тексту взяті не в звичні лапки, як в html коді, а в одинарні. В echo можна помістити текст із змінною:

```
<?php echo "перший текст" .$text. "другий текст"; ?>
```

В такому випадку в лапки береться лише сам текст, а перед змінною і після неї, тобто перед другим текстом ставимо крапку.

В PHP є можливість «склеювати» змінні між собою в операторі виводу:

```
<?php echo $a . $text; ?>
```

2.2. Змінні в PHP та їх функції

Уявіть собі, що в нас є декілька типів даних, це можуть бути числові, чи текстові значення й нам необхідно застосувати до них якусь конкретну дію, операцію. Простіше буде присвоїти ці значення визначеним символам й прописувати в коді скрипта не самі значення, а оперувати з символами, яким ці значення належать. Отже вище описані символи являються своєрідними контейнерами, що містять присвоєний їм тип данних. І такі "контейнери" в PHP прийнято називати - **змінними**.

Будь-яка змінна завжди починається знаком долара, тобто "$". Окрім цього всі змінні наділені іменами. Імена можуть бути довільними, наприклад "$name", або просто "$a". Але завжди слід пам'ятати, що вони є чутливими до регістру. Тобто $var і $VAR насправді зовсім різні змінні.

Змінним завжди присвоюється якесь значення, іншими словами певний тип даних. Оператором присвоєння служить знак рівності "=" а в кінці, після присвоєного значення, яким може бути ціле, або дробове число, чи текстовий рядок, обов'язково ставиться крапка з комою.

Приклад:

```
<?php
$var=2;
?>
```

або

```
<?php
$var="Bob";
?>
```

В першому випадку значення змінної $var, тобто число "2" в лапки не береться, а ось в другому – текстове значення присвоєне змінній береться вже в лапки.

Нижче написано невеликий скрипт для того наочної демонстрації застосовування **змінних в php.**

```
<?php
$a=34;
$c=5;
$ress=$a+$c;
echo $ress;
?>
```

Присвоюємо змінним $a і $c певні значення. В змінній $ress виконується арифметична дія, іншими словами змінній $ress присвоюється сума значень змінних $a і $c. А оператор echo виводить результат в вигляді числа "39".

Окрім цього існує можливість перетворення змінних з одного типу в інший. Ця дія реалізується посередністю спеціальних операторів перетворення, яких є п'ять:

(string) – рядок;

(integer) – ціле число;

(double) – число з плаваючою комою;

(boolean) – булевська змінна; (true/false).

Наприклад нам потрібно вивести результат на екран в вигляді не дробового, а цілого числа, тобто не 46.77, а лише 46. Значить для цього необхідно перетворити змінну з плаваючою комою в цілочислову. Ось як це виглядає:

```
<?php
$a = 46.77;
echo ((integer)$a);
?>
```

2.3. Константи в php програмуванні

Константи в php це ті ж самі змінні лише з цією різницею, що присвоєні їм дані неможливо змінити жодним способом.

Константи також наділені іменем, як і змінні, але їхні імена зазвичай прописують буквами верхнього регістру.

В php програмуванні константи створюються за допомогою функції **define**. Ось приклад застосування констант:

```
define("DATA", "20th July");
```

Тобто синтаксис написання наступний:

```
define("ім'я константи", "присвоєне значення");
```

Якщо ви хочете вивести на екран браузера значення константи, то в операторі виводу слід просто вказати її ім'я. Наприклад:

```
<?php
define("DATA", "20th July");
echo DATA;
?>
```

2.4. Арифметичні операції в PHP.Логічні оператори в PHP й оператори порівняння.

З змінними можна проводити такі операції, як: додавання, віднімання, ділення, множення, порівняння, збільшення. Нижче приведений приклад виконання арифметичних операцій в php коді:

```
<?php
$a=5 + 11;
$b=$a - 2;
$c=$a * 7;
$d=$a / $c;
? >
```

Нижче наведений приклад показує, як можна збільшити змінну на одиницю й вивести результат з допомогою оператору виводу:

```
<?php
$a=5;
$a++;
echo$a;
? >
```

Для порівняння між собою двох змінних, вірніше присвоєних їм даних використовують наступні **оператори порівняння в php**:

рівно (= =);
не рівно (!=);
більше (>);
менше (<);
менше або рівно (<=);
більше або рівно (>=).

Приклад:

```
<?php
if($name= =" ")
{
echo"Введіть своє ім'я!";
}
?>
```

Наведений фрагмент php коду застосовується в скриптах, що призначені для обробки html форм. Тобто якщо поле форми не заповнене, іншими словами рівне пустоті, то оператор виводу виводить на екран повідомлення про помилку.

А **логічні оператори в php** застосовуються в скриптах для перевірки конкретної умови на предмет істини, або невірного значення:

$a and $b; - істина є тоді, якщо одна і друга змінна є істинними;
$a or $b; - істина є тоді, якщо або одна, або друга змінна є істинною;
!$a; - істина є тоді, якщо змінна $a не є істинною;

$a && $b; - істина є тоді, якщо одна і друга змінна є істинними; $a || $b; - істина є тоді, якщо або одна, або друга змінна є істинною.

2.5. Оператор else і оператор if в PHP

Умовний **оператор else і оператор if** використовують в скриптах для перевірки певних умов. Отже маємо певну умову, якщо результат перевірки даної умови є істинна, то виконується одна дія, в противному випадку – інша.

Ось синтаксис:

```
if (умова) {дія, що виконується при даній умові}
else
{
виконується інша дія при неможливості виконання умови
}
```

Часто буває, що умов є декілька і їх всіх потрібно перевірити. Для подібної цілі використовують оператор **elseif**:

```
if (умова1)
{
дія 1
}
elseif (умова2) //інакше при виконанні другої умови
{
виконується дія 2
}
elseif (умова3)
{
дія 3
}
else //якщо жодна з умов не виконується
{
дія 4
}
```

2.6. Цикли в PHP

Цикли дозволяють виконувати конкретну дію декілька разів підряд. Період протягом якого відбувається проходження циклу називається - **ітерацією**.

В PHP існує чотири типи циклів: **while**; **do-while**; **for**; **foreach**. Останній призначений в основному для роботи з масивами.

Цикл while використовується коли є певна умова, а також дія, що буде відбуватись доти, доки дана умова вірна. Якщо умова перестає бути вірною, виконання циклу припиняється:

```
while (умова) {дія, що буде виконуватись доти, поки дана умова вірна}
```

В операторі **циклу do-while** спочатку відбувається певна дія, а вже потім здійснюється перевірка чи дана умова все ще вірна:

```
do
{
```

```
дія, що буде виконуватись доти, поки умова вірна
}
while (умова);
```

Синтаксис **циклу for** наступний:

```
for (вираз1 - керуюча змінна; вираз2 - умова при виконанні якої виконується дія; вираз3 - розраховується в кінці кожного циклу)
{
дія
}
```

Приклад для даного оператора циклу:

```
<?php
$c = 5;
for ($i = 1; $i <= $c; $i++) /*визначаємо керуючу змінну, далі перевіряємо умову, збільшуємо керуючу змінну на одиницю й знову перевіряється умова*/
{
echo "$i"; //виводимо змінну $i, якщо умова вірна
}
?>
```

2.7. Перевірка полів форми

Перевірка полів форми в php здійснюється за допомогою умовних операторів. Для того, щоб перевірити поля форми, чи заповнені вони користувачем при передачі даних з html форми, на сайті нам буде необхідно використати такі умовні оператори, як **if** й **else**, та оператор порівняння. Припустимо в нас є три поля форми, яким відповідають змінні $name, $email і $mess.

Ось яким буде приклад php коду для перевірки полів форми:

```
<?php
if (($name == "") || ($password == "") || ($email == ""))
//якщо поля форми пусті
{
echo "Форма незаповнена повністю! Заповніть всі поля форми!";
//оператор echo виведе на екран вказаний текст
exit (); //зупиняємо виконання скрипта
}
else //інакше
{
echo "Повідомлення відправлено успішно!"; /*якщо поля форми заповнені програма виведе повідомлення такого змісту*/
}
?>
```

2. ПОСЛІДОВНІСТЬ ВИКОНАННЯ РОБОТИ

2.1. Необхідно створити пакет програм по створенню простої бази даних(БД), її доповненню, пошуку запису, видаленню запису і її редагуванню.

2.2. Пакет програм повинен містити загальне меню по виконуваних діях в якому повинен бути встановлений лічильник відвідування цієї сторінки. Пакет програм повинен оперувати з графічними об'єктами (наприклад, фотографії однокурсників), якщо це передбачено в самій БД.

2.3. Текст програм повинен бути написаний на мовах HTML і php. Безпосередня робота з базою даних повинна здійснюватися за допомогою пакету СУБД MySQL. Сервер MySQL повинен знаходитися на машині localhost (там же де і Web-сервер Apache). Програми на php повинні взаємодіяти з сервером MySQL за допомогою включених в php функцій взаємодії з MySQL.

2.4. Після вивчення представлених нижче текстів програм студент повинен розробити подібний пакет програм для БД, заданої у 1 та 2 лабораторній роботі. Кожний студент повинен мати свій username в СУБД MySQL. Нижче розглянутий приклад створення користувача з username sydorak.

```
> mysql -u root -p
Enter password:
Welcome to MySQL monitor.   Commands end with ; or \g.
Your MySQL connection id is 72 to server version: 3.22.32

Type 'help' for help.
```

```
mysql>use mysql;
Database changed
mysql> select * from user;
+-----------+-------+-----------------+-------------+-------------+-----------+
| Host      | User  | Password        | Select_priv |Insert_priv  | Update_pri|
+-----------+-------+-----------------+-------------+-------------+-----------+
| localhost | root  | 3aa3ee1e1d77f653 | У          | У           | У         |
| cel       | root  |                 | У          | У           | У         |
| localhost |       |                 | N          | N           | N         |
| cel       |       |                 | N          | N           | N         |
| localhost |sydorak| 75cbfbdb376c43a9 | N          | N           | N         |
| localhost | admin | 12ad110a21defd86 | У          | У           | У         |
+-----------+-------+-----------------+-------------+-------------+-----------+
6 rows in set (0.00 sec)
```

```
mysql>insert into user values
('localhost','sydorak',password('sydorak'),
       -> 'y','y','y','y','y','n','y','n','n','n','n','n','n');
Query OK, 1 row affected (0.00 sec)
```

```
mysql> select * from user;
+-----------+--------+-----------------+-------------+-------------+----------+
| Host      | User   | Password        | Select_priv | Insert_priv | Update_pr|
+-----------+--------+-----------------+-------------+-------------+----------+
| localhost | root   | 3aa3ee1e1d77f653 | У           | У           | У        |
| cel       | root   |                 | У           | У           | У        |
| localhost |        |                 | N           | N           | N        |
| cel       |        |                 | N           | N           | N        |
| localhost | alex   | 75cbfbdb376c43a9 | N           | N           | N        |
| localhost | admin  | 12ad110a21defd86 | У           | У           | У        |
| localhost | sydorak| 7f6731414c9c046a | У           | У           | У        |
+-----------+--------+-----------------+-------------+-------------+----------+
7 rows in set (0.00 sec)

mysql> quit
Bye
>
```

Перезавантажуємо сервер MySQL:

```
> mysqladmin -u root -p reload
Enter password:
>
```

2.5. Далі студент повинен створити свою базу даних і таблицю в ній з якими працюватиме створений пакет програм на php. Для їх створення необхідно підключитися до машини localhost. Пароль входу в базу даних у всіх однаковий і відповідає username. Для його заміни (якщо це потрібно) необхідно виконати команду (ввівши старий пароль):

```
> mysqladmin -u sydorak -h 'localhost' -p password 'sydorak'
Enter password:
>
Потім перенавантажувати MySQL (ввівши новий пароль):
> mysqladmin -u sydorak -h 'localhost' -p reload
Enter password: >
```

Тексти HTML документів для формування форм:

index.html

```
<html>
<head>
<title>Menu</title>
</head>
<body>
<b>
<a href="../php/form1.html">
1.Добавити запис в базу даних </a><br>
<a href="../php/form2.html">
2. Знайти запис по П.І.Б. </a><br>
<a href="../php/form3.html">
3.Видалити запис </a><br>
<a href="../php/form4.html">
4.Відредагувати запис </a>
</b>
<h5><p>Число відвідувань:
<?PHP
```

```
        $filename = "counter.dat";
        $fp = fopen($filename,"r");
        if ($fp) {
                $counter=fgets($fp,10);
              fclose($fp);
        } else {                $counter=0;
        }
        $counter++;
        print $counter;
        $fp = fopen($filename,"w");
        if ($fp) {
                $ii=fputs($fp,$counter);
                fclose($fp);
        }
?>
</h5>
</body>
</html>
```

Файл counter.dat доцільно заздалегідь створити і відкрити доступ до нього для всіх користувачів за допомогою команди:

```
>chmod 0777 counter.dat
```

form1.html

```
<html>
<head>
<title>form1</title>
<body>
<form method=post action="form1.php">
<br><b>Введіть Ваше ім'я:</b>
<br><input name="user_name" value="" size=30>
<br><br><b>Введіть номер Вашого телефону:</b>
<br><input name="phone" value="" size=10>
<br><br><b>Введіть коротку характеристику:
<br><TEXTAREA NAME="job" ROWS=10 COLS=40></TEXTAREA>
<br><br><b>Введіть ім'я фото лат.буквами:</b>
<br><input name="img" value="nofoto.jpg" size=20>
<br><br><input type="submit" value="Зареєструвати">
</form>
</body>
</html>
```

form2.html

```
<html>
<head>
<title>Виборка даних</title>
<body>
<form action="form2.php" method="post">
<b>Введите П.І.Б.<b>
<input type="text" name="user_name" size=30>
<input type="submit" value="Ввести!">
</form>
```

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

form3.html

```
<html>
<head>
<title>Виборка даних</title>
<body>
<form action="form3.php" method="post">
<b>Введите П.І.Б. для видалення:<b>
<input type="text" name="user_name" size=30>
<input type="submit" value="Ввести!">
</form>
</body>
</html>
```

form4.html

```
<html>
<head>
<title>Коректування даних</title>
<body>
<form action="form4.php" method="post">
<b>Введіть П.І.Б. особи, дані для якої необхідно змінити:<b>
<br><input type="text" name="user_name" size=30>
<input type="submit" value="Ввести!">
</form>
</body>
</html>
```

Тексти програм на мові PHP:

form1.php

```
<?
/* Визначаємо значення змінним */
$hostname="localhost";
$username="sydorak";
$password="sydorak";
/* Ім'я бази даних */
$dbName="sydorakdb";
/* Таблиця MySQL */
$usertable="sydoraktab1";
/* Створити з'єднання */
MYSQL_CONNECT($hostname,$username,$password) OR DIE("Не можу
під'єднатися");
/* Вибір БД */
MYSQL_SELECT_DB($dbName) or die("Не можу вибрати БД");
/* Введення інформації в БД */
$query="INSERT INTO $usertable VALUES
('$user_name','$phone','$job','$img')";
$result=MYSQL_QUERY($query);
/* Закрити з'єднання */
MYSQL_CLOSE();
print "Запис введений в БД! <br>";
```

```
print "<a href='../php/'>Назад в меню</a>";
?>
```

form2.php

```
<?php
$hostname="localhost";
$username="sydorak";
$password="sydorak";
$dbName="sydorakdb";
$usertable="sydoraktab1";
mysql_connect($hostname,$username,$password) or die("Не можу
під'єднатися ");
mysql_select_db("$dbName") or die("Не можу вибрати БД");
/* Вибрати співробітників, імена яких починаються на $usr_name */
$query="select * from $usertable where user_name like '$user_name%'";
$result=mysql_query($query);
/* Скільки знайшлося таких співробітників? */
$number=mysql_numrows($result);
/* Роздрук вибраних співробітників */
$i=0;
if ($number == 0)
{
{
print "Немає даних про $user_name в БД <br>";
}
elseif ($number > 0)
{
while ($i < $number)
{
/* Привласнення змінним значень полів user_name, phone, job, img для
i-го рядка */
$user_name=mysql_result($result,$i,"user_name");
$phone=mysql_result#@:;
$job=mysql_result#@;;
$img=mysql_result#@
```

form3.php

```
<?php
$hostname="localhost";
$username="sydorak";
$password="sydorak";
$dbName="sydorakdb";
$usertable="sydoraktab1";
mysql_connect($hostname,$username,$password) or die("Не можу
під'єднатися ");
mysql_select_db("$dbName") or die("Не можу вибрати БД");
/* Видалення запису для поля user_name */
$query="delete from $usertable where user_name='$user_name'";
$result=mysql_query($query);
print "Запис видалений <br>";
print "<a href='../php/'>Назад в меню</a>";
  ?>
```

form4.php

```
<?php
$hostname="localhost";
$username="sydorak";
$password="sydorak";
$dbName="sydorakdb";
$usertable="sydoraktab1";
mysql_connect($hostname,$username,$password) or die("Не можу
під'єднатися ");
mysql_select_db("$dbName") or die("Не можу вибрати БД");
/* Вибір запису по полю user_name */
$query="select * from $usertable where user_name='$user_name'";
$result=mysql_query($query);
/* Визначення існування вибраного запису */
$number=mysql_numrows($result);
if ($number==0)
{
print "Немає такого запису!<br>";
print "<a href='../php/'>Назад в меню</a>";}
else
{
/* Якщо запис вибраний з бази даних, вивести всі її поля у форму для
редагування і видалити її */
$phone=mysql_result($result,0,"phone");
$job=mysql_result($result,0,"job");
$img=mysql_result($result,0,"img");
print "<img src='pics/$img' height=130><br>";
/* Видалення запису */
$query="delete from $usertable where user_name='$user_name'";
$result=mysql_query#@:;
/* Створення форми для редагування */
print "<form method=post action='form5.php'>";
print "<br><b>Введите Ваше ім'я:</b>";
print "<br><input name='user_name' value='$user_name' size=30>";
print "<br><br><b>Введите номер Вашого телефону:</b>";
print "<br><input name='phone' value='$phone' size=10>";
print "<br><br><b>Введіть коротку характеристику:";
print "<br><TEXTAREA NAME='job' ROWS=10 COLS=40>$job</TEXTAREA>";
print "<br><br><b>Введите ім'я фото лат.буквами:</b>";
print "<br><input name='img' value='$img' size=20>";
print "<br><br><input type='submit' value='Зареєструвати'>";
print "</form>";
}
?>
```

form5.php

```
<?
/* Визначаємо значення змінним */
$hostname="localhost";
$username="sydorak";
$password="sydorak";
```

```
/* Ім'я бази даних */
$dbName="sydorakdb";

/* Таблиця MySQL */
$usertable="sydoraktab1";

 /* Створити з'єднання */
mysql_connect($hostname,$username,$password) or die("Не можу
під'єднатися ");

    /* Вибір БД */
mysql_select_db("$dbName") or die("Не можу вибрати БД");

/* Введення інформації в БД */
$query="INSERT INTO $usertable VALUES
('$user_name','$phone','$job','$img')";
$result=MYSQL_QUERY($query);
/* Закрити з'єднання */
MYSQL_CLOSE();
print "Запис введений в БД! <br>";
print "<a href='../php/'>Повернутись до меню</a>";
?>
```

3. ПРЕДСТАВЛЕННЯ РЕЗУЛЬТАТІВ У ЗВІТІ

3.1. Представити результати виконання лабораторної роботи і аналітичний матеріал відповідно п.п. 2.1-2.5.

3.2. Дати відповіді на контрольні запитання.

3.3. Короткі висновки за результатами роботи.

Контрольні питання

1. Дайте визначення елемента HTML, тегу, атрибутів.
2. Який з тегів HTML треба використовувати щоб зробити нумерований список?
3. Для чого використовується елемент BODY і які він має атрибути?
4. Як правильно вставляти запис в таблицю в php?
5. Для чого використовуються індекси в php?
6. Для чого призначена команда *CREATE*?

Список літератури

1. Пасічник В.В.Організація баз даних та знань: підручник для ВНЗ/ В.В. Пасічник, В.А. Резніченко.-К.: Видавнича група BHV,2006.-384с.
2. https://estuary.dev/database-triggers/
3. https://blog.devart.com/how-to-create-a-view-in-mysql.html#create-view-statement
4. А.Н.Наумов, А.М.Вендров і ін., "Системи керування базами даних і знань", М.:Фінанси і статистика, 1991р.
5. Гайдаржи В. І., Дацюк О. А. Основи проектування та використання баз данних : навчальний посібник / В. І. Гайдаржи, О. А. Дацюк. — [2 вид., виправл. і доповн]. — К. : Політехніка, 2004 . – 256 с.

Навчальне видання

Створення пакету програм на php і html для управління функціями бази даних

Методичні вказівки до лабораторної роботи №4 з курсу “Бази даних та знань” для студентів спеціальності 125 Кібербезпека та захист інформації //

Укладачі: Партика А.І, канд. техн. наук, асистент
Гарасимчук О.І., канд. техн. наук, доц.