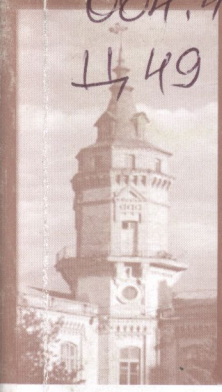


Національний технічний університет України «Київський Політехнічний Інститут»

НТУУ «КПІ»
1898

О. В. Цеслів

WEB-ПРОГРАМУВАННЯ

Міністерство освіти і науки, молоді та спорту України
Національний технічний університет України
«Київський політехнічний інститут»

О. В. Цеслів

WEB-ПРОГРАМУВАННЯ

Навчальний посібник

Київ
НТУУ «КПІ»
2011

Гриф надано Методичною радою НТУУ «КПІ»
(Протокол № 8 від 15.04.2010 р.)

Рецензенти: *О. Ю. Чубукова*, д-р екон. наук, проф.,
Київський національний університет технологій та дизайну

О. І. Стасюк, д-р техн. наук, проф.,
Державний економіко-технологічний університет транспорту

Відповідальний редактор *В. О. Капустян*, д-р фіз.-мат. наук, проф.,
Національний технічний університет України
«Київський політехнічний інститут»

Цеслів О. В.

Ц49 WEB-програмування [Текст] : навч. посіб. / О. В. Цеслів. – К.: НТУУ «КПІ», 2011. – 298 с.

Описано спектр технологій створення Web-документів, зокрема форматування тексту, створення списків, таблиць, графічних об'єктів, використання засобів інтерактивного спілкування з користувачем, баз даних. Розглянуто об'єктно-орієнтовані технології та програмування мовою PHP. Навчальний посібник зорієнтовано на розв'язування бізнес-задач, реалізованих у прикладах, які часто трапляються в дійсних розробках.

Для студентів спеціальності «Економічна кібернетика».

УДК 004.738.5(075.8)+330.46
ББК 004.738.5(075.8)+330.46

ЗМІСТ

РОЗДІЛ 2.

МОВА PHP

РОЗДІЛ 3

КЕРУЮЧІ СТРУКТУРИ

ВСТУП

На сучасному етапі розвитку ринкової економіки під впливом нових інформаційних технологій відбуваються докорінні зміни в технології управління. Компанії не представляють своєї роботи без Internet, електронної пошти, FTP, WWW. Всесвітня павутина дозволяє оперативно отримати необхідну інформацію про діяльність компанії. Звичайно, ця інформація повинна оперативно обновлюватися. В зв'язку з цим важливим завданням вищої школи слід вважати випуск спеціалістів, які можуть поєднувати знання економічних спеціальностей з широким використанням інформаційних технологій на практиці.

На сьогоднішній день опубліковано багато праць, присвячених практичному створенню WEB - сайтів [1-10]. Більшість цієї літератури [4,6,7,8,9], це довідники для професіоналів. Водночас майже відсутні роботи методологічного плану, навчальні посібники, які б систематизували теоретичні і практичні результати.

Даний посібник сформований за результатами викладання курсу лекцій студентам старших курсів факультету менеджменту та маркетингу Національного технічного університету України «Київський політехнічний інститут».

Посібник орієнтований на студентів спеціальності «Економічна кібернетика» при вивченні дисципліни «WEB - програмування». Автор сподівається, що матеріали посібника будуть корисними для студентів та аспірантів інших спеціальностей, які цікавляться сучасними інформаційними технологіями та їх практичним застосуванням.

Навчальний посібник складається з дванадцяти розділів, які слід вивчати послідовно. Перший розділ містить основні відомості про мову роз-

мітки сторінок HTML, без знання якої неможливо взагалі створення додатків в Internet.

У другому розділі ми приступимо до вивчення самого PHP, тут описується структура програми, синтаксис PHP, константи, змінні, вирази й основні конструкції мови. Третій розділ присвячений керуючим структурам PHP. При програмуванні на PHP ви дуже часто, будете зіштовхуватися з масивами. Як працювати з масивами, ви довідаєтеся в четвертому розділі.

У п'ятому розділі розглядається створення власних підпрограм — функції. Описані корисні стандартні функції PHP, які ви будете використовувати при створенні своїх сценаріїв. Шостий розділ присвячений регулярним виразам. Як працювати з файлами й каталогами, ви довідаєтеся в сьомому розділі. У восьмому розділі розглянутий механізм сесій й cookies, що допоможе вам при розробці складних сценаріїв.

Бібліотека GD дозволяє створювати на PHP різні спеціальні зображення: зображення для лічильників, рекламні банери, логотипи. Про все це ви прочитаєте в дев'ятому розділі.

Жоден серйозний Internet-проект не обходиться без баз даних. Сервер MySQL є одним із самих вдалих реалізацій SQL-серверів, тому не розглянути сервер баз даних, встановлений на комп'ютерах більшості хостинг-провайдерів, просто неможливо. Як написати сценарій, що працює з MySQL, ви довідаєтеся в десятому розділі.

Будь-який Internet-додаток по визначенню є мережевим. Мережні функції PHP розглянуті в одинадцятому розділі. Дванадцятий розділ присвячений публікації сайту на сервері та рекламі в Інтернет.

Автор висловлює глибоку вдячність рецензентам —д.е.н., професору О.Ю. Чубуковій, д.т.н., професору О.І. Стасюку за цінні поради, які сприяли покращенню посібника.

РОЗДІЛ 1. МОВА HTML

Ідея рішення проблеми обміну документами між різними комп'ютерами й додатками через Internet заснована на мові розмічування гіпертексту HTML (HyperText Markup Language). Ця мова була створений більше 15 років тому, як стандарт оформлення документів і була прийнята переважною більшістю користувачів Internet, а головне, - всіма виробниками програмного забезпечення й устаткування для Web. Документи, розмічені згідно HTML, можуть читатися на будь-якому комп'ютері, на якому встановлена всього лише одна програма перегляду таких документів - браузер.

Завдяки мові розмітки HTML клієнт Web може на екрані свого комп'ютера переглянути документ у тому вигляді, у якому його задумав розроблювач: з певними розмірами шрифту й розбивкою на абзаци, з потрібним розташуванням малюнків, гіперпосилань. Текстової документ, складений на HTML, має розмір у байтах у кілька разів менший, чим розмір аналогічного документа, підготовленого в текстовому процесорі (наприклад, Word).

Розмітка документа — це опис різних фрагментів документа і їхнього взаємного розташування. Виконується розмітка за допомогою символів ASCII, а точніше, арабських цифр, символів латинського алфавіту й деяких розділових знаків. Із цих символів набираються команди мови HTML - теги, або, інакше кажучи, дескриптори.

У мові HTML є безліч тегів, серед яких - теги створення заголовка документа, завдання параметрів шрифту, креслення ліній, вставки гіперпосилань, графічних елементів, звуків, відео.

Історія мов розмітки почалася в 60-і роки 20 століття, коли співробітники компанії IBM взялися за рішення завдань перенесення документів між різними платформами й операційними системами. Результатом їхніх

зусиль стала мова GML (General Markup Language - загальна мова розмітки), яка призначалася для використання на ЕОМ сімейства IBM. Мова GML надалі була розширена, а в 80-х роках пройшла стандартизацію в ISO (Міжнародна організація стандартизації). Ця потужна й універсальна мова розмітки, яка отримала назву SGML (Standard General Markup Language), використовувалася військовим відомством США при оформленні технічної документації. Однак SGML широкого поширення не одержала через свою складність і високу вартість.

Наступний етап розвитку мов розмітки пов'язаний з іменами вчених-фізиків, співробітників CERN у Женеві. Так, наприкінці 80-х років Тім Бернерс-Лі зайнявся проблемою зберігання й відображення даних, отриманих колегами. В основу нової мови Бернерс-Лі поклав мову SGML і прийоми роботи з гіпертекстом, так появилася мова - HTML.

Можливість застосування гіпертексту була запропонована Ваневаром Бушем в 1945 році, а термін «гіпертекст» уперше був введений Тедом Нельсоном в 60-х роках, тобто задовго до появи Internet. Поняття «гіпертекст» позначає електронний документ, що містить у собі посилання на інші документи.

Створення HTML привело до появи нової технології поширення гіпертекстових документів в Internet. Однак для широкого впровадження World Wide Web, крім мови HTML, потрібна була розробка протоколу передачі гіпертексту HTTP (HyperText Trasport Protocol - протокол передачі гіпертексту), що дозволив здійснювати обмін документами HTML. Саме цей протокол дав можливість додатку-клієнта знаходити й використовувати ресурси, що зберігаються на іншому комп'ютері.

Стандарт HTML 4.01. був затверджений в 1999 році. У посібнику ми будемо дотримуватися рекомендацій консорціуму W3C по застосуванню HTML-елементів і використанню каскадних таблиць стилів.

Гіпертекст — це електронний документ, який містить гіперпосилання на інші документи. ***Гіпертекстова технологія*** це інформаційна технологія, що базується на використанні гіпертекстів.

Декілька web-документів на одну тему, що є на деякому комп'ютері чи належать одному власникові, утворюють - web-сайт.

Web-дизайн - це сукупність правил і рекомендацій, якими мають користуватися автори, якщо вони хочуть, щоб їхні сторінки були інформативними і виглядали привабливо.

Одне з найважливіших правил web-дизайну полягає у структуризації інформації, вдалому поділі її на окремі частини і налагодженні зв'язків між ними. Згідно з чинним стандартом абзаци на класичній web-сторінці відокремлюються порожнім рядком і не мають відступів у першому рядку.

Сучасні редактори, такі як FrontPage, Dreamweaver, MS Word тощо, дають змогу створювати WEB - сторінки методом конструювання, тобто без застосував кодів мови HTML (Hyper Text Markup Language). Вони генерують цей код автоматично.

Для підготовки html-файлу можна використати текстовий редактор NotePad(Блокнот), що дає змогу готувати файли у текстовому форматі. Після написання html-файл потрібно зберегти його на диску з деякою назвою з розширенням назви html.

1.1.Тегова модель файлу

Команди мови HTML називаються *тегами.* Теги бувають одинарними і парними. Більшість тегів парні, як, наприклад, тег означення html-файлу: <HTML> ... </HTML>. Такі теги називаються інакше *контейнерами.* Контейнер може містити текст та інші теги.

Парні теги позначають початок і кінець області дії відповідної команди. Теги записують у кутових дужках. Тег, що закриває область дії, має косу риску. Не забувайте її писати, інакше тег працюватиме неправильно.

Тег може містити параметри, які користувач записує у першому блоці тега через пропуск чи з нового рядка, наприклад:

<BODY TEXT="red">...</BODY>.

Нечислові значення параметрів прийнято записувати у лапках.

У середині пари тегів <HEAD>...</HEAD> описують заголовок документа. Головна частина заголовка документа заголовок Windows-вікна, який пишуть у середині пари тегів <TITLE>...</TITLE>. Інші елементи заголовка вивчатимемо пізніше.

Тег <!— текст —> позначає коментар. Текст у цього тега виводитися на екран не буде. Коментар можна записати також у середині парного тега <COMMENT> текстовий коментар </COMMENT>.У середині пари тегів <BODY параметри >...</BODY> записують увесь текст сторінки. Цей текст відображатиметься у вікні браузера. Розглянемо головні параметри тега BODY:

Таблиця 1.1. **Параметри тега BODY**

BACKGROUND = "d:\myweb\1.bmp"	Задає шлях до картинки для тла.
BGCOLOR = "white"	Задає білий колір тла, якщо не використовується тло – картинка.
BGPROPERTIES="fixed"	Фонове зображення не прокручується.
TEXT = "black"	Задає колір тексту (тут чорний) на сторінці.

1.1.1. Форматування тексту для web-сторінок

Розглянемо теги, які використовують для форматування тексту. Теги форматування символів тексту (вони парні):

Таблиця 1.2. **Теги форматування символів тексту**

<B> текст </B>	Товстий шрифт тексту.
<I> текст </I>	Шрифт- *курсив.*
<U> текст </U>	Підкреслений шрифт.
_{текст}	Нижній індекс. Наприклад, щоб отримати вираз H_2O_2, пишуть H₂O₂.
^{текст}	Верхній індекс, наприклад, I^a, a^2.
<BIG> текст </BIG>	ВЕЛИКИЙ шрифт.
<SMALL>текст </SMALL>	Малий шрифт.
<EM> текст </EM>	Виокремлений *курсивом* текст (те саме, що тег <I>).
<B><I>текст</I></B>	*Товстий курсив.* Цей приклад демонструє застосування принципу вкладення тегів.

Теги для розміщення тексту (вони одинарні):

<P> Цей тег означає початок нового абзацу.

 Наступний за цим тегом текст буде наведено у новому рядку без пропуску рядка.

<HR> У рядку буде проведена горизонтальна лінія.

Заголовки і теги вирівнювання. Заголовок - окремий абзац. Є шість видів заголовків, які відрізняються розмірами символів:

Таблиця 1.3.**Теги заголовків**

Теги	Результат на екрані
<H1>Заголовок 1</H1>	Заголовок 1
<H2>Заголовок 2</H2>	Заголовок 2
<H3>Заголовок 3</H3>	Заголовок 3

<H4>Заголовок 4</H4>	Заголовок 4
<H5>Заголовок 5</H5>	Заголовок 5
<H6>Заголовок 6</H6>	Заголовок 6

Заголовок за замовчуванням вирівнюється до лівого краю вікна. Якщо вирівнювання заголовка чи іншого елемента *N* сторінці потрібно задати явно, то використовують теги вирівнювання:

Таблиця 1.4. **Теги вирівнювання тексту**

<CENTER> елемент </CENTER>	Вирівнювання до центру
<LEFT> елемент </LEFT>	Вирівнювання до лівого краю
<RIGHT> елемент </RIGHT>	Вирівнювання до правого краю

Ненумерований список утворюють за допомогою наступних тегів

<LH> Назва списку </LH>

<UL>

<LI>елемент 1

<LI>елемент 2

</UL>

Нумерований список створюють за допомогою парного тега <OL>... </OL> з необов'язковим параметром TYPE і одинарних тегів <LI> так:

<LH> Назва списку </LH>

<OL TYPE="1">

<LI> елемент 1

<LI> елемент2

</OL>

Значення "i" чи "I" параметра TYPE задає нумерацію римськими малими (i, ii, iii, iv, ...) чи великими (I, II, III, IV, ...) цифрами, а значення "a"

чи "A" - латинськими малими (a, b, c, d, ...) чи великими (A, B, C, ...) літерами.

Список тлумачень використовують для пояснення термінів, створення словників тощо. Його утворюють за допомогою парного тега <DL>...</DL> і двох одинарних тегів <DT>, <DD> так:

<LH>Заголовок</LH>

<DL>

<DT> термін

<DD> тлумачення 1

<DD> тлумачення 2

</DL>

Приклад 1.1. Створити файл з назвою file1.html.

```
<HTML><HEAD>
<tItle> Мій факультет</tItle>
<B><center><H1>Факультет менедженту та маркетингу </H1> </CENTER>
</B></HEAD>
<Body Bgcolor="yellow"><center>(Декан - проф.<B><I>Гавриш Олег Анатолійович </B></I>)</center>
<small><center><I>Навчальний корпус 1, кімн. 246</I></center> </small>
<small><center><I>Телефон для довідок: 441-18-86</I> </center> </small>
<font color="red">
<center><p><BR><I><LH>ЕКОНОМІКА І ПІДПРИЄМНИЦТВО</I>
</LH></BR></P> </CENTER></font>
<UL>
<LI>Економіка підприємства
<LI>Економічна кібернетика
<LI>Міжнародна економіка
<LI>Маркетинг
```

```
</UL>
<font color="red">
<LH><center><BR><I>МЕНЕДЖМЕНТ</I></BR></center></LH>
</font>
<UL>
<LI>Менеджмент організацій
<LI>Менеджмент зовнішньоекономічної діяльності
</UL>
<UL><left>
<BR><BR><LI>Факультет здійснює підготовку висококваліфікованих фахівців з менеджменту, економіки і підприємництва, які здатні приймати обґрунтовані рішення з управління персоналом, ринком, виробництвом, зовнішньоекономічною діяльністю та спроможні забезпечити ефективне функціонування національної економіки в умовах ринку.
<BR><BR><LI>Термін підготовки: <I>бакалавр</I>-4 роки;
<I>спеціаліст</I>-5років;<I>магістр</I>-5 років.
<BR><BR><LI>Випускники обіймають керівні посади в державних органах, підприємствах та фірмах, фінансових, страхових, консалтингових структурах ринкової економіки.
<BR><BR><LI>Абітурієнти складають вступні випробування з <B>математики, економічної і соціальної географії України, української мови та літератури.</B></UL></Body></HTML>
```

1.2. Створення таблиць

У звичайних текстових редакторах таблиці використовують для наочного подання числової чи текстової інформації. У web-дизайні таблиці відіграють велику роль. Часто їх використовують для позиціювання графічних чи інших об'єктів на екрані. Такі таблиці утворюють невидимими межами (рамками), а в клітинках розташовують картинки, тексти тощо.

Таблиці створюють за допомогою таких тегів:

<TABLE параметри>

<TC> Заголовок таблиці </TC>

Тут пишемо теги для заповнення клітинок таблиці рядок за рядком

</TABLE>

Для заповнення клітинок таблиці використовують такі теги (закриваючі теги можна не зазначати):

Таблиця 1.5. **Теги заповнення клітинок таблиці**

<TR>...</TR>	Формують рядок таблиці.
<TH>текст</TH>	Формують клітинку з заголовком рядка чи
<TB>текст</TB>	Формують текст кожної клітинки.

Заголовки рядків і стовпців виводитимуться товстішим шрифтом

Приклад 1.2. Створити файл з назвою file2.html

<HTML><HEAD>

<TITLE>Студентська поліклініка "КПІ"</TITLE></HEAD>

<BODY bgcolor="silvery">

<H2>

<LH>Студентська поліклініка НТУУ

КПІ</LH></H2>

<TABLE border=3 bgcolor="yellow" bordercolor="BLACK">

<TC> <h3><LH><I>Графік роботи лікарів факультетів (інститутів)</I></LH></h3> </TC>

<TR><H6>

<TH>Факультет</TH>

<TH>Прізвище, ім'я, по-батькові</TH>

```
<TH>каб</TH>
<TH>Режим роботи по днях</TH>
</H6></TR>
<TR><B><TH align="left">ВІТІ, ФЕА, ФПМ, ІПСА</TH></B>
<TD>БОГАЧЕВА Лариса Корніївна</TD>
<TD>3-05</TD>
<TD><BR>непарні 09.00-14.00</BR><BR>парні 14.00-
19.00</BR></TD>
</TR>
<TR><B><TH align="left">ІТС, ЗФ, ММІ, ІХФ</TH></B>
<TD>ДМИТРЕНКО Марія Броніславівна</TD>
<TD>3-10</TD>
<TD><BR>непарні 09.00-14.00</BR><BR>парні 14.00-
19.00</BR></TD>
</TR>
<TR><b><TH align="left">ФАКС, ФТІ, ФЛ, ФЕЛ, ІФФ</TH></b>
<TD>ВОРОНА Едуард Миколайович</TD>
<TD>3-07</TD>
<TD><BR>непарні 14.00-19.00</BR><BR>парні 09.00-
14.00</BR></TD>
</TR>
<TR><b><TH align="left">ФММ, ФП, ІЕЕ, ПБФ, Іноземні студенти
</TH></b>
<TD>ШИРАНТ Наталія Петрівна</TD>
<TD>3-07</TD>
<TD><BR>непарні 09.00-14.00</BR><BR>парні 09.00-
14.00</BR></TD>
</TR></TABLE></center></BODY></HTML>
```

1.3. Графічні об'єкти і гіперпосилання

Вставляння графічних і відео файлів. Графічні зображення, такі як фотографії, картинки, піктограми тощо, зберігаються на серверах в окремих файлах з розширеннями bmp, та іншими і відображаються на web-сторінці за допомогою тега <IMG> з параметрами:

<IMG SRC="адреса графічного файлу" ALT = "альтернативний текст" ALIGN="left" WIDTH=240 HEIGHT=200>

Дія тега. Обов'язковим є лише перший параметр SRC. Альтернативний текст - це текст, який виводитиметься на місці картинки, якщо браузер не може прийняти графічний файл або якщо режим відображення графіки вимкнено. ALIGN задає місце розташування картинки на екрані, а параметри WIDTH і HEIGHT - її розміри за шириною і висотою в пікселях або відсотках. Зображення можна подати в рамці. Графічний об'єкт можна створити в Paint.

1.3.1 Вставляння звуку і відеозображення

Важливо пам'ятати, що звукові файли мають розширення: au, wav, mid, midi, ra, а відеофайли - avi, vivo, mpeg. Щоб вставити звук чи відео, достатньо як значення параметра HREF у тезі гіперпосилання задати шлях до відповідного звукового чи відеофайлу, який вже є на диску, наприклад, так:

<A HREF="sound.wav">прослухайте лекцію </A>.

Текст прослухайте лекцію стане гіперпосиланням, клацнувши на якому можна почути лекцію, деяку інформацію, яка була заздалегідь записана.

Приклад 1.3. Створити файл з назвою file3.html

```
<html><head>
<title>Карта "КПІ"        </title></head>
<body bgcolor="Blue">
```


<h1>Карта НТУУ "КПІ" </h1><hr>

<IMG src="picture/map.gif" width="650" height="550" border="1">

</body></html>

1.4. Фрейми

Поняття про фрейми. Якщо матеріали web-сайту структуризовані логічно за темами і мають базуватися на декількох сторінках з навігацією за допомогою гіперпосилань, то такий сайт реалізовують із застосуванням фреймів. Фрейм у перекладі з англійської означає кадр, рамка, прямокутна область. Фрейми поділяють вікно браузера на частини, в яких відображають зміст сторінок сайту. Кожній сторінці має відповідати свій html-файл. Кожна сторінка повинна мати логічний заголовок.

1.4.1 Файлова структура сайту з фреймами

Для створення із застосуванням фреймів потрібно скласти як мінімум три html-файли: один основний і два чи більше допоміжні. Один допоміжний файл потрібний для заповнення стартовою інформацією лівого фрейму, інший – правого.

Гіперпосилання вставляють за допомогою парного тега <A параметр>...</A>, де параметр HREF = "адреса файлу". Гіперпосиланням може бути текст або картинка. Розглянемо випадок, коли гіперпосиланням є текст. Нехай у реченні "Я навчаюсь в КПІ" слово "КПІ" потрібно зробити гіперпосиланням на файл "file1.htm", який містить додаткові відомості про КПІ. Це роблять так:

Я навчаюсь в <A HREF = "file1.html"> КПІ </A>.

Приклад 1.4. Створити файл з назвою index.html

<HTML><HEAD>

<title>Це мій сайт з фреймами </title></HEAD>

<FRAMESET COLS="25% ,75% ">

<FRAME SRC="leftframe.html"

NAME= "left"

<!--або "лівий" або інша назва фрейму-->

SCROLLING="no"

<!--або "yes" або "auto"-->

FRAMEBORDER="1"

<!--або "0" межа фреймів є чи ні-->

BORDER = "15" <!--товщина межі->

MARGINHIGHT= "10"

<!-- відступи від країв->

MARGINWIDTH=" 10"

NORESIZE

<!--не можна пересувати межу-->

BORDERCOLOR = "red" >

<FRAME SRC = "rightframe.html"

NAME="right" <!--або "правий" тощо-->

</FRAMESET>

<NOFRAME>

<!--Текст, що відображатиметься у браузері, який не підтримує фреймів, наприклад: >

Вибачте, цей сайт містить фрейми. Скористайтесь іншим браузером для його перегляду. </NOFRAME></HTML>.

Допоміжний rightframe.html

<html><head><title>Це мій правий фрейм</title></head>

<body bgcolor="Blue">

<IMG src="picture/logo.gif" width="500" height="70"

```
        border="1"> </center>
    <font color="Yellow">
    <center><h1>НТУУ "КПІ" запрошує</h1><hr></center></font>
    </body></html>
```

Допоміжний leftframe.html

```
    <html><head>
        <title>Це мій лівий фрейм</title></head>
    <body bgcolor="Yellow">
    <font color="Blue">
    <H2>Сайт НТУУ "КПІ"</H2></font><br><br>
    <a href ="file5.html" target ="right">
    <IMG src="picture/1.gif" width="60" height="100"border="0"></a>
<br><a href ="file1.html" target="right">Мій факультет</a> <br>
    <a href  ="file2.html" target ="right">Поліклініка</a><br>
    </body></html>
```

Розглянемо ще один файл з фреймами.

Приклад 1.5. Файл з фреймами.

```
<HTML><HEAD>
<TITLE></TITLE>
</HEAD>
<FRAMESET
ROWS="25%,50%,25%">
<FRAME  SRC="1.HTML">
<FRAMESET COLS="25%,75%">
<FRAME SRC="2.HTML" >
<FRAME SRC="file1.HTML">
</FRAMESET>
```

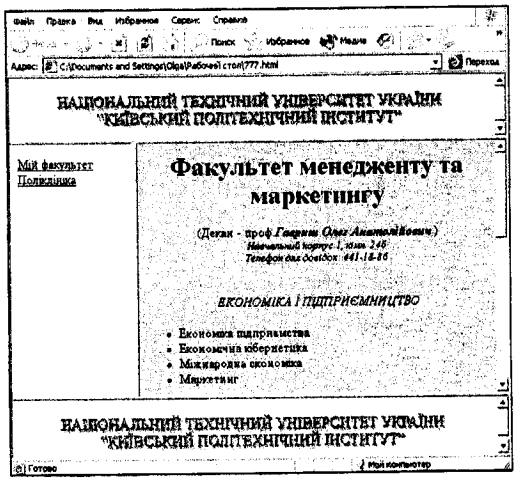


Рис 1.1. **Вікно з фреймами прикладу 1.5**

<FRAME SRC="1.HTML">
</FRAMESET>
</HTML>

1.4.2. Форми

Форми являють собою найбільш важливі інтерактивні елементи HTML. З їхньою допомогою користувач може повертати коментарі із приводу відвідування певного вузла, пересилати запити або реєструватися. Розроблювач задає питання створюючи форму, а користувач відповідає на них заповнюючи її. Зміст форми або передається сценарію CGI (*Common Gateway Interface*) - стандарт інтерфейсу, для зв'язування зовнішньої програми з веб-сервером; або по електронній пошті посилається розроблювачу. Сам процес створення форми складається із двох етапів. Перший полягає в створенні самої форми, а другий містить у собі створення на сервері сценарію CGI. Форма створюється за допомогою різних тегів й атрибутів, вкладених у пару тегів <FORM></FORM>.

Елемент <FORM>

Елемент **<FORM>** є необхідною умовою для всіх форм. Він може мати наступні атрибути:

- **method**

Цей атрибут визначає спосіб пересилання даних сценарію CGI. Тут протокол GET обраний за замовчуванням, але в більшості випадків розроблювачі користуються протоколом POST, що дозволяє передавати більші обсяги даних.

- **action**

Цей атрибут визначає шлях до сценарію CGI або адресу електронної пошти.

- **enctype**

Цей атрибут визначає спосіб кодування вмісту форми. Іншими словами

він повідомляє браузеру про спосіб кодування інформації перед відсиланням серверу. За замовчуванням використовується значення **x-www-form-encoded**.

Синтаксис форми для сценарію: **<FORM method="get" або "post" action="URL сценарію" ></FORM>.**

Синтаксис форми для пошти: **<FORM method="get" або "post" action="mailto:адреса" ></FORM>.**

Елемент <INPUT>

Елемент **<INPUT>** є базовим для всіх елементів форми. Він використовується для впровадження у форму кнопок, графічних зображень, прапорців, перемикачів, паролів і текстових полів. Незважаючи на зовнішні відмінності форм всі вони пересилають сценарію **CGI** дані у вигляді пари **ім'я: значення**. Елемент може мати вісім атрибутів, що позначуються як **type:**

- **TEXT**

Однорядкове текстове поле, використовується для введення інформації, яку не можна ввести в жодному з інших елементів форми. Сюди вводяться імена, адреси, посади, телефони, та дані практично будь-якого типу. Елемент може мати атрибути:

- **maxlength** – задає максимально припустиму довжину значення, що вписується, у символах;
- **size** – задає максимально припустиму довжину поля в символах;
- **value** – задає значення за замовчуванням, яке можна міняти.

Синтаксис :**<INPUT type="TEXT" name="Hobby" maxlength="35" size="20" value="Shopping">**

Shopping

- **PASSWORD**

Однорядкове поле, у якому замість символів, що вводять, відображаються зірочки. Елемент може мати атрибути:

- **maxlength** – задає максимально припустиму довжину значення, що вписується, у символах;
- **size** – задає максимально припустиму довжину поля в символах;
- **value** – задає значення за замовчуванням, яке можна міняти.

Синтаксис :**<INPUT type="PASSWORD" name="PASSWORD_BOX" maxlength="35" size="20">**

- **HIDDEN**

Ще один тип прихованого введення інформації. Дозволяє пересилати сценаріям інформацію, що не може бути змінена користувачем. Деякі програми **CGI** використовують приховані поля для передачі інформації з однієї сторінки в іншу, наприклад, ім'я або номер. Такий підхід істотно полегшує роботу користувача, рятуючи його від необхідності повторного введення даних. Наприклад для пересилання файлу з вихідним кодом **HTML** використовується наступна конструкція :

<INPUT type="HIDDEN" name="file" value="anyfile.html">

- **CHECKBOX**

Прапорці використовуються для надання можливості користувачеві відповісти односкладово: **так/немає істина/неправда більше/менше** й т.д. Ви глядає звичайно у вигляді хрестика або пташки. Елемент може мати атрибути:

- **checked** – задає початковий статус прапорця за замовчуванням;
- **value** – задає значення за замовчуванням, яке можна міняти.

Синтаксис:**<INPUT type="checkbox" name="send_mail"**

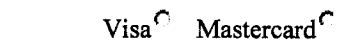
value="yes" checked>

- **RADIO**

Перемикачі багато в чому нагадують прапорці, відрізняючись лише більш широкими функціональними можливостями вибору. У групі перемикачів може бути обраний лише один. Для кожного перемикача вказується окремий елемент **INPUT**.

Приклад 1.6. Приклад використання перемикачів.

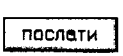
Visa <INPUT type="radio" name="payment_type" value="visa">
Mastercard <INPUT type="radio" name="payment_type"
value="mastercard">
American Express <INPUT type="radio" name="payment_type"
value="AmEx" checked>



- **SUBMIT**

Клацання на цій кнопці приводить до пересилання змісту форми сценарію, що був заданий атрибутом **action** в елементі **<FORM>**. За допомогою кнопок можна обчислювати суму, завантажувати сторінки, пересилати дані, скидати значення.

Синтаксис: <FORM method="get" або "post"
action="mailto:name@domen.ru" >
<INPUT type="submit" value="послати"></FORM>



- **RESET**

Кнопка використовується для відновлення значень, заданих за замовчуванням. Якщо значення за замовчуванням не передбачено, то воно просто

обнулиться. Ширина кнопки може мінятися залежно від інших елементів. Має так само атрибут **value**.

Синтаксис :**<INPUT type="reset" value="очищення">**

- **IMAGE**

Багато в чому схожий на кнопку **SUBMIT**, тільки як кнопку використовує зображення. Однією з переваг є можливість передачі координат миші користувача, що дозволяє організувати карту зображень. Елемент може мати атрибути:

- **src** – задає **URL**(адресу) файлу із зображенням;
- **align** – задає вирівнювання зображення щодо тексту за допомогою значень **TOP , MIDDLE** і **BOTTOM**;
- **name** –задає ім'я карти, яке так само пересилається сценарію разом з координатами.

Синтаксис :**<INPUT type="image" src="knopka.gif">**

- **BUTTON**

Створює іншу кнопку, браузери користувачів можуть використовувати значення атрибута **value** як вихідне ім'я файлу.

Синтаксис :**<INPUT type="button" value="кнопка">**

- **FILE**

Створює керуючий елемент – вибір файлу.

Синтаксис :**<INPUT type="file">**

- **ACCESSKEY**

Задає кнопку, при натисканні якої відбувається обробка поля.

Синтаксис :**<INPUT accesskey="a">**

- Приклад натисніть Alt+a:

- 

- **ID**

Задає ім'я для посилання.

Синтаксис :**<INPUT id="ім'я">**

- **SIZE**

Задає ширину елемента в пікселях.

Синтаксис :**<INPUT size="число">**

- **DISABLED**

Відключає можливість змінювати вміст поля або положення кнопки.

Синтаксис:**<INPUT disabled="Shopping">**

Елемент <TEXTAREA>

За допомогою цього елемента створюється область для введення й перегляду тексту. Він може використовуватися й не в складі форми, а як самостійна деталь сторінки. Область введення допомагає заощадити місце завдяки смугам прокручування. Може мати атрибути:

- **name** – задає ключове слово, по якому сценарій може звертатися до його змісту;
- **rows** – задає висоту області в рядках;
- **cols** – задає ширину області в символах.

Приклад 1.7. Синтаксис :<FORM><H3>Введи текст</H3>
<TEXTAREA name="ключове слово" rows=5 cols=30>Область для введення тексту
</TEXTAREA>
<INPUT type="reset" value="очищення"></FORM>

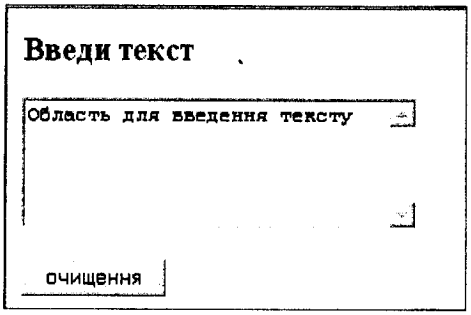


Рис.1.2 **Форма прикладу 1.7.**

Елемент <SELECT>

Елемент **<SELECT>** може приймати форму списку, що розкривається, або меню елементів. Має вкладений тег **<OPTION>** й атрибути :

- **name** – задає ім'я;
- **size** – задає максимальна кількість елементів списку, що одночасно відображаються на екрані;
- **multiple** – задає можливість одночасного вибору декількох значень.

Елемент <OPTIONS>

Елемент же **<OPTIONS>** задає можливі варіанти вибору меню **<SELECT>**

Синтаксис:**<OPTION value="n" selected>значення**

має атрибути:

- **selected** – задає вибране слово;
- **value** – задає значення вибраного слова для сценарію.

Приклад 1.8. Приклад використання меню.

```
<H3>Вибери потрібне</H3>
<SELECT multiple>
<OPTION value=a>Перший</OPTION>
```

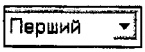
<OPTION value=b>Другий</OPTION>

<OPTION value=c>Третій</OPTION>

<OPTION value=d>Четвертий</OPTION>

</SELECT>

Вибери потрібне



Приклад 1.9. Приклад використання оператора SELECT.

<SELECT size=1>

<OPTION selected value=1>Виберіть:</OPTION>

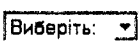
<OPTION value=2> Перший</OPTION>

<OPTION value=3> Другий</OPTION>

<OPTION value=4>Третій</OPTION>

<OPTION value=5> Четвертий</OPTION>

</SELECT>



Елемент <OPTGROUP>

Елемент **<OPTGROUP>** застосовується для логічного згруповування елементів **<OPTION>** усередині тега **<SELECT>** має атрибут **label**:

Синтаксис : <SELECT size=1>

<OPTGROUP label="Перша група» >

<OPTION selected value=1>Виберіть:</OPTION>

<OPTION value=2> Перший </OPTION>

<OPTION value=3> Другий </OPTION>

</OPTGROUP>

<OPTGROUP label="Друга група">

<OPTION value=4>Третій</OPTION>

<OPTION value=5> Четвертий</OPTION>

</OPTGROUP>

</SELECT>

Елемент <ISINDEX>

Це найпростіший елемент, що дозволяє створити подібність форми й уведення рядка, що містить текст і генерує запит.

Приклад : **<ISINDEX prompt="Рядок для уведення критерію пошуку" >**

Допустимо, що на поточній сторінці заданий базовий **URL** за допомогою елемента

<BASE href="URL пошукового засобу в Internet">

тоді, якщо користувач уведе в поле ключові слова для пошуку **слово1, слово2, слово3**, то браузер згенерує й відішле запит для пошукової машини сервера у вигляді:

http://www.назва.домен/?слово1+слово2+слово3.

Якщо пошукова програма сервера підтримує стандартний синтаксис запиту з використанням знаків ? і +, пошук буде здійснений.

Елемент <BUTTON>

Елемент **<BUTTON>** є альтернативою елементу **<INPUT>** з більшими можливостями – наприклад із завданням альтернативного тексту.

Синтаксис :**<BUTTON><IMG src=...> </BUTTON>**

- **name** – задає ім'я елементу;
- **value** – задає значення елементу;
- **type** – при використанні як кнопка приймає значення : **button, submit й reset**;
- **disabled** – робить недоступним даний елемент;
- **tabindex** – визначає положення в послідовності переходу клавішею **Tab**, відключені поля форм не беруть участі у черговості;

- **accesskey** – задає клавішу доступу;
- **id** – задає ім'я для посилання.

Приклад: **<BUTTON name="submit" type="submit">відправити </BUTTON >** послати

Елемент <LABEL>

Елемент **<LABEL>** застосовується для альтернативного завдання інформації для керуючих полів форми. Підтримує атрибут **for**, що зв'язує елемент **<LABEL>** з іншими елементами форми. Значення атрибута **for** повинне збігатися зі значенням атрибута **id** зв'язаного керуючого елемента.

Приклад 1.10. Використання елемента <LABEL>.

```
<FORM action="URL" method="post">
 <TABLE>
 <TR>
 <TD><LABEL for="fname">Ім'я</LABEL>
 <TD><INPUT type="text" name="firstname" id="fname">
 <TR>
 <TD><LABEL for="lname">Прізвище</LABEL>
 <TD><INPUT type="text" name="lastname" id="lname">
 </TABLE>
 </FORM>
```

Приклад 1.11. Приклад створення форми.

```
 <FORM action="URL" method="post">
<P>
 <LABEL for="firstname">Ім'я: </LABEL>
 <INPUT type="text" id="firstname"><BR>
 <LABEL for="lastname">Прізвище: </LABEL>
 <INPUT type="text" id="lastname"><BR>
```

<LABEL for="email">email: </LABEL>
<INPUT type="text" id="email">
<INPUT type="radio" name="sex" value="Чоловічий">Чоловічий
<INPUT type="radio" name="sex" value="Жіночий">Жіночий
<INPUT type="submit" value="Відправити"> <INPUT type="reset">
</P></FORM>

Елемент <FIELDSET>

Елемент **<FIELDSET>** дозволяє логічно згруповувати елементи форми.

Синтаксис:**<FIELDSET> ім'я </FIELDSET>**

Елемент <LEGEND>

Елемент **<LEGEND>** дозволяє давати найменування логічним групам елементів форми.

Синтаксис: **<LEGEND>ім'я</LEGEND>**

Приклад 1.12. Використання тегу <LEGEND>.

<FORM>
<FIELDSET>
<LEGEND>Група 1</LEGEND>
<INPUT type="text" id="name1">
<INPUT type="text" id="name2">
</FIELDSET>
<FIELDSET>
<LEGEND>Група 2</LEGEND>
<INPUT type="text" id="name3">
<INPUT type="text" id="name4">
</FIELDSET></FORM>

Приклад 1.13. Створення форми реєстрації учасників конференції.

```
<HTML><HEAD><TITLE>ФОРМИ</TITLE>
<meta http-equiv="Content-Type" content="text/html;
  charset=windows-1251"></HEAD>
<BODY>
<H2>РЕЄСТРАЦІЯ УЧАСНИКІВ КОНФЕРЕНЦІЇ</H2>
<FORM METOD="GET">
ВВЕДІТЬ ІМ'Я ДЛЯ РЕЄСТРАЦІЇ  <INPUT TYPE="TEXT"
NAME="REGNAME"><p>
ВВЕДІТЬ ПАРОЛЬ: <INPUT TYPE="PASSWORD"
NAME="PASSWORD1" MAXLENGTH=8><p>
ПІДТВЕРДЖЕННЯ ПАРОЛЯ:  <INPUT TYPE="PASSWORD"
NAME="PASSWORD2" MAXLENGTH=8><p>
ВАШ ВІК<br><INPUT TYPE="RADIO" NAME="AGE" VALUE="LT20"
CHECKED> ДО 20
<INPUT TYPE="RADIO" NAME="AGE" VALUE="20_30" CHECKED>20-
30
<INPUT TYPE="RADIO" NAME="AGE" VALUE="30_50"
CHECKED>30_50
<INPUT TYPE="RADIO" NAME="AGE" VALUE="ПЕ50" CHECKED>
СТАРШЕ 50<BR><BR>
ЯКИМИ МОВАМИ ВОЛОДІЄТЕ
<INPUT TYPE="CHECKBOX" NAME="LANGUAGE" VAL-
UE="RUSSION" CHECKED> РОСІЙСЬКА
<INPUT TYPE="CHECKBOX" NAME="LANGUAGE" VAL-
UE="ENGLISH" CHECKED> АНГЛІЙСЬКА
<INPUT TYPE="CHECKBOX" NAME="LANGUAGE" VALUE="FRENCH"
CHECKED> ФРАНЦУЗСЬКА
```

```
<INPUT TYPE="CHECKBOX" NAME="LANGUAGE" VAL-
UE="GERMAN" CHECKED> НІМЕЦЬКА<BR><BR>
ЯКИЙ ФОРМАТ ДАНИХ ДЛЯ ВАС НАЙКРАЩИЙ
    <BR>
    <SELECT NAME="FORMAT" SIZE=2>
    <OPTION SELECTED VALUE="HTML">HTML
    <OPTION  VALUE="PLAIN TEXT">PLAIN TEXT
    <OPTION  VALUE="POSTSCRIPT">POSTSCRIPT
    <OPTION VALUE="PDF">PDF
    </SELECT>
    <BR><BR>НАЗВА ВАШИХ ТЕЗІВ<BR>
    <TEXTAREA NAME="WISH" COLS=40 ROWS=3>
    </TEXTAREA><BR><BR>
    <INPUT TYPE="SUBMIT" VALUE="OK"><INPUT TYPE="RESET"
VALUE="ОТМЕНИТЬ"></FORM></BODY></HTML>
```

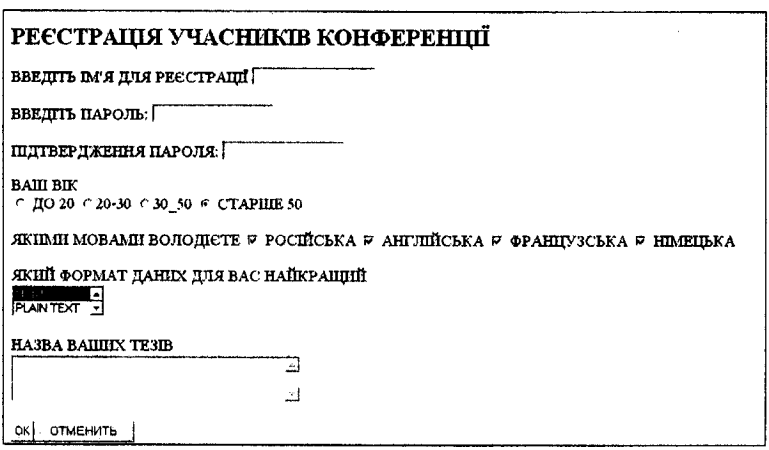


Рис.1.3.**Форма реєстрації учасників конференції прикладу 1.13**

1.5. Таблиці стилів

У попередніх розділах ви познайомилися із простими засобами, підготовки тексту для Web-сторінок - це створення абзаців, заголовків, назви шрифтів і т.д. Однак порівняння цих скромних засобів з можливості спеціалізованих програм підготовки текстів (текстових процесорі видавничих систем) говорить явно не на користь HTML.

Нагадаємо, що при роботі в текстовому процесорі (наприклад, Word) ви можете довільним чином форматувати текст. Причому можливо переносити атрибути формату з одного документа в інший. Це можливо завдяки тому, що своє бачення документа ви можете сформулювати мовою стилів. У міру необхідності можна створювати й редагувати стилі, які визначають розміри, відбиття абзаців, міжрядкові інтервали й т.д.

Наблизитися до подібних технологій при створенні Web-сторінок дозволяють таблиці стилів, які додаються до звичайних HTML-файлів. Можливості таблиць стилів навіть більше широкі, чим можливості стилів звичайного текстового процесора. Користуючись таблицями стилів, розроблювач може з точністю до пікселя визначити положення на сторінці будь-якого об'єкта, підняти накладення одного об'єкта на інший, автоматизувати застосування стилів до об'єктів і класів.

1.5.1. Поняття про каскадні таблиці стилів

Стандарти таблиці стилів для Web були розроблені консорціумом W3C в 1995-96 роках і були названі *Cascading Style Sheets (каскадні таблиці стилів),* або скорочено *CSS.* Присутність слова «каскадні» у цьому тексті означає, що таблиці стилів дозволяють створювати ієрархію стилів. В HTML-документ представляється як структура вкладених елементів різного рівня, у якій стиль елемента більше низького рівня має пріоритет перед стилем зовнішнього елемента високого рівня.

1.5.2. Підтримка таблиць стилів

Таблиці стилів дозволяють звільнитися від деяких специфічних елементів HTML, які консорціум W3C уважає застарілими й рекомендує використати замість них властивості стилів з CSS. Ситуація ускладнюється тим, що існують кілька типів таблиць стилів. Сьогодні все популярнішою стає розширювана мова стилів XSL (eXtensible Style Language), що використовує синтаксис мови XML (eXtensible Markup Language - розширювана мова розмітки).

Популярні браузери Internet Explorer й Netscape, починаючи з версій 4.0, забезпечують підтримку таблиці стилів *CSS1 (Cascading Style Sheets Level 1 – таблиці каскадних стилів, рівень І),* хоча й у різному ступені. Вже існує друга версія таблиць стилів *CSS2.*

HTML-документ являє собою безліч вкладених один в одного елементів(наприклад, елементи BODY, P, HI, H2 й інші). Для кожного елемента передбачений припустимий набір властивостей, що визначається специфікацією HTML. У коді HTML прийнято властивості записувати у вигляді атрибутів, які вказуються в початковому тегу кожного елемента, (наприклад, гарнітуру шрифту конкретного абзацу, кольори тла документа).

Властивість стилю *- це параметр стильового оформлення документа, що визначається специфікацією CSS.*

В таблицях стилів значення властивості приєднується до нього за допомогою двокрапки. Наприклад, призначення червоних кольорів задається записом:

color:red

а розмір шрифту

font-size:14pt

Значення властивостей стилю записуються, без лапок. Однак якщо значення складається з декількох слів із пропусками (наприклад, найменування шрифту), тоді лапки (одинарні ' або подвійні ") ставляться.

1.5.3. Визначення вбудованого стилю (атрибут style)

Перш ніж почати розглядати використання CSS, зупинимось на найпростішому способі завдання стилю елемента HTML через атрибут style. Ви можете в процесі розробки сторінки швидко змінити стиль елемен додавши до тегу атрибут style із вказівкою необхідних властивостей. Допустимо, ви хочете, щоб абзац був набраний шрифтом розміром 6pt. Для цього потрібно записати в такий спосіб:

<P style="font-size: 6pt ">

При відображенні елемента P браузер відповідно до цього запису замінить розмір основного шрифту, прийнятого за замовчуванням, на шрифт розміром 6pt. Наведений спосіб форматування називається *вбудованим стилем.*

Вбудований стиль можна застосовувати до рядкових елементів (наприклад SPAN). Наступний код привласнює червоний кольор одному слову в заголовку:

<H1><SPAN style="color:red"> Захист </SPAN>комп'ютерних мереж</H1>

Ви можете встановити вбудований стиль для блоку, при цьому властивості стилю будуть застосовані до всіх елементів, що входять у блок.

Приклад 1.14. Використання вбудованого стилю для блока.

```
<html>
<DIV style="font-family:serif;color:brown">
 <h1>Частина перша</H1> </h1>Розділ 1</h1>
<h3>ТАБЛИЦІ СТИЛІВ</h3> Стандарти таблиць стилів для Web були розроблені консорціумом W3C <BR><BR>
```

</DIV></html>

У всіх вкладених елементах H1, H2, H3 буде використаний шрифт сімейства serif коричневих кольорів. Значення атрибута style обов'язково обрамляють в лапки (одинарні ' або подвійні "), причому властивості відокремлюють один від одного крапками з комами (пробіли не обов'язкові).

Вбудовані елементи. Елементи, які не форматуються як блокові, є вбудованими елементами.

Приклад 1.15. Вбудовані елементи.

```
<html>
<P>В абзац можна розміщувати
<B style="background-color:
red"> вбудовані елементи. </B>
Їх може бути
</style>
<EM style="background-color:
lightsteelblue"> декілька </EM>.</P>
<B style="background-color: red">так</B>
</style></html>
```

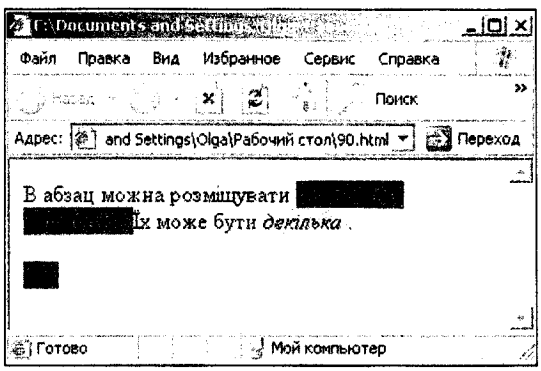


Рис.1.4. Вікно прикладу 1.12

1.5.4. Формування таблиць стилів. CSS-правила

Відповідно до специфікації CSS стильові властивості вводяться за допомогою *визначення стилю,* що прийнято позначати фігурними дужками. Наприклад, призначення курсивного накреслення тексту виконується за допомогою визначення:

{font-style:italic}

В одному визначенні стилю може записуватися кілька властивостей. Властивості перераховуються через крапку з комою, наприклад:

{color:blue;font-family:Arial;text-align:right}

Відповідно до специфікації CSS призначення стилю тому або іншому елементу HTML здійснюється так:

Елемент{ Визначення стилю }

Наприклад, якщо ви хочете, щоб всі елементи абзаців P відображалися шрифтом Arial розміром 14 pt, запишіть наступний рядок:

P{font-family:Arial; font-size: 14pt}

Приведемо ще один приклад - це призначення стилю елементу, позначеному в документі ідентифікатором id="f1". Нехай цей елемент уводить у документ малюнок, і ви хочете, щоб останній мав певні розміри: 150 пікселів завширшки й 100 пікселів у висоту. Це досягається за допомогою наступного коду:

#f1{width: 150; height:100}

Елемент, до якого ставиться обумовлений стиль, узагальнено називається *селектором.* У попередніх двох прикладах селектори були представлені: HTML - елементом P й ідентифікатором f1. Селектори, що збігаються з іменами тегів, позначають *класи,* а селектори, що починаються зі знака #(«рамка»), відповідають *унікальним ідентифікаторам* елементів.

Селектори позначаються по імені елементів, до яких застосовується визначення стилю. Наприклад, селектор TABLE указує на те, що визначення стилю ставиться до елементів Table. Селектори нечутливі до регістра, однак ми будемо дотримуватися прийнятої в CSS запису елементів прописними буквами.

Запис, що складається із селектора й визначення стилю, називається CSS-правилом. Таблиці стилів - це фактично набори CSS-правил, які задають властивості форматування елементів документа.

1.5.5. Внутрішні й зовнішні таблиці стилів

Раніше ми розглянули найпростіший спосіб стильового форматера – вбудовані стилі, що мають істотний недолік - вони не дозволяють відокре-

мити засобів форматування документа від його змісту. Крім того, оголошення вбудованого стилю доводиться повторювати для кожного документа, що форматується, протягом усього Web-сайту.

Від цих недоліків вільний інший спосіб введення стилів — це розміщення таблиці стилів у заголовній частині документа (в елементі HEAD). Заголовний стиль, дає можливість керувати оформленням документа.

Вбудовані й заголовні стилі належать відповідно до специфікації CSS до ***внутрішніх таблиць стилів.*** *Це назва вказує на те, що визначення стилів виконується в рамках одного документа. У той же час можливе завдання стилів у файлі с розширеннями .css або jss. Такий спосіб визначення називається* ***зовнішніми таблицями стилів.***

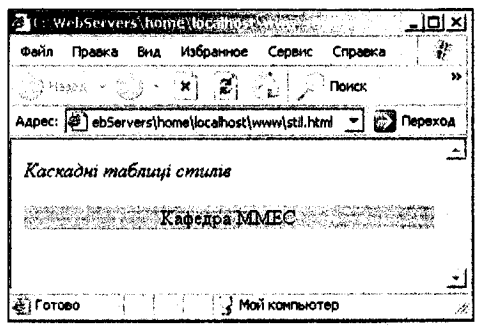


Рис.1.5 Вікно прикладу 1.16

Приклад 1.16. Зв'язування таблиць стилів з документом.

Створюємо файл "my.css"

```
<STYLE TYPE="text/css">
    <!-i
    I{text-transform: uppercase;
    background-color: Aqua;
    }
    P{background-color: #F08080;
    text-align : center;
    }
    -i>
    </STYLE>
```

Зв'язування дозволяє зберігати таблицю стилів в окремому файлі й приєднувати її до документів за допомогою тега, що задає в розділі <head>

```
<html><head>
<LINK REL="stylesheet" TYPE="text/css" href="my.css">
</head><body>
<I>Каскадні таблиці стилів</I>
<P>Кафедра ММЕС</P>
</body></html>
```

1.5.6. Введення заголовного стилю (елемент STYLE)

Як відзначалося вище, CSS являє собою мову опису інформації про стиль форматування елементів HTML-документа. Тому стосовно HTML-документа таблиці стилів CSS є сторонніми елементами. Для розміщення інформації про стиль застосовується спеціальний контейнер <STYLE>, що вставляється в заголовну частину документа й має вигляд:

```
<STYLE type="text/css">
<!— Опис таблиць стилів —> </STYLE>
```

У цьому коді значення text/css атрибута type повідомляє браузеру, що в поточному файлі застосований текст мовою CSS. Всі формальні описи стилів, що являють собою CSS-правила, містяться в елементі STYLE.

Пояснимо визначення стилів у наступному прикладі. Всі елементи будуть відображені шрифтом розміру 15pt, а деякі елементи H2, виділені в клас curs, будуть представлені курсивом з розміром 16pt. У той же час елемент, позначений ідентифікатором id="uk", буде виведений білим шрифтом на зеленому тлі.

Поєднання CSS-правил з конкретними елементами документа виконується за допомогою атрибутів class або id. Наведений приклад таблиць стилів може застосовуватися до такого елемента BODY.

Приклад 1.17. Внутрішні таблиці стилів.

```
<head><title> Внутрішні таблиці стилів</title>
 <STYLE type="text/css">
<!— Нижче  опис стилів —>
h2     {font-size:15pt}
H2.curs   {font-size:16pt;font-style.italic}
#uk {background-color:green;color:white}
</STYLE>
<!- Кінець опису стилів >
</HEAD>
<body>
<h2>Заголовок н2</h2>
<h2 class=”curs”>Перший заголовок класу curs</H2>
<h2 id="uk">Заголовок uk    </H2>
<h2 class="curs">Другий заголовок класу curs</H2>
</body>
```

1.5.7. Види селекторів

У специфікаціях CSS визначено кілька видів селекторів, основні з них використані в попередньому прикладі. Розглянемо можливі види селекторів докладніше.

Селектори типу

Селектор типу встановлює відповідність між визначенням стилю й типом HTML елементів, до яких цей стиль буде застосований. Наприклад:

P {font-family:arial; font-size:14pt}

Використається селектор типу й задає шрифт для всіх абзацних блоків HTML, а інше правило:

H2 {font-family:monospace; background-color:magenta}

визначає для всіх заголовків другого рівня H2 моноширинний шрифт, пурпурне тло.

Селектори класу

HTML-елемент або група елементів, оформлених одним стилем, утворюють клас. Клас може позначатися так само, як і тип HTML-елемента, наприклад абзаци документа утворюють клас P. Однак частіше класу привласнюється формальне ім'я, що надалі буде пов'язане з певним набором стильових властивостей. Допустимо, ви можете заголовки першого рівня H1 кольори згрупувати в клас із ім'ям blue:

H1.blue {color:blue}

Ім'я класу відокремлюється від імені елемента крапкою. Інший приклад завдання класу:

H2.curs {font-size:16pt;font-style-.italic}

Як бачимо, до складу класу входять не всі елементи одного типу H2, а тільки ті, які форматуються однаковим стилем. У той же час тому самому класу можуть належати різнотипні HTML- елементи. Допустимо, ви хочете задати відображення різних елементів (абзаців P, DIV і т.д.) на жовтому тлі. Для цього в список стилів уводиться CSS-правило із крапкою перед ім'ям класу, але без вказівки назви елементів, наприклад

.bgyellow {background-color:yellow}

Селектор .bgyellow позначає клас, а вираз у фігурних дужках – визначення стилю, що буде ставитися до будь-якого елемента bgyellow. Приналежність HTML-елемента цьому класу повинна бути зазначена в атрибуті class, наприклад:

<DIV class="bgyellow">

<P class="bgyellow”>

<SPAN class="bgyellow">

Селектори, що ставляться до будь-якого елемента документа, у специфікації прийнято називати *універсальними селекторами.*

Селектори id

При розробці Web-сторінки часто виникає необхідність змінити стиль тільки одного елемента. Простіше всього це зробити в такий спосіб. Задайте за допомогою селектора - id індивідуальний стиль, і цей стиль буде автоматично привласнений конкретному елементу, позначеному ідентифікатором id.

Наприклад, щоб виділити червоними кольорами текст абзацу, позначеного ідентифікатором id= "red", можна скористатися наступним CSS-правилом: #red {color:FF0000}. Потрібний елемент абзацу повинен містити одноіменне значення атрибута id.

Приклад 1.18. Приклад використання селектора.

```
<head>
<style type="text/css">
 #red {color:red}
 </style></head>
<body>
<p id="red"> абзац буде відображений червоним шрифтом </p>
</body>
```

В одному документі цей атрибут з певним значенням може зустрічатися тільки один раз.

1.6. Значення властивостей

Розрізняють числові й символьні значення властивостей.

Числові значення застосовуються для завдання розмірів, наприклад, ширини блоку, розміру шрифту, товщини рамки, міжрядкових інтервалів. Значення виражається десятковим числом, за яким звичайно слідує розмірність. Розмірність записується після числа без пробілу. Наприклад, 8pt,

1cm, 2.5in, 130%. У випадку негативного значення перед числом ставиться знак мінус (наприклад, -15px). Числові значення можуть виражатися в абсолютних або відносних одиницях.

Абсолютні значення задають точний розмір елемента й приводяться в стандартних одиницях виміру довжини, наприклад, у дюймах, сантиметрах або міліметрах.

Відносні значення визначають розмір елемента щодо іншого, елемента. Наприклад, ширина зображення може виражатися у відсотках щодо ширини блоку, у який вкладений малюнок, або щодо розміру вікна браузера. Міжсимвольний інтервал часто задається в одиницях ***em*** ширини символу основного шрифту (букви «m»). Інші одиниці відносних значень наведені в таблиці.

Таблиця 1.6. **Одиниці відносних значень**

Умовне позначення	Найменування одиниці виміру	Приклад
Абсолютні одиниці		
in	дюйм	width:.25in
cm	сантиметр	height:1.5cm
mm	міліметр	margin-left: 12
pt	пункт (1pt= 1/72in)	font-size:16pt
pc	пік(1pc= 12pt)	Line-height:1.2pc;
Відносні одиниці		
px	піксель	left:200px
em	ширина букви «т»	letter-spacing:
ex	висота букви «х»	font-size:2ex
%	відсоток	width:150%

Символьні значення складаються з букв латинського алфавіту або з комбінацій букв, цифр і спеціальних символів. Форма запису визначається згідно правилам CSS. Символьні значення привласнюються багатьом стильовим властивостям наприклад, властивостям шрифтів font-family (гарні-

тура шрифту) і style (стиль виведення шрифту), властивості тексту text-align (вирівнювання). Іншим прикладом використання символьних значень є властивості кольорів color й background-color, значення яких задаються в символьному форматі RGB-моделі. Відзначимо, що в CSS для опису кольорів застосовуються й інші формати RGB-моделі, які не підтримуються в HTML: десятковий формат (наприклад, rgb (128,0,128)) і процентний формат (rgb(50%,0,50%)).

1.6.1. Властивості шрифтів

В CSS передбачена безліч властивостей для керування шрифтами (завдання розміру, накреслення й т. д). Ці властивості можна призначати разом із властивостями, що визначають кольори шрифту, тло, відступи, міжсимвольну відстань. Розглянемо коротко тільки *властивості шрифтів:*

font-family - задає гарнітуру шрифту, що буде використана для виведення тексту. Значенням цієї властивості може бути назва конкретного шрифту (наприклад, Arial) або назва сімейства шрифтів;

font-size- розмір шрифту;

font-style - задає стиль виведення символів. Можливі наступні значення властивості: normal (звичайний), italic (курсив) і oblique (похилий);

font-weight - визначає ступінь «жирності» шрифту. Для цього використовуються цілі числа з діапазону від 100 до 900 із кроком 100 одиниць. Однак застосовують ключові слова: bold (жирний), bolder (більше жирний) і lighter (більше тонкий);

font-variant - вказує варіант накреслення поточного шрифту. Для цієї властивості браузери підтримують тільки два значення: small-caps (відображення малими прописними буквами) і normal (не впливає на відображення).

1.6.2. Властивості тексту

Зрозуміло, що задати шрифт - ще не значить визначити зовнішній вигляд тексту. Необхідно також вказати властивості, які відповідають за міжрядковий інтервал, відстань між словами й буквами й т.д. Ці властивості, називається ***властивостями тексту***, визначаються для абзаців і навіть для всього документа, тобто задаються на рівні блоків (P, DIV, BODY, SPAN, STRONG й ін.). Перелічимо властивості тексту, передбачені специфікацією CSS:

letter-spacing - встановлює відстань між буквами (міжрядковий інтервал). За замовчуванням цій властивості привласнюється значення normal. Можна вказувати відстань у будь-яких абсолютних одиницях;

word-spacing - встановлює відстань між словами. Аналогічно letter-spacing значенням цієї властивості за замовчуванням normal. Інші значення можуть бути задані в абсолютних одиницях (наприклад, 10px,2mm);

text-indent - задає відступ першого рядка абзацу (новий рядок) властивість застосовується до блоків, і її значення визначається в абсолютних одиницях (наприклад, 4mm, 1cm, 20pt і т.д.) або у відсотках від ширини (наприклад, 5% або 10%). За замовчуванням значення властивості text-indent дорівнює нулю. Якщо цій властивості привласнити негативне значення, то замість абзацного відступу одержимо виступ першого рядка;

text-align - задає горизонтальне вирівнювання для тексту, розміщеного усередині елемента (наприклад, P або DIV). Ця властивість приймає: center (по центру), left (уліво), right (вправо) і justify (по ширині). За замовчуванню текст вирівнюється по лівому краю. Ефект вирівнювання помітний при великому розмірі шрифту й малій ширині вікна браузера;

vertical-align - встановлює розташування тексту й малюнків по вертикалі щодо базової лінії. Властивість може приймати, значення: baseline (вирівнювання по базовій лінії, приймається за замовчуванням), sub (вирі-

внювання по лінії нижнього індексу), super (вирівнювання по лінії верхнього індексу);

line-height - визначає міжрядковий інтервал. Значення його можна задавати в абсолютних одиницях (наприклад, 16pt, 3mm), у відсотках (130%), а також кількістю рядків.

Як приклад спільного використання властивостей тексту приведемо HTML-код сторінки, що містить блокові й рядкові елементи з різними значеннями параметрів форматування.

Приклад 1.19. Властивості шрифтів і тексту.

```
<html><head><title> Властивості шрифтів і тексту </title>
<body>
<p style="font-family;helvetica Cyr; font-size:12pt;font-style:italic">
Абзац, що відтворений курсивним шрифтом Helvetica, що має розмір 12pt.
<p style="font-family:Helvetica Cyr; font-size;larger; font-variant:small-caps">
Цей абзац відтворюється шрифтом Helvetica, що має розмір larger і накреслення small-caps.
<p style="font-wight: bold">
  Це приклад набору формули, що має верхній і нижній індекси: S
<SPAN style="vertical-align:sub" >
result
 </SPAN>
 =Ax
<span style="vertical-align:super"> 2</span>.<br>
<style="font-weight:bold; text-indent: 12mm">
```

Даний абзац має відступ 12 мм, текст відображається напівжирним шрифтом

Times New Roman

причому назва шрифту набрана з розрядкою – міжсимвольною відстанню 3pt.

Цей абзац демонструє стильові можливості по відображенню

&laquonaїхавших» рядків

Розмір шрифту large, міжрядковий інтервал 9pt.

Даний абзац має міжрядковий інтервал 20pt і відображається шрифтом 12pt.

Перший рядок має &laquoвиступ» 15 мм.

Щоб цей &laquoвиступ» помістився у вікно браузера

для блоку P задане ліве поле 15 мм.

</body></html>

1.6.3. Властивості кольорів і тла

Відзначимо, що вживання стильової властивості background-color у цьому випадку еквівалентно використанню атрибута bgcolor для елемента BODY. Замість назви властивості background-color в таблицях можна вказувати його скорочення background, наприклад,

Style="background:red".

background-image - визначає вставку фонового зображення. Значенням властивості є URL(адреса) малюнка;

background-repeat призначає повтор фонового малюнка. Ця властивість застосовується, якщо розмір малюнка менше видимої області елемента. Щоб малюнок повторювався по горизонталі й по вертикалі, задавати

властивість background-repeat не потрібно (вона за замовчуванням має значення repeat). Для повторення малюнка тільки по горизонталі вибирається значення repeat-x, тільки по вертикалі - repeat-y, а для відключення повторення no-repeat.

Приклад 1.20. Приклад фонового зображення.

```
<html><head>
<title>Фоновий малюнок </title>
<STYLE type="text/css">
        h1 {color:red}
        body{background-
image:url(telecom.jpg) ;
        background-color:yellow;
        background-repeat:no-repeat}
</STYLE>
</head><body>

<h1>Приклад фонового зображення</h1>
</body></html>
```

Рис.1.6. **Вікно прикладу 1.20**

Код документа з фоновими малюнками, що повторюються уздовж осі відрізняється від наведеного тільки заміною значення властивості background-repeat на repeat-x.

1.6.4. Властивості блоку

Текстові HTML-елементи можна представити у вигляді прямокутних блоків. До таких елементів належать P, DIV, і навіть тіло документа Body. Для них в CSS передбачена спеціальна група властивостей, що дозволяє задавати поля, границі, відступи, розміри блоку. Розглянемо окремо кожну із властивостей блоку:

margin - це ключове слово позначає набір властивостей, що визначають кожне із чотирьох полів навколо блоку. У набір входять властивості margin-top (верхне поле), margin-right (праве поле), margin-bottom (нижнє поле), margin-left (ліве поле);

border - позначає набір властивостей відображення границі елемента. Розрізняють властивості border-style (керування виведенням границі), border-width (ширина границі), border-color (коліри границі). Властивості границь можна застосовуватися не тільки до блокових елементів, але й до рядковим елементів (наприклад, SPAN). При цьому рядковий елемент автоматично перетворюється в блок, відділений від іншого тексту порожніми рядками.

Значеннями властивості border-style є dotted (границя з точок), dashed (пунктирна границя), solid (звичайна суцільна границя), double (подвійна лінія), groove («утоплена» лінія границі), ridge (опукла границя), insert (об'єкт, «втиснений» у сторінку), outset (опуклий; об'єкт).

Ширину границі (властивість **border-width)** можна задавати як в абсолютних одиницях (наприклад, 3px), так і за допомогою ключових слів: thin (тонка), medium (середня) і thick (товста). Кольори границі (властивість **border-color)** визначається символьною назвою кольорів, кодом #RRGGBB, у двійковому форматі (наприклад, rgb (56,28,18)) або процентному форматі (наприклад, rgb(25%,30%,70%));

padding - визначає пропуск між вмістом блоку й границею. Для кожної, сторони є своя властивість: padding-top, padding-right, padding-п й padding-left;

weight height - задають відповідно ширину й висоту блоку без підрахунку відступів, границь і полів;

float - визначає розміщення поточного елемента по горизонталі стосовно зовнішнього елемента;

clear - скасовує дія властивості float.

1.6.5. Властивості списків

Таблиці стилів дозволяють управляти відображенням списків: задавши гарнітуру, розмір і кольори шрифту, вибирати вид маркерів (кружок, малюнок і т.д.). Нагадаємо, що маркер - це символ або зображення яким починається кожен рядок списку.

Ми розглянемо властивості, призначені для форматування маркерів списків: list-style-type, list-style-image, list-style-posit: скорочена форма запису цих трьох властивостей має вигляд list-style.

list-style-type - задає маркери для впорядкованих (нумерованих) і неупорядкованих (маркірованих) списків. Набір припустимих значень властивості list-style-type включає 22 значення, більшість з яких не підтримуються розповсюдженими браузерами. Тому зупинимося на значеннях, які часто використовуються:

- square - маркер у вигляді квадратика;
- circle - кружок;
- disc - затемнений кружок (значення за замовчуванням);
- decimal - десяткові числа, починаючи з 1;
- lower-roman - рядкові римські цифри, наприклад, i, ii, iii;
- upper-roman - прописні римські цифри, наприклад, I, II, III;
- lower-latin або lower-alpha - рядкові латинські букви;
- upper-latin або upper-alpha - прописні латинські букви, наприклад A, B;
- none - маркер не відображається;

list-style-image - задає маркер у вигляді картинки (зображення);

list-style-position - встановлює позицію маркера в рядку списку. Значення цієї властивості звичайно задає додатковий відступ перед кожним рядком списку.

Запитання для самоконтролю

1. Що таке гіперпосилання?
2. Що таке гіпертекст?
3. Що таке web-документ?
4. Для чого призначена програма-браузер?
5. Що таке web-сайт?
6. Яка структура простого web-документа?
7. Для чого призначена мова HTML?
8. Що таке тег і які є теги?
9. Які параметри може мати тег BODY?
10. Який тег позначає початок нового абзацу?
11. Які теги позначають товстий, курсивний і підкреслений шрифти?
12. Які теги призначені для вирівнювання елементів на сторінці?
13. Яке призначення тега FONT?
14. Яких значень можуть набувати параметри тега FONT?
15. Які є типи списків?
16. Як створити ненумерований список?
17. Як створити нумерований список?
18. Як створити список означень?
19. Яке призначення тега TABLE?
20. Які параметри може мати тег TABLE?
21. Які теги формують у таблиці рядки, клітинки-заголовки і звичайні клітинки?
22. Як у таблиці об'єднати декілька клітинок в одну?
23. Який параметр використовують для вирівнювання елементів?
24. Що таке фрейми?
25. Які файли потрібні для створення сайту з фреймами?

26. Яке призначення основного html-файлу?

27. Що відображають зазвичай у лівому фреймі сайту?

28. Які параметри може мати тег <FRAMESET>?

29. Які параметри може мати тег <FRAME>?

30. Як створюються складні конфігурації фреймів?

31. Що означає запис COLS = "30% ,*"?

32. Що означає запис COLS = "1*, 4*"?

33. Що означає запис COLS = "120, 240, *"?

34. Які фрейми створить параметр COLS = "25%, 50%, 25%"?

35. Яке призначення параметра TARGET?

36. Яке призначення тега-контейнера NOFRAME?

37. Яке призначення таблиці стилів?

38. Які є способи взаємодії таблиці стилів і html-файлу?

39. У чому полягає спосіб зв'язування?

40. Що таке каскадні таблиці стилів?

41. З чого складається таблиця стилів?

42. Назвіть властивості категорії border.

43. Назвіть властивості категорії font.

44. Назвіть властивості категорії text.

45. Назвіть властивості категорії margin.

46. Назвіть властивості категорії padding.

47. Які одиниці вимірювання застосовують у мові CSS?

48. Для чого групують властивості?

49. Що таке успадковування властивостей?

50. Яке призначення тега STYLE?

51. Яке призначення тега DIV?

РОЗДІЛ 2. МОВА PHP

2.1. Загальні поняття та опис PHP

Використання такої мови, як PHP і бази даних MySQL, дозволяє робити сайти динамічними: такими що налаштовуються та містять інформацію, яка змінюється в реальному часі.

У посібнику, основну увагу приділено реальним додаткам, які можуть бути використані маркетинговими та менеджерскими службами. Почнемо з розгляду простої інтерактивної системи замовлень, а потім ознайомимося з різними складовими частинами PHP й MySQL.

PHP - це серверна мова створення сценаріїв (або сторони сервера), розроблена спеціально для Web. В HTML-сторінку можна впровадити код PHP, який буде виконуватися при кожному її відвідуванні. Код PHP інтерпретується Web-сервером і генерує HTML або інший формат виведення, що спостерігається відвідувачем сторінки.

Розроблена PHP в 1994 році Расмусом Лердорфом (Rasmus Lerdorf). Ця мова була прийнята рядом талановитих людей і перетерпіла три основних редакції, поки не стала широко розповсюдженим продуктом. В квітні 2009 року нею користувалося майже в двадцять п'ять мільйонів доменів в усьому світі.

PHP - це продукт із відкритим вихідним кодом (Open Source). У користувача є доступ до вихідного коду. Його можна використати, змінювати й вільно поширювати іншим користувачам або організаціям.

Спочатку PHP було скороченням від *Personal Home Page (Персональна початкова сторінка)*, але потім ця назва була змінена відповідно до рекурсивної угоди по найменуванню GNU (GNU = Gnu's Not Unix) і тепер означає *PHP Hypertext Preprocessor (Препроцесор гіпертексту PHP)*. В даний час основною версією PHP є п'ята.

Програми PHP можуть виконуватися двома способами: як сценарний додаток Web-сервером й як консольні програми. Оскільки, нашим завданням є програмування web-додатків, ми переважно будемо розглядати перший спосіб.

Справа в тому, що PHP, як правило, використовується для програмування додатків, пов'язаних з Інтернетом. Однак, PHP можна ще використати як інтерпретатор командного рядка, в основному в *nix-системах. Останнє можливо за допомогою CORBA й COM інтерфейсів, а також за допомогою розширення PHP-GTK. При такому використанні PHP можливе рішення наступних завдань:

- створення додатків інтерактивного командного рядка;
- створення крос-платформних GUI додатків за допомогою бібліотеки PHP-GTK;
- автоматизація деяких завдань під Windows й Linux

MySQL (вимовляється *май-эс-кю-эл)* — дуже швидка, надійна *система управління реляційними базами даних (СУРБД).* База даних дозволяє ефективно зберігати, шукати, сортувати й одержувати дані. Сервер MySQL управляє доступом до даних, дозволяючи працювати з ними одночасно декільком користувачам, забезпечує швидкий доступ до даних і гарантує надання доступу тільки користувачам, що мають на це право. Отже, MySQL є багатокористувацьким, багатопотоковим сервером. Він застосовує *SQL (Structured Query Language — мову структурованих запитів),* стандартну мову структурованих запитів до бази даних. MySQL з'явився на ринку в 1996 р., але його розробка почалася ще в 1979 р. В теперішній час пакет MySQL доступний як програмне забезпечення з відкритим вихідним кодом.

Приступаючи до створення сайту системи електронної торгівлі, можна використати безліч різних продуктів. Потрібно вибрати апаратне за-

безпечення для Web-сервера, операційну систему, програмне забезпечення Web-сервера, систему керування базами даних і мову програмування або створення сценаріїв. Найдоцільніше вибрати Денвер.

«Денвер» - комплекс, призначений для зручного налагодження скриптів, не виходячи в Інтернет. Денвер містить у собі декілька найбільш популярних серверів сторонніх виробників, що працюють в Windows. Ви можете розробляти й тестувати сайти в Windows, а потім переносити їх на реальний хостінг, в Unix.

Чому проект називають «Денвер»? Вся справа в скороченнях. «Джентльменський набір Web-розроблювача», урізаний до перших букв, виглядає як «Д.н.w.р.». При швидкому прочитанні й виходить шукане місто.Денвер був придуманий і створений однією єдиною людиною Дмитрием Котеровим. Версія Денвера для Unix-подібних операційних систем поки не існує.

Ви завжди можете встановити **«Денвер»** та додаткові програми й документацію, відвідавши адресу http://www.denwer.ru. Кожен дистрибутив містить простий і зручний інсталятор, призначеним для настроювання компонентів під конкретну систему.

Мова програмування PHP спеціально призначена для роботи в Internet. Це скриптова мова з обробкою на сервері, яка вбудовується в HTML. Обробка PHP кода здійснюється на сервері, ще до того, як WEB сторінка буде передана браузеру.

Мови програмування бувають двох видів: інтерпретатори та компілятори. Транслятор – це це програма, що переводить код написаний на одній мові програмування, на іншу. Наприклад, програма p2c, яка переводить Pascal – код в C – код, є транслятором. Компілятор – це той самий транслятор. Він переводить код, написаний на мові високого рівня, в ма-

шинний код. В результаті роботи компілятора створюється, як правило двійковий виконуваний файл(WIN /DOS з розширенням .exe,.com).

Інтерпретатор - це інший тип мови програмування. Інтерпретатор нічого не переводить, а лише виконує код. Він аналізує код програми і виконує кожний оператор.

По продуктивності інтерпретатори значно вступають компіляторам, оскільки машинний код виконується значно швидше.

PHP-це не інтерпретатор і не компілятор. PHP - щось середнє між компілятором та інтерпретатором. Розглянемо, яким чином здійснюється обробка PHP сценарію. На вхід PHP подається сценарій. Він переводить, транслює його в спеціальний байт-код, а потім PHP виконує байт-код, при цьому він не створює виконуваний файл. Байт-код значно компактніше звичайного коду, тому його легше інтерпретувати(виконувати). Тому можна сказати, що PHP більше інтерпретатор, ніж компілятор. Продуктивність роботи PHP достатня для роботи в Internet.

Використання має свої переваги:

- Вам не потрібно піклуватись при об'єм виділеної пам'яті, закривати файл по закінченню роботи, за вас це робить інтерпретатор.

- Вам не потрібно думати над типами змінних та оголошувати їх.

- Відлагодження програми теж робота інтерпретатора.

Напишемо першу програму на PHP.

Приклад 2.1. Перша програма.

```
<html> <head><title>Приклад</title>
</head> <body>
   <?
    echo "Привіт";
    ?>
</body></html>
```

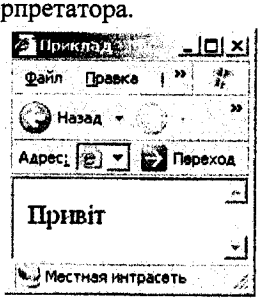


Рис.2.1.**Вікно прикладу2.1.**

Код PHP записується в спеціальні початковий та кінцевий теги(<?,?>), що дозволяє вам входити та виходити з "режиму PHP". Інструкції розділяються символом "крапка з комою".

Кінцевий тег (?>) теж кінець оператора, тому наступні варіанти еквівалентні:

```
<?php
  echo "Запишемо текст";
?>
<?php echo " Запишемо текст " ?>
```

2.1.1. Ідентифікатори

Ідентифікатори - це імена змінних. Імена ідентифікаторів підпорядковуються деяким простим правилам:

- Ідентифікатори можуть мати будь-яку довжину й складатися з букв, цифр, символів підкреслення й знаків долара. Однак при використанні в ідентифікаторах знаків долара варто бути уважними. Ідентифікатори не можуть починатися із цифри.
- У PHP ідентифікатори чутливі до регістра. $tireqty і $TireQty далеко не рівнозначні. Спроба використання рядкових символів замість прописних і навпаки - помилка програмування. Виключення із цього правила становлять вбудовані PHP-функції - їхні імена можуть вводитися в будь-якому регістрі.
- Ідентифікатори змінних можуть збігатися з іменами вбудованих функцій.
- Однак це звичайно викликає плутанину, тому подібних ситуацій варто уникати. Не можна також створювати функції, ідентифікатори яких збігаються з ідентифікаторами вбудованих функцій.

Крім змінних, переданим з HTML-форми, можна оголошувати свої власні змінні. Одна з особливостей PHP полягає в тому, що змінні не обо-

в'язково оголошувати перш, ніж їх можна буде використати. Змінна буде створюватися при першому присвоєнні їй значення.

2.1.2. Типи змінних PHP

Тип змінної пов'язаний з видом даних, що зберігаються в ній.

PHP підтримує наступні типи даних:

- Integer (цілий) - Використовується для цілих чисел.
- Boolean логічні змінні;
- Double (подвійної точності) - Використовується для дійсних чисел.
- String (строковий) - Використовується для рядків символів
- Array (масив) - Використовується для зберігання декількох елементів даних одного типу
- Object (об'єкт) - Використовується для зберігання екземплярів класів

PHP підтримує також типи **pdfdoc** і **pdfinfo,** якщо вони були встановлені з підтримкою PDF (Portable Document Format — формат переносних документів).

Мова PHP досить ліберальна стосовно типів. У більшості мов програмуванні змінні можуть містити тільки один тип даних, і цей тип повинен бути оголошений перш, ніж змінну можна буде використати, як це має місце в С. У PHP тип змінної визначається привласненим їй значенням. Наприклад, при створенні змінних $zukor і $tovar їхні початкові типи були визначені в такий спосіб:

```
$zukor =0;
$tovar =  0.00;
```

Оскільки змінній $zukor було привласнено цілочислове значення 0, ця змінна має тип integer. Аналогічно, змінна $tovar має тип double. Як не дивно, у сценарій цілком можна було б додати наступний рядок:

```
$tovar =  "He1lo" ;
```

У цьому випадку змінна $tovar одержала б тип string. PHP у будь-який момент часу змінює тип змінної відповідно до даних, що зберігаються в ній. Подібна можливість зміни типів може виявитися винятково корисною. Запам'ятайте, що PHP "автоматично" розпізнає тип даних, що знаходиться у змінні. Існує функція gettype(), яка повертає тип, який PHP призначив змінній.

Приклад 2.2. Визначення типу змінної.

```
<?
  $var = "5";
  $var1 = true;
  echo(gettype($var));
  echo "<br>";
  echo(gettype($var1));
?>
```

В першому випадку PHP повертає string, в другому Boolean.

2.1.3. Приведення типів

Використовуючи приведення типів, можна імітувати, начебто змінна або значення має інший тип. Для цього досить перед змінною, тип якої потрібно перетворити, помістити в круглих дужках тимчасовий тип. Наприклад, дві створені вище змінні можна було б оголосити з використанням приведення типів.

```
$zukor  = 0;
$tovar  = (double) $zukor  ;
```

Другий рядок означає "Взяти значення, що зберігається в змінній $zukor, інтерпретувати як значення типу double і зберегти в змінній $tovar.

Приклад 2.3. Використання функції **settype**(), яка явно встановлює тип .

```
<?
  $var = "5";
  echo(gettype($var));
  settype($var,integer);
  echo "<br>";
  echo(gettype($var));
?>// Результат string та integer.
```

Поряд з константами, обумовленими користувачем, PHP має велику кількість власних констант. Ці константи можна легко переглянути, виконавши команду phpinfo(). В результаті виводиться список визначених змінних і констант PHP, а також інша корисна інформація.

2.1.4. Константи

Між константами та змінними існують певні відмінності :

1. Перед іменем константи немає знаку dollar ($);
2. Константи можуть бути визначені тільки через використання функції define(), але не звичайним присвоєнням;
3. Константи, можуть бути визначені і доступ до них може бути отриманий, в будь-якому місці, не залежно від правил області видимості змінних.

Приклад 2.3. Використання констант.

```
<?php
define("CONSTANT", "Hello world.");
echo CONSTANT; // виводить "Hello world."
?>
```

2.1.5. Область дії змінних

Термін *область дії (scope)* відноситься до розділів сценарію, усередині яких доступна конкретна змінна. У PHP використаються наступні три основних типи областей дії:

• Глобальні змінні, оголошені в сценарії, доступі у всьому сценарії, але *не усередині функцій.*

• Змінні, що використовуються усередині функції, є локальними для даної функції.

• Змінні, що використовуються усередині функції, які оголошені як глобальні, відносяться до глобальним змінних з такими ж іменами. Всі змінні, що використовуються за замовчуванням будуть глобальними.

2.1.6. Знаки операцій. Арифметичні операції. Математичні функції

Знаки операцій – це символи, які можна використовувати для маніпуляції значеннями й змінними шляхом виконання над ними операцій. Деякі із цих операцій будуть потрібні для обчислення загальної суми й податку для замовлення клієнта.

Ми вже згадували дві операції: операцію присвоювання (=) і операцію конкатенації рядків (.) Тепер давайте ознайомимося з повним списком операцій.

У загальному випадку операції можуть виконуватися над одним, двома й трьома аргументами, причому більшість із них виконується над двома аргументами. Наприклад, операція присвоювання має два аргументи - адреса, що вказує ліворуч від символу, і вираз, що вказується праворуч. Ці аргументи називаються *операндами,* тобто елементами, з якими повинна виконуватися операція.

Арифметичні операції дуже прості - це звичайні математичні операції. Вони наведені в таблиці 1.

Таблиця 2.1.**Арифметичні операції PHP**

Операція	Знак
Додавання	+
Віднімання	-
Множення	*
Ділення	/
Обчислення остачі по модулю(5 % 2=1)	%

Таблиця 2.2.**Математичні функції PHP**

Назва функції	Математичний запис
abs(x)	|x|
cos(x)	Cos(x)
sin(x)	Sin(x)
tan(x)	Tan(x)
log(x)	Log(x)
pow(x,y)	x^y
sqrt(x)	$\sqrt{x}$
exp(x)	e^x
atan(x)	Arctg(x)
acos(x)	Arccos(x)
asin(x)	Arcsin(x)

Для кожної із цих операцій можна зберігати результат виконання операції наприклад:

```
$c=$a + $b.
```

Додавання й віднімання працюють так, як і очікується. Результатом їхнього обчислення є, відповідно, сума й різниця значень, що зберігаються в змінних $a і $b.

Символ віднімання - можна використовувати й у якості унарної операції (операції, що приймає один аргумент, або операнд) для позначення негативних чисел. Наприклад: $a = -1.

Множення й ділення також працюють як звичайно. Операція взяття модуля повертає остачу від ділення змінної $a на змінну $b. Розглянемо наступний фрагмент коду:

$a = 27; $b = 10;

$result = $a%$b;(27%10=7).

Варто звернути увагу, що арифметичні операції звичайно застосовуються до цілих чисел або значень із подвійною точністю.

Приклад 2.4. Приклад використання математичних функцій.

$$y = \sqrt[5]{x^2 + 7.2} - |x - 5| + sin\frac{\pi x}{3}$$

x=2 y=-0.5127

```
<?
$x=2;
$y=pow($x*$x+7.2,1./5)-abs($x-5)+sin(3.14*$x/3);
echo $y;
?>
```

2.1.7. Рядкові операції

Операцію конкатенації рядків можна використовувати для додавання двох рядків із зберігання результату, наприклад:

```
$a = "Іван ";
$b = " Петренко";
$result= $a.$b;
```

Тепер змінна **$result** містить рядок "Іван Петренко".

Ми вже знайомі з операцією =, основною операцією присвоювання. Це завжди означає операцію присвоювання й читається як "встановлюється рівним".

Наприклад: $цукор = 0.

Цей запис повинен читатися як "значення змінної $цукор встановлюється рівним нулю." Імена змінних доцільно писати на латині.

У результаті використання операції присвоювання повертається підсумкове значення. Це дозволяє виконувати дії на зразок наступних:

$b=6+($a=7);

У результаті значення змінної $b встановлюється рівним 13. Це справедливо для всіх операторів присвоювання: значенням всього оператора присвоювання є значення, привласнене лівому операнду.

При оцінці значення виразу дужки застосовуються з метою підвищення пріоритету виразу, як це робилося в наведеному прикладі. Дужки працюють точно так, як і в математиці.

Крім простих операцій присвоювання існує набір комбінованих операцій присвоювання. Кожна з них являє собою скорочену форму запису якої-небудь іншої операції зі змінною й присвоєння результату цієї змінній

$a+=1;

що еквівалентно запису

$a=$a+1.

Об'єднані операції присвоювання існують для кожної з арифметичних операцій і для операції конкатенації рядків. Перелік всіх об'єднаних операцій присвоювання й результат їхньої дії наведений у таблиці 3.

Таблиця 2.3. **Операцій присвоювання**

Символ операції	Використання	Еквівалентна операція
+=	a += $b	$a=$a+$b
-=	$a-=$b	$a=$a-$b
*=	$a*=$b	$a=$a*$b
./=	$a/=$b	$a=$a/$b
%=	$a%=$b	$a=$a%$b
.=	$a.=$b	$a=$a.$b

2.1.8. Префіксний і суфіксний інкремент і декремент

Операції префіксного й суфіксного інкремента (++) і декремента (--) аналогічні операціям += і -=, але з декількома відмінностями.

Всі операції інкременту роблять подвійну дію - вони збільшують і привласнюють значення. Давайте розглянемо наступний код:

```
$a=10;
echo  ++10a.
```

У другому рядку використовується операція префіксного інкремента, який називається так тому, що символ ++ записується перед $a. У результаті спочатку значення $a збільшується на 1, а потім повертається збільшене значення. У цьому випадку значення $a збільшується до 11, а потім 11 повертається й виводиться. Значенням усього виразу буде 11. Розглянемо наступні рядки

```
$a=4 ; echo  $a++.
```

У цьому випадку дії виконуються у зворотному порядку. Тобто, спочатку значення $a повертається й виводиться, а потім збільшується на 1. Значенням усього виразу є 4. Саме це значення й буде виведено. Однак після виконання цього оператора значення змінної $a дорівнює 5.

Нескладно догадатися, що операція - діє аналогічно, тільки тут значення $a зменшується на 1, а не збільшується.

Операція посилання & (амперсанд), що може й використовуватися в сполученні з операцією присвоювання. Звичайно, коли змінна присвоюється інший змінній, створюється копія першої змінної, котра зберігається де-небудь у пам'яті. Наприклад,

$a = 5;

$b = $a.

Ці рядки коду створюють другу копію значення $a й зберігають її в змінній $b. Якщо згодом значення $a змінити, значення $b не зміниться:

$a = 7; // значення $b буде, як і раніше, дорівнює 5.

Створення копії можна уникнути, використовуючи операцію посилання &, наприклад

$a = 5;

$b = &$a;

$a = 7; // тепер значення $a й $b рівні 7.

Операції порівняння використовуються для порівняння двох значень. Вирази, в яких використовуються ці операції, повертають залежно від результату порівняння логічні значення **true** (істинно) або **false (хибно).**

Операція рівності == (два знаки рівності) дозволяє перевірити рівність двох значень. Наприклад, вирази

$a==$b.

Результатом цього виразу буде **true,** якщо вони рівні, або **false,** якщо вони не рівні.

2.1.9. Логічні операції

Логічні оператори , на відміну від порозрядних, працюють із логічними змінними (**boolean**) і інтенсивно використовуються в керуючих конс-

трукціях: цикли й умови. Логічні змінні, або більш правильно, змінні типу **Boolean** мають лише два значення: **true(істина)** і **false (неправда)**. У виразах **true** й **false** можна замінити на 1(будь-яке відмінне від 0 число) і 0, відповідно.

Таблиця 2.4.**Логічні операції**

Символ операції	Назва	Використання	Результат
!	НЕ	!$b	Повертається true, якщо значення $a false, і навпаки.
&&	И	$a&&$b	Повертається true, якщо обидві змінні $a і $b мають значення true; у іншому випадку повертається false.
||	АБО	$a\|\|$b	Повертається true, якщо кожна зі змінних $a або $b або обидві мають значення true; у іншому випадку $a\|\|$b повертається false.
and	І	$a and $b	Та ж операція, що й &&, але з меншим пріоритетом.
or	АБО	$a or $b	Та ж операція, що й ||, але з меншим, пріоритетом.

Логічні оператори для змінних типу **Boolean** виконують роль операторів додавання, віднімання для звичайних змінних.

Приклад 2.5. Використання логічних функцій.

```
<?php
$flag1 = true;
$flag2 = false;
if($flag1 || $flag2)
// Хоч одна з двух змінних ($flag1 $flag2) має значення істина
{
echo "<p>Умова: true (Одна із змінних істина)</p>";
// Так
} else {
echo "<p>Умова: false (Обидві змінні хибні)</p>";
// Ні обадві мають значення false
}?>
```

PHP підтримує також ряд інших операцій порівняння, які перераховані в таблиці 5.

Таблиця 2.5.**Операцій порівняння**

Символ операції	Назва	Використання
==	дорівнює	$a==$b
===	ідентично	$a===$b
!=	Не дорівнює	$a!=$b
<>	Не дорівнює	$a<>$b
<	менше	$a<$b
>	більше	$a>$b
<=	Менше або дорівнює	$a<=$b
>=	Більше або дорівнює	$a>=$b

2.1.10. Порозрядні операції

Порозрядні операції дозволяють обробляти цілі числа як послідовності з їхніх розрядів. Імовірно, у PHP ці операції прийдеться використовувати не дуже часто, проте, їхній перелік наведений у таблиці 2.6.

Таблиця 2.6. **Порозрядні операції**

Символ операції	Назва	Використання	Результат
&	Порозрядне І	$a&$b	Розряди, установлені в одиничні стани $a й $b, установлюються в одиничні стани в результаті.
|	Порозрядне АБО	$a\|$b	Розряди, установлені в одиничні стани в $a або $b, установлюються в одиничні стани в результаті.
~	Порозрядне НЕ	~$a	Розряди, установлені в одиничні стани $a, установлюються в нульові стани в результаті, і навпаки.
^	Порозрядне що виключає АБО	$a^$b	Розряди, установлені в одиничні стани в $a або $b, але не в обох змінних, установлюються в одиничні стани в результаті.
<<	Зміщення вліво	$a<<$b	Розряди в змінної $a зрушуються вліво на $b позицій.
>>	Зміщення вправо	$a>>$b	Розряди в змінної $a зрушуються вправо на $b позицій.

Приклад 2.6. Використання порозрядних функцій.

```
<?
  echo(4<<2); // дорівнює 16
  echo"<br>";
  echo(5>>1); //  дорівнює 2
  echo"<br>";
  echo(6&5); //  дорівнює 4
  echo"<br>";
  echo(6|5); // дорівнює 7
  echo"<br>";
  echo(6^5); // дорівнює 3
?>
```

Приклад 2.7. Використання логічних функцій.

```
<?php
  $flag1 = true;
  $flag2 = true;
  $flag3 = false;
  if($flag1 && ($flag2 || $flag3))
  {
    echo "<p>Умова істина<p>";
  }
  else
  {
    echo "<p></p>Умова хибна<p>";
  }
?>
```

Розберемо приклад 2.8. Двійковий код для 4 дорівнює 100, при зсуві вліво на 2 розряди код 100 стає рівним 10000, що відповідає десятковому

значенню 16. Інші приклади можуть бути розібрані аналогічно. Зверніть увагу на те, що зсув вліво на n позицій еквівалентний множенню на 2n, а зсув вправо зменшує відповідне значення в 2n раз із відкиданням дробової частини результату (тому 5>>1 дорівнює 2). Основне призначення цих операторів - швидкі обчислення.

Крім описаних, існує також ряд інших операцій. Символ коми (,) використовується для поділу аргументів функцій і елементів інших списків. Звичайно він застосовується в міру необхідності.

Операція подавлення помилки. Наприклад: $a=@(57/0);

Без символу операції @ цей рядок буде генерувати попередження про ділення на нуль (можете це перевірити). При використанні операції @ вивід повідомлення про помилку подавлюється.

При такому подавленні повідомлень про помилки необхідно створити деякий код обробки помилок з метою перевірки на предмет генерації попереджень. Якщо PHP встановлений з активізованою функцією track_errors, повідомлення про помилку зберігається в глобальній змінній $php_errormsg.

2.2. Найпростіші приклади вбудовування PHP в HTML

HTTP-протокол, що лежить в основі Web, не зберігає інформації про стан сеансу. Це означає, що будь-яке звернення клієнта сервер сприймає як звернення нового клієнта, і навіть якщо для завантаження картинок з поточної сторінки клієнт формує запит, сервером він сприймається як запит нового клієнта, ніяк не пов'язаного з тим, що тільки що завантажив сторінку. Дана схема працювала досить добре для статичних сторінок, але стала зовсім неприйнятної для динамічних. У зв'язку із цим в Web були уведені механізми сесій й cookies, які в даний момент підтримують всі учасники Web: клієнти, proxy-сервери та кінцеві сервери.

Неможливість збереження стану в рамках HTTP-протоколу означає, що змінні й масиви, визначені в одному скрипті, не запам'ятовуються при

переході до іншого скрипта або при перезавантаженні поточної сторінки. Існує кілька способів передачі даних з одного скрипта в інший.

Дані можна передати методом POST, наприклад, через HTML-форму. У цьому випадку дані, уведені в елементи керування HTML-форми, будуть розміщені в елементах суперглобального масиву $_POST, ключі яких будуть збігатися з назвами елементів керування HTML- форми.

Другий спосіб передачі даних між скриптами полягає в збереженні проміжних даних у файлах або базі даних.

Третій спосіб це збереження значень змінних в cookies. Cookies - це текстові файли на комп'ютері клієнта, де зберігають пари "ім'я/значення".

Приклад 2.8. Приклад обчислення в формі.

```
<html>
<head>
<title>Магазин</title>
</head>
<body>
<h1>Магазин </h1>
<h2>Перша форма</h2>
</body>
</html>
```

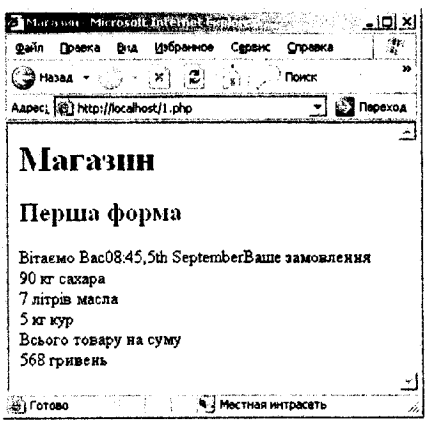


Рис.2.2 **Вікно прикладу 2.8**

Під заголовком введемо наступні рядки

```
<h2>Перша форма</h2>
<?
echo"<p>Вітаємо Вас";
echo date("H:i,jS F");
?>
```

Ускладнюємо форму.

```
<html>
<head>
<title>Магазин</title></head>
<body>
<h1>Магазин </h1>
<h2>Перша форма</h2>
<?
$cukor=90;$maclo=7;$kuri=5;
echo"<p>Вітаємо Вас";
echo date("H:i,jS F");
echo"Ваше замовлення";
echo"<br>";
echo $cukor." кг цукру<br>";
echo $maclo." літрів масла<br>";
echo $kuri." кг кур<br>";
?>
</body></html>
```

Порахуємо на яку суму зроблено замовлення.

```
<html><head><title>Магазин</title></head>
<body>
<h1>Магазин </h1>
<h2>Перша форма</h2>
<?
define("CinaCukru",5);
define("CinaMaclo",4);
define("CinaKuri",18);
$cukor =90;$maclo =7;$kuri =5;
echo"<p>Вітаємо Вас";
```

```
echo date("H:i,jS F");
echo"Ваше замовлення";
echo"<br>";
echo $cukor." кг цукру<br>";
echo $maclo." літрів масла<br>";
echo $kuri." кг кур<br>";
echo "Всього товару на суму<br>";
$Tovar=$cukor*CinaCukru+$maclo*CinaMaclo+$kuri*CinaKuri;
echo $Tovar."  гривень";
?>
</body></html>
```

2.2. Використання операцій

Приклад 2.9. Передача даних методом POST.

```
<html><head><title>Магазин</title></head>
<body>
<form action=" ekspert.php " method="POST">
<table border=0>
<tr bgcolor=#66ff33>
 <td width=150>Товари</td>
 <td width=15>Вартість</td>
</tr><tr>
 <td>Цукор</td>
 <td align=center>
<input type="text" name="cukor" size=3 maxlength=3></td>
</tr><tr><td>Масло</td>
 <td align=center>
<input type="text" name="maclo" size=3 maxlength=3></td>
</tr><tr>
```

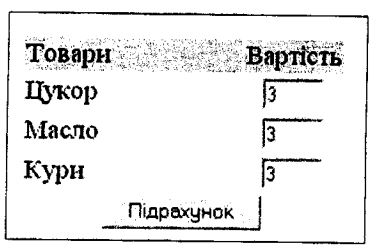


Рис.2.3. **Вікно приладу 2.9**

```
<td>Кури</td>
<td align=center><input type="text" name="kuri" size=3 max-
length=3></td>
</tr><tr>
<td colspan=2 align=center><input type=submit value= "Підрахунок" >
</td></tr></table></form> </body></html>
```

Обчислити підсумкову суму для форми замовлення Магазина. Для цього в файл " ekspert.php " необхідно ввести наступний код.

```
<html><head><title>ЧЕК</title></head>
<body>
<html><head><title>Магазин</title></head>
<body>
<h2>ЧЕК</h2>
<?
$cukor =$_POST['cukor'];
$maclo =$_POST["maclo"];
$kuri =$_POST["kuri"];
$suma = 0;
$suma += $cukor;
$suma += $maclo;
$suma += $kuri;
$Tovar = 0.00;
define("CinaCukor",8);
define("CinaMaclo",5);
define("CinaKuri",20);
$date = date("H:i, jS F");
echo "<p>Ваше замовлення";
echo $date;
```

ЧЕК

Ваше замовлення18:53, 19th December

Ваш вибір:
3 цукор
3 масло
3 кури
Всього товару на суму
99 гривні

Всього99.00

Рис.2.4. Результуюче вікно прикладу 2.11

```
echo "<br>";
echo "<p>Ваш вибір:";
echo "<br>";
if( $suma == 0 )
{
echo "Ви нічого не замовили!<br>";
}
else
{
if ( $cukor>0 )
echo $cukor." цукор<br>";
if ( $maclo>0 )
echo $maclo." масло<br>";
if ( $kuri>0 )
echo $kuri." кури<br>";
}
echo "Всього товару на суму<br>";
$Tovar=$cukor*CinaCukor+$maclo*CinaMaclo+$kuri*CinaKuri;
echo $Tovar."  гривні";
$Tovar=number_format($Tovar, 2, ".", " ");
echo "<P>Всього".$Tovar."</p>";
?>
</body></html>
```

Як бачите, у цьому фрагменті коду задіяно кілька операцій. Операції додавання (+) і множення (*) використовуються для обчислення числових значень, а операція конкатенації рядків (.) - для форматування виведення у вікні браузера.

При уважному розгляді зроблених обчислень може виникнути питання, чому вони були організовані саме в такому порядку. Підсумок здається правильним, але чому множення виконалося перед додаванням? Це обумовлено *пріоритетом* операцій, тобто порядком їхнього виконання.

2.3. Пріоритети і асоціативність: обчислення виразів

У загальному випадку операції мають пріоритети, або порядок їхніх обчислень. Крім того, операціям властива асоціативність, що визначає порядок виконання операцій з однаковим пріоритетом. У загальному випадку операції можуть виконувати зліва направо, справа наліво.

Таблиця 2.7. **Пріоритет виконання операцій**

Асоціативність	Операції
ліва	,
ліва	or
ліва	xor
ліва	and
права	print
ліва	= += -= *= /= .= %= &= |= ^= ~= <<= >>=
ліва	? :
ліва	||
ліва	&&
ліва	|
ліва	^
ліва	&
не асоціативна	== != === !==

не асоціативна	< <= > >=
ліва	<< >>
ліва	+ - .
ліва	* / %
права	! ~ ++ -- (int) (float) (string) (array) (object) @
права	[
не асоціативна	new

У цій таблиці операції, що мають найнижчий пріоритет, приводяться у верхній частині, а пріоритети зростають зверху вниз.

Зверніть увагу, що найвищим пріоритетом володіє операція, яку ми ще не розглядали: старі добрі круглі дужки. Вони підвищують пріоритет будь-якого, виразу, що знаходиться в них. Саме з їхньою допомогою, якщо буде потреба, можна змінювати правила пріоритету.

Розглянемо наступний фрагмент:

$Товар =$Товар*(1+$Податок).

Якби ми записали

$Товар = $Товар*1+$Податок .

Операція множення, що має більш високий пріоритет у порівнянні з операцією додавання, виконувалася б першою, що привело б до невірного результату. Використовуючи круглі дужки, можна домогтися, щоб спочатку обчислювався підвираз 1 + $Податок.

У виразі можна використати будь-яку необхідну кількість наборів дужок. При цьому першим буде обчислюватися вираз, вкладений в перші дужки.

2.4. Функції для роботи зі змінними

PHP надає також ряд функцій перевірки типів. Кожна із цих функцій приймає змінну як аргумент і повертає значення **true** або **false.** От їхній перелік:

- is_array();
- is_double(). is_float(), is_real() (Це та сама функція);
- is_long, is_int(), is_integer() (Це та сама функція);
- is_string();
- is_object().

2.5. Перевірка коректності даних

Перевірці коректності даних, що вводять користувачем необхідно приділяти досить велику увагу, оскільки неопрацьовані помилки, що виникають при неправильному уведенні даних, приводять до помилок у роботі скрипта, найчастіше катастрофічним. Припустимо, ви створюєте форму для відправлення користувачем листа, при цьому адресу електронної пошти необхідно вводити користувачеві. У цьому випадку, для коректної роботи програми ви повинні зробити дві речі:

- Перевірити, що поле, у яке заноситься електронна адреса непусте (оскільки користувач може просто забути ввести адресу, і, якщо цей випадок необроблений, виникне помилкова ситуація);
- Перевірити відповідність введеної адреси за допомогою регулярного виразу.

Крім звичайних помилок користувача, необхідно також виключити ситуації, у яких можливо зловмисне введення некоректних даних.

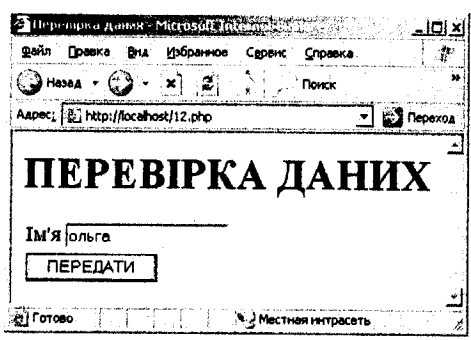


Рис 2.5. **Вікно прикладу 2.10**

Для цього текст, що вводить користувачем, необхідно обробити функціями видалення HTML-тегів (для виключення можливості написання скриптів на JavaScript й Visual Basic) і зворотних слешів (для виключення можливості написання скриптів на Perl). Набір дій, необхідний для перевірки коректності даних, що вводять користувачем, включає наступні етапи:

- перевірка того, що користувач увів дані;
- перевірка допустимості даних, які введені користувачем (як правило, здійснюється за допомогою регулярних виразів);
- обробка тексту, введеного користувачем функцією htmlspecialchars для видалення HTML-тегів;
- обробка тексту, уведеного користувачем функцією stripslashes для видалення зворотних слешів.

Перевірка того, що користувач увів дані, може здійснюватися, приміром, за допомогою функції **empty**:

Приклад 2.10. Перевірка даних.

```
<html>
<head>
 <title>Перевірка даних</title>
</head>
<body>
 <h1>ПЕРЕВІРКА ДАНИХ</h1>
 <form action="p.php" method="post">
 <table border=0>
 <tr><td>Ім'я</td><td><input type=text name="name" ><br></td> </tr>
 <tr><td colspan=2><input type=submit value="ПЕРЕДАТИ"> </td>
</tr></table> </form></body></html>
```

Файл p.php

```
<?
```

```
$name = $_POST["name"];
if (empty($name))
  echo "Ви не ввели ім'я";
   else
  echo $name;
?>
```

2.5.1. Перевірка допустимості даних

Нехай нам треба перевірити дані форми для відправлення повідомлення гостьової книги. Як правило, така перевірка здійснюється за допомогою регулярних виразів. Розглянемо приклад, у якому створюється регулярний вираз для перевірки адреси електронної пошти.

Будемо виходити з того, що адреса повинна мати вигляд something@server.com. Як бачимо, в адресі дві складові - ім'я користувача й ім'я домену, які розділені знайомий @. В імені користувача можуть бути

присутнім букви нижнього й верхнього регістрів, цифри, знаки підкреслення й мінуса, крапки. Для перевірки роздільника між ім'ям користувача й ім'ям домену у виразах потрібно додати +@. Таким чином, регулярний вираз(див.розділ 6), що перевіряє ім'я користувача й наявність роздільника має такий вигляд:

```
"/[0-9a-z_]+@[0-9a-z_^\.]"
```

Для перевірки доменного імені додаємо такий вираз:

```
"\.[a-z]{2,3}/i"
```

Поєднуючи ці кроки, одержуємо наступний регулярний вираз для перевірки адрес електронної пошти:

```
"/[0-9a-z_]+@[0-9a-z_^\.]+\.[a-z]{2,3}/i"
```

Точно в такий же спосіб ви можете перевірити й інші поля, що заповнює користувач.

2.5.2. Видалення HTML - тегів і зворотних слешів

Текст, що вводить користувач необхідно обробити функціями видалення HTML-тегів (для виключення можливості написання скриптів на JavaScript й Visual Basic) і зворотних слешів (для виключення можливості написання скриптів на Perl). Приміром, якщо змінна **$name** містить текст із ім'ям користувача, то обробка цього тексту виглядає так:

```
<?
 $name = substr($HTTP_POST_VARS["name"],0,32);
 $name = htmlspecialchars(stripslashes($name));
?>
```

Ще одна функція, призначена для перевірки стану змінних має наступний прототип: int isset(mixed var). Ця функція приймає як аргумент ім'я змінної й повертає значення **true,** якщо змінна існує, і **false** у противному випадку. Змінну можна видалити, використовуючи супутню функцію **unset().** Вона має наступний прототип: int unset(mixed var).

Розглянемо приклад використання цих трьох функцій.

```
<?
 $name = $_POST["name"];
 if (empty($name))  echo "Ви не ввели ім'я";
       if (!isset($name)) echo "Змінна не існує";
  unset($name);
?>
```

2.5.3. Повторна інтерпретація змінних

Привести змінну до відповідного типу можна за допомогою наступних функцій.

int intval(mixed var) ;

double doubleval(mixed var);

string strval(mixed var).

Кожна з них приймає змінну, як аргумент і повертає значення відповідного типу.

Запитання для самоконтролю

1. Що таке мова PHP?
2. Типи даних в мові PHP?
3. Як передати змінну з одного скрипта в інший?
4. Які Ви знаєте операції порівняння?
5. Що таке приведення типів?
6. Які Ви знаєте порозрядні операції?
7. Які пріоритети виконання операцій?

Задачі для розв'язування

1. Обчислити висоти трикутника, знаючи координати його вершин.
2. Визначити висоту трикутника, якщо площа трикутника дорівнює S, а основа більше висоти на величину A.
3. Знайти значення функції при x=1.

$$y = 9{,}2 * \cos x^3 - \left|\sin \frac{x}{1{,}2 - x}\right|.$$

4. Знайти значення функції при x=2.

1. $y = \cos(x^2 + 1{,}43\pi) + \frac{x}{2}.$

5. Знайти значення функції при довільному x.

2. $y = \frac{\cos|2x|}{2\pi - x} - \sin^2 \frac{x - 3{,}1}{2\pi}.$

6. Обчислити вартість розмови, якщо відома кількість хвилин, тариф. Передати змінні з одного скрипта в інший.
7. Обчислити периметр трикутника по заданих координатах його вершин. Передати змінні з одного скрипта в інший.

РОЗДІЛ 3. КЕРУЮЧІ СТРУКТУРИ

3.1. Умовний оператор

Керуючі структури - це мовні структури, які дозволяють управляти потоком виконання програми або сценарію. Їх можна розділити на умовні структури (або структури розгалуження) і структури повторення, або цикли. Далі розглядаються характерні реалізації кожного виду структур у PHP.

Щоб забезпечити своєчасну реакцію на інформацію, що вводиться користувачем, код повинен бути спроможним приймати рішення. Конструкції, які вказують програмі про необхідність прийняття рішень, називаються *умовними операторами.*

Для прийняття рішень застосовується **оператор if**, який використовує умову. Якщо умова має значення **true,** буде виконуватися наступний за ним блок коду. Умова в операторі if повинна записуватися в круглі дужки().

Наприклад, якщо форма замовлення Магазина не містить ні цукру, ні масла, ні курей, імовірно, це зв'язано з випадковим натисканням кнопки Submit. Замість спілкування ("Замовлення оброблене"), на сторінці слід б відобразити більш корисне повідомлення.

Якщо відвідувач взагалі не замовляє продукти, імовірно, має сенс вивести повідомлення ("Ви нічого не замовили на попередній сторінці!"). Це легко досягається за допомогою наступного оператора:

```
if ($Tovar == 0);
echo " Ви нічого не замовили на попередній сторінці! <br>";
```

У цьому операторі використовується умова $Tovar == 0. Пам'ятайте, що операція рівності (==) діє інакше, ніж операція присвоювання (=).

Умова $Tovar == 0 буде мати значення **true,** якщо значення змінної $Tovar дорівнює нулю. Якщо значення змінної $Tovar не дорівнює ну-

лю, значення буде дорівнює false. Коли значенням умови буде true, оператор echo виконається.

```
<?
$Tovar  =2;
if($Tovar  ==2) echo "товар";
?>
```

Повний умовний оператор

```
<?
$Tovar  =0;
if($Tovar  ==2) echo "tovar";
else
echo "no tovar";
?>
```

Приклад 3.1. Найти найбільше з трьох чисел.

```
<?
$a=1;$b=5;$c=4;
echo "<p>найбільше</p>";
if (($a>$b)&&($a>$c)) echo $a;
if (($b>$a)&&($b>$c)) echo $b;
if (($c>$a)&&($c>$b)) echo $a;
?>
```

Приклад 3.2. Розв'язок квадратного рівняння.

```
<?
$a=1;$b=2;$c=1;
echo "<p> Розв'язок квадратного рівняння</p>";
$d=sqrt($b*$b-4*$a*$c);
if ($d<0) echo "net pozvjazky";
else
```

```
{$x1=(-$b+sqrt($b*$b-4*$a*$c))/2;
echo $x1;
$x1=(-$b+sqrt($b*$b-4*$a*$c))/2;
echo $x1;}?>
```

3.2. Блоки коду

Часто усередині умовного оператора if, потрібно виконати більше одного оператора. Перед кожним з них зовсім не обов'язково поміщати новий оператор if. Замість цього послідовність операторів можна згрупувати в блок. Для оголошення блоку оператори повинні, бути записані у фігурні дужки:

```
If ($Tovar= 0 )
{
echo "<font color=red>" ;
echo   " Ви нічого не замовили на попередній сторінці! <br>";
echo "</font>";
}
```

Тепер три рядки коду, вкладені у фігурні дужки, є блоком. Коли значенням умови буде true, виконуються всі три рядки. Якщо значенням умови буде false, всі три рядки будуть ігноруватися.

Приклад 3.3. Приклад використання блоків коду.

```
<?
$a=2;$b=3;$c=5;
$tovar=$a+$b+$c;
if($tovar==0)
echo " no tovar";
else
{
if($a==2) echo $a."kg";
```

```
if($b==1) echo $b."kg";
if($c==5) echo $c."kg";
}
?>
```

Як ми вже відзначали, вирівнювання коду в PHP значення немає. Відступи потрібні тільки для підвищення читабельності. В основному, вони використовуються для того, щоб з першого погляду було видно, які рядки будуть виконуватися при яких умовах, які оператори згруповані в блоки і які з операторів є частиною циклів або функцій.

3.3. Оператори else

Часто потрібно приймати рішення не тільки про виконання тієї або іншої дії, але й вибирати певний набір можливих дій у противному випадку. Оператор else дозволяє визначити альтернативну дію, що повинна виконуватися, якщо значенням умови в операторі if виявиться false. У розглянутому прикладі необхідно попереджати клієнтів, коли вони нічого не замовляють. З іншого боку, якщо вони роблять замовлення, замість попередження потрібно вивести список замовленого.

Якщо змінити код і додати в нього оператор else, можна відображати або попередження, або підсумкову інформацію.

```
If ($Tovar == 0)
{
echo   " Ви нічого не замовили на попередній сторінці! <br>";
}
else
{
echo  $cukor. " кг<br>" ;
echo $maclo." кг<br>";
echo $kuri." кг<br>" ;
```

```
}
```

Вкладаючи операторі if один в іншій, можна будувати більше складні логічні процеси. Наступний код не тільки здійснить відображення підсумкової інформації, коли значення умови $Tovar ==0 дорівнює true, але забезпечить відображення кожного з підсумкових рядків тільки при виконанні їх власної умови.

```
if($Tovar ==0){
echo   " Ви нічого не замовили на попередній сторінці!<br>";
}
else
{
if ( $cukor>0)
echo $cukor." кг<br>";
 if ( $maclo>0)
echo $maclo." кг<br>";
 if ( $kuri>0)
echo $kuri." кг<br>"; }
```

3.4. Оператори elseif

Магазин надає знижки при замовленні великої кількості цукру. Схема знижок виглядає в такий спосіб:

- Придбання менше 10 т. цукру - без знижки
- Придбання 10-49 т. цукру - знижка 5%
- Придбання 50-99 т. цукру - знижка 10%
- Придбання 100 і більше т. цукру - 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 *(&&)*

Приклад 3.4. Обчислюємо знижку.

```
<?
define("cina",5);
$zukor=50;
if($zukor<10) $znizhka=0;
 elseif($zukor>=10 && $zukor<=49)$znizhka=5;
 elseif($zukor>=50 && $zukor<=99)$znizhka=10;
 elseif($zukor>=100)  $znizhka=15; ;
echo $zukor*cina-$znizhka;
?>
```

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

При використанні каскадних наборів операторів elseif варто пам'ятати, що буде виконуватися тільки один із блоків операторів. У даному прикладі оскільки всі умови є взаємовиключними — у кожний момент може виконуватися тільки одна з них. Якби умови були записані так, *що одночасно* могло б виконуватися кілька умов, виконувався б тільки блок або оператор, що випливає за першою правильною умовою.

3.5. Оператор Switch

Оператор switch працює аналогічно операторові if, але дозволяє умовному виразу мати як результат більше двох значень. В операторі if умова при значення true або false. В операторі switch умова може приймати будь-яку кількість різних значень доти, доки вона має простий тип (integer, string або double). Для обробки кожного зі значень, на які потрібно реагувати, необхідно записати оператор case, а також (необов'язково) можна визначити випадок для обробки за замовчуванням будь-яких значень, для яких спеціально не заданий оператор case.

Приклад 3.5. Приклад використання оператора Switch.

Магазин бажає знати, які форми реклами працюють на нього найбільше ефективно. Для цього у форму замовлення можна додати питання. Додайте у форму наступний HTML-код, після чого вона повинна виглядати так:

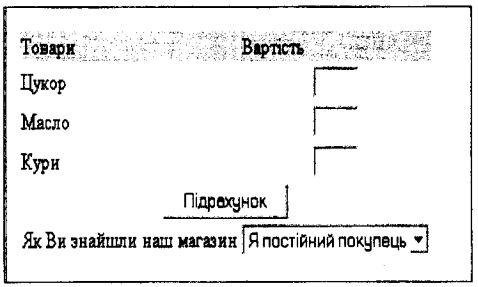


Рис.3.1. **Вікно прикладу 3.5.**

```
<tr>
<td>Як Ви знайшли наш магазин </td>
 <td>select  name="find">
<option  value="a">Я постійний покупець
<option  value="b"> реклама TV
<option  value="c">реклама в газетах
<option  value="d">близько живу
</select> </td> </tr>
```

Наведений HTML-код додав нову змінну форми, значенням якої буде "a", "b", "з" або "d". Цю нову змінну можна було б обробити за допомогою послідовності операторів **if** і **elseif:**

```
If($find=="a")
echo  "<P> постійний покупець.";
elseif($find=="b")
echo  "<P>Спасибі TV .";
 elseif($find=="c")
echo  "<p>дякуючи рекламі в газетах.";
elseif($find="d")
echo  "<P>дякую.";
```

Або ж можна було б записати оператор switch:

```
switch($find)
case "a":
echo ""<P> постійний покупець.";
break;
case "b":
echo " дякуючи TV.";
break;
case "c":
echo  " дякуючи рекламі в газетах"
break;
default:
echo  "<P>Випадково.";
break;
```

Цей оператор switch працює інакше, ніж оператор if або elseif. Оператор if впливає тільки на один оператор, якщо тільки спеціально не використовувати фігурні дужки для створення блоку операторів. Оператор switch діє по-іншому. Коли оператор case в операторі switch активізується, оператори виконуються доти, поки не зустрінеться оператор break. Без нього оператор switch виконував би весь код, що міститься за оператором case, умови якого були б вірними. При досягненні оператора break буде виконуватися рядок коду, що розміщений за оператором switch.

Приклад 3.6. Використання оператору switch з числовими значеннями.

```
<?
$zukor=40;
switch($zukor){
case 50: echo "pogano";break;
```

```
case 40: echo "dobre";break;
}
?>
```

Приклад 3.7. Використання оператору switch з текстовими значеннями.

```
<?
$name="ann";
switch($name){
case "ann": echo "privit<p> ".$name;break;
case "olga": echo "privit<p>".$name;break;
}
?>
```

Приклад 3.8. Використання оператору switch при передачі даних.

```
<form action="handler.php" metod=post>
<select type=text name='switchfield'>
<option value="1">Редагування заголовку
<option value="2">Редагування абзацу
<option value="3">Редагування посилання
</select><br>
<input type=submit name=send
value="Отправить">
<input type="Hidden" name="id_article"
value=><? echo $id_article; ?>
<input type="Hidden" name=id_page val-
ue=<? echo $id_page; ?>>
<input type="Hidden" name=pos value=
<? echo $pos; ?>>
</form>
```

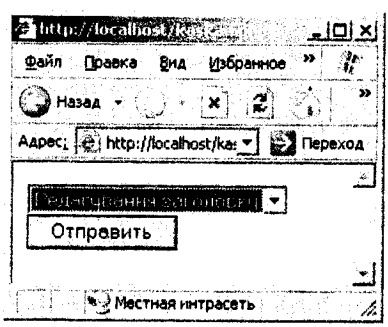


Рис.3.2. **Вікно прикладу 3.8**

handler.php

```
<?
$action="kaskad.php";
switch($_POST['switchfield'])
{
case 1:
$action="51.php";
break;
case 2:
$action="52.php";
break;
case 3:
$action="53.php";
break;
}
echo "<html><head>
<meta http-equiv='Refresh' content='0;URL=$action?
id_article=".$POST['id_article']."&
pos=".$_POST['pos']."&
id_page=".$_POST['id_page']."'>
</html></head>";
?>
```

Один або декілька операторів else, elseif або switch не дозволяють зробити нічого такого, чого не можна було зробити за допомогою набору операторів if. У кожній конкретній ситуації треба використовувати саме ті умовні оператори, які виглядають найбільш читабельно.

3.6. Ітерація: повторення дій

Одне із завдань, з якими комп'ютери завжди справлялися дуже успішно – автоматизація дій , що повторюються. Якщо якійсь оператор потрібно виконувати багато разів, для повторення певних частин програми варто застосувати цикл.

Магазину потрібна таблиця, яка відображає вартість доставки, що буде додаватися до вартості замовлення клієнта. Вартість доставки залежить від відстані й може бути обчислена за допомогою простої формули. HTML-код, що відображає цю таблицю, наведений в прикладі 3.9.

Приклад 3.9. Використання циклів для створення таблиць.

```
<html><body>
<table border = 0 cellpadding = 3>
<tr><td  bgcolor= "#CCCCCC" align = center> Відстань </td>
<td bgcolor = "#CCCCCC" align = center>Ціна</td></tr>
   <?
   $Відстань = 50;
   while ($Відстань<= 250 )
   {
    echo "<tr>\n  <td align = right>$Відстань</td>\n";
    echo "  <td align = right>". $Відстань / 10 ."</td>\n</tr>\n";
    $Відстань += 50;
   }
   ?>
</table></body></html>
```

3.7. Цикли

Найпростіший вид циклу в PHP - цикл while. Подібно оператору if, цей оператор заснований на перевірці умови. Розходження між циклом while і оператором if складається в тому що якщо умова приймає значення

true, оператор if виконує наступний оператор коду тільки один раз. Цикл while виконує блок операторів багаторазово, доти, доки умова приймає значення true.

У загальному випадку цикл while використовується, коли не відомо, скільки ітерацій буде потрібно для виконання умови. Якщо ж потрібна фіксована кількість ітерацій, варто подумати про використання циклу for.

Основна структура циклу while має вигляд

While(умова) вираз

Приклад 3.10. Приклад використання циклу while.

```
$num=1;
while($num<=5)
{
echo $num."<br>";
$num++;}
```

Умова перевіряється на початку кожної ітерації. Якщо вона приймає значення false, блок не буде виконуватися й цикл завершується. Після цього виконується оператор, що є наступним за циклом.

3.7.1. Цикли for

Описаний вище спосіб використання циклів while дуже розповсюджений. Спочатку встановлюється початкове значення лічильника. Перед кожною ітерацією значення лічильника перевіряється в умові. Наприкінці кожної ітерації значення лічильника змінюється. За допомогою циклу for такого роду цикл можна записати в більш компактній формі.

Основна структура циклу for має вигляд

for (вираз1; умова; вираз2) вираз3;

- *Вираз1* виконується один раз на початку циклу. Звичайно в ньому встановлюється початкове значення лічильника.
- Вираз *умова* перевіряється перед кожною ітерацією. Якщо вираз

повертає значення false, ітерація припиняється. Значення лічильника в ньому порівнюється із граничним значенням.

- *Вираз2* виконується наприкінці кожної ітерації, в ньому змінюється значення лічильника.
- *Вираз3* виконується один раз для кожної ітерації. Як правило, це вираз являє собою блок коду й містить властиво тіло циклу.

Приклад циклу while, можна переписати із застосуванням циклу for. PHP-код матиме наступний вигляд:

```
for ($distance=50;$distance<=250;$distance+=50)
{
Echo "<tr>\n<td  align=right>$distance</td>\n";
echo  "<td  align=right>".$distance/10 . "</td>\n</tr>\n";
}
?>
```

У функціональному змісті цикли **while** і **for** ідентичні. Однак цикл **for** більш компактний і містить на два рядки менше. Ці типи циклів еквівалентні - жоден з них нічим не краще й не гірше іншого. У конкретній ситуації можна використати той, який здається більше підходящим.

```
for($i=l;$i<=$numnamas;$i++)
$temp= "name$i" ;
echo $temp."<br>"; // будь-яка необхідна обробка
```

Динамічно створюючи імена змінних, можна звертатися до кожного з полів по черзі.

3.7.2 Цикли do while

Останній тип циклів, які ми розглянемо, **do..while** має вигляд

Do вираз;

while (умова);

Цикл do..while відрізняється від циклу **while** тим, що в ньому умова перевіряється наприкінці. Це означає, що в циклі **do..while** оператор або блок усередині циклу виконується завжди не менш одного разу.

В цьому прикладі, умова має значення **false** і ніколи не може прийняти значення **true,** тому цикл виконується тільки один раз.

```
$num=100;
do{
echo $num."<br>";
}
while($num<1);
```

Запитання для самоконтролю

1. Які Ви знаєте керуючі конструкції?
2. Синтаксис умовного оператора?
3. Що таке цикл?
4. Які цикли є в мові PHP?

Задачі для розв'язування

1. Обчислити значення функції:

$$y = \begin{cases} ae^{\sin x + \cos x}, & \text{при } x < -5; \\ \cos^2 x + \sin^2 x, & \text{при } -5 < x < 5; \\ ab \lg(bx), & \text{при } x > 5, \end{cases}$$

2. Ввести номер студента зі списку, вивести його прізвище.
3. Вести номер місяця, вивести номер кварталу.
4. Ввести номер автобуса, вивести кінцеву зупинку.
5. Скласти програму табуляції значень функції y = Cos(x) для аргументу x, який змінюється від 0 до 1800 з кроком 50.
6. Протабулювати функцію y=cos(x) на відрізку $[0,\pi]$ з кроком h=0.1 і визначити мінімальне значення функції на цьому проміжку.

РОЗДІЛ 4. МАСИВИ

Змінні розглянуті раніше є скалярними. Масив - це змінна, в якій зберігається набір елементів. Скалярна змінна — це іменована комірка пам'яті, у якій зберігається значення, масив – це іменована комірка пам'яті, у якій зберігається *набір* значень, що дозволяє групувати звичайні скалярні значення. Список товарів, що поставляють у Магазин, може бути представлений як масив.

Після того, як інформація збережена у вигляді масиву, з нею можна виконувати ряд операцій. Використовуючи конструкції циклів, можна вводити та виводити елементи масиву, виконуючи ті самі дії над кожним елементом масиву. Весь обсяг інформації можна переміщати як єдиний блок. Таким чином, всі елементи можуть бути передані у функцію за допомогою одного рядка коду. Наприклад, якщо потрібно впорядкувати товари за алфавітом, для цього слід використати в PHP – функцію **sort().**

Значення, що зберігаються в масиві, називаються елементами масиву. Кожний елемент масиву має пов'язаний з ним *індекс* (називаний також *ключем),* що використовується для доступу до елемента.

У більшості мов програмування масиви мають чисельні індекси, які, починаються з нуля або одиниці. PHP також підтримує такий тип масивів. Але крім того, PHP підтримує також *асоціативні* масиви. Як індекси в асоціативних масивах можуть використовуватися практично будь-які значення, але, як правило, такими є рядки. Розгляд почнемо із чисельно індексованих масивів.

4.1.Чисельно індексовані масиви. Ініціалізація чисельно індексованих масивів

В PHP існують наступні методи ініціалізації масивів.

Приклад 4.1. Приклад ініціалізації масиву.

```
<?
  $car[] = "passenger car";
  $car[] = "land-rover";
  echo($car[1]); // виводить "land-rover"
?>
```

Індекс масиву можна указати явно:

```
<?
  $car[0] = "passenger car";
  $car[1] = "land-rover";
  echo($car[1]); // виводить "land-rover"
?>
```

Розглянемо альтернативний спосіб, для цього запишемо наступний рядок коду:

$products=array("zukor","maclo","kuri");

У результаті створюється масив products, що містить три заданих значення: "zukor", " maclo " і " kuri ". Зверніть увагу, що подібно інструкції echo, array() в дійсності є скоріше мовною конструкцією, ніж функцією.

Залежно від необхідного вмісту масиву, його не завжди треба ініціалізувати вручну. При наявності даних, які потрібні в іншому масиві, можна просто копіювати один масив в інший за допомогою операції =. Якщо в масиві необхідно зберігати зростаючу послідовність чисел, для автоматичного його створення можна використати функцію range(). Наступний рядок коду створює масив numbers, що містить числа від 1 до 10:

$numbers=range(1,10)

Якщо інформація зберігається у файлі на диску, елементи масиву можна завантажити безпосередньо з файлу. Якщо дані масиву зберігаються в базі даних, їх можна завантажити безпосередньо з бази даних. Можна

також використати різні функції для виділення частини масиву або для зміни порядку проходження елементів масиву.

4.2. Доступ до елементів масиву

Для доступу до змінної використовується її ім'я. Якщо змінна є масивом, доступ до її елементів здійснюється через ім'я змінної й ключ, або індекс. Ключ, або індекс, вказує, до якого значення реалізується доступ. Індекс вказується у квадратних дужках після імені.

Наприклад, щоб використати вміст масиву products, необхідно ввести $products[0], $products[l] і $products[2].

Нульовий елемент є першим елементом масиву. Ця ж схема нумерації використовується в C++, Java і ряді інших мов програмування. Як і у випадку інших змінних, зміст елементів масиву змінюється за допомогою операції =. Так наступний рядок замінить перший елемент масиву "zukor" елементом "moloko".

$products[0]=“ moloko ".

Для відображення вмісту масиву можна ввести:

echo "$products[0] $products[1] $products[2] $ptoducts[3]".

Подібно іншим змінним PHP, масиви не потрібно ініціалізувати або створювати заздалегідь. Вони автоматично створюються при першому використанні. Наступний код створить цей же масив $products:

$products[0]="zukor”,

$products[1]=" maclo",

$products[2]="kuri".

Якщо масив $products ще не існує, перший рядок створює новий масив тільки з одним елементом. Наступні рядки додають значення в масив.

4.3. Використання циклів для доступу до елементів масиву

Оскільки масив індексується по послідовних номерах, для спрощення відображення його вмісту можна використати цикл for:

```
for($i=0;$i<3;$i++) echo "$products[$i]<br>";
```

Приклад 4.2. Введення числових значень масиву.

```
<?
$number=array(1,2,3);
for($i=0;$i<count($number);$i++)
{
echo ($number[$i]);
}
?>
```

Приклад 4.3. Введення текстових значень масиву.

```
<?
$number=array("1","2","3");
for($i=0;$i<count($number);$i++)
{
echo "$number[$i])";
}
?>
```

Цей цикл створить таке ж виведення, як і попередній код, але для виводу кожного з елементів великого масиву буде потрібно введення меншого обсягу коду вручну. Можливість використання простого циклу для доступу до кожного елемента - чудова властивість чисельно індексованих масивів. Виконати циклічний перегляд в асоціативних масивах не настільки легко, але зате вони дозволяють привласнювати індексам осмислені значення.

4.4. Визначення IP – адреси відвідувача

Для Визначення IP – адреси відвідувача слід звернутися до елементу суперглобального масиву($_SERVER["REMOTE_ADDR"])

Приклад 4.4. Визначаємо IP адресу.

```
<?
// Визначаємо IP адресу
 $ip=($_SERVER["REMOTE_ADDR"]);
//сторінку
echo "Сторінку $_SERVER[PHP_SELF] відвідав
користувач з IP-адресою".$_SERVER["REMOTE_ADDR"];
?>
```

Визначення сторінки, на якій опрацьовується скрипт

```
<?
 echo  $_SERVER['PHP_SELF'];
?>
```

4.5. Асоціативні масиви

При створенні масиву товарів ми надали PHP можливість привласнити кожному елементу індекс, обумовлений за замовчуванням. Це означає, що перший доданий елемент став 0 елементом, другий - 1 і т.д. PHP підтримує також асоціативні масиви. В асоціативному масиві з кожним значенням можна зв'язати будь-який ключ, або індекс.

4.5.1. Ініціалізація асоціативного масиву. Доступ до елементів асоціативного масиву

Наступний код створює асоціативний масив, у якому назви товарів використовуються як ключі, а їхньої ціни - як значення.

```
$price=array("zukor"=>100, "maclo"=>10, "kuri"=>4  );
```

Як і раніше, доступ до елементів здійснюється через ім'я змінної й ключ, тому до інформації, збереженої в масиві prices можна звернутися як

$prices["zukor "], $prices["maclo "] і $pices[" kuri "].

Наприклад:

```
echo $price["zukor "];
```

Подібно чисельно індексованим масивам, асоціативні масиви можуть створюватися й ініціалізуватися по одному елементу. Наступний код створює цей же масив $prices. Замість створення масиву із трьома елементами ця версія створює масив тільки з одним елементом, а потім додає в нього ще два елементи.

```
$prices=array("zukor"=>100)
$prices["maclo"]=10;
$prices["kuri"]=4;
```

Нижче наведений ще один еквівалентний фрагмент коду. У цій версії масив взагалі не створюється явно. Він створюється при додаванні в нього першого елемента

```
$prices["zukor "]=100;
$prices["maclo "]=10;
$prices["kuri "]=4;
```

4.6. Організація циклів з використанням each() і list()

Оскільки в асоціативному масиві індекси не є числами, для роботи з масивом не можна використати простий лічильник у циклі **for.** Наступний код виводить елементи нашого масиву $prices.

Приклад 4.5. Використання функцій each() і list().

```
while($element=each($price))
{
echo $element["key"];
echo"-";
echo $element["value"];
echo"<br>";}
```

Раніше були розглянуті цикли **while** і оператор **echo.** У наведеному вище прикладі коду використовується функція each(), що раніше не зустрічалася. Ця функція повертає поточний елемент масиву й робить поточ-

ний наступний елемент. Оскільки функція **each()** викликається усередині циклу **while,** вона по черзі повертає кожний з елементів масиву й припиняє своє виконання по досягненні кінця масиву.

У цьому прикладі змінна **$element** є масивом. При виклику функції **each()** вона надає масив із чотирма значеннями й чотирма індексами комірок масиву. Комірки **key** і 0 містять ключ поточного елемента, а комірки **value** і **1** - значення поточного елемента. Хоча вибір комірок не має значення, ми використали іменовані комірки, а не нумеровані.

Це ж можна зробити більше витонченим і звичним способом - використати функцію **list()** для поділу масиву на ряд значень. Два значення, передані функцією each()- можна розділити в такий спосіб:

$list($product,$price)=each($prices).

Цей рядок використовує функцію **each()** для одержання поточного елемента з масиву **$prices,** повертає його у вигляді масиву й робить наступний елемент поточним. Крім того, функція **list()** використовується для перетворення елементів 0 і 1 масиву, що повертаються функцією **each()** у дві нових змінних: **$product** і **$price.** Можна циклічно переглянути весь масив **$prises,** повторюючи його вміст, скориставшись наступним коротким сценарієм

```
$prices=array("zukor"=>100, "maclo"=>10, "kuri"=>4  );
while (list($product,$price)= each($prices))
echo "$product-$price<br>".
```

Цей сценарій працює, як і попередній, але його легше читати, оскільки функція **list()** дозволяє привласнювати імена змінним.

При використанні функції **each()** варто пам'ятати, що масив відслідковує поточний елемент. Якщо в тому самому сценарії елемент необхідно використати двічі, за допомогою функції **reset()** буде потрібно знову вста-

новити поточний елемент на початок масиву. Щоб знову виконати циклічний перегляд масиву **prices** треба ввести:

```
reset($prices);
while (list($product,$price)= each($prices))
echo "$product-$price<br>";
```

У результаті поточний елемент буде знову встановлений на початок масиву, що дозволить знову виконати в ньому перегляд.

4.7. Цикл foreach

Систаксис циклу foreach.

```
Foreach($array as [$key=>]$value)
{
оператори
}
```

Приклад 4.6. Використання циклу foreach.

```
<?
$number=array("first"=>"1","second"=>"2","third"=>"3");
foreach($number as $index=>$val)echo "$index=>$val<br>"
?>
```

Цикл foreach без ключа

```
<?
$number=array("first"=>"1","second"=>"2","third"=>"3");
foreach($number as $index)echo $index;
?>
```

Виводить 123

4.8. Багатовимірні масиви

Масив не обов'язково повинен бути простим списком ключів і значень - кожна комірка масиву може містити інший масив. Таким чином, можна створити двовимірний масив. Двовимірний масив можна уявити со-

бі у вигляді матриці, або таблиці, ширина й висота якої вимірюється рядками й стовпцями.

Для визначення даних у масиві запишемо:

```
$products = array( array ( "бакалія", "цукор", 100 ),
                   array( "масла", "масло", 10),
                   array ( "м'ясо", "кури", 4));
```

Для відображення вмісту цього масиву можна було б вручну звернутися до кожного з елементів у наступному порядку:

```
echo " |".$products[0] [0] . " | ". $products [0] [1] . " | " .$products [0] [2] ." |
<br>";
echo " |".$products[1] [0] . " | ". $products [1] [1] . " | " .$products [1] [2] ." |
<br>";
echo " |".$products[2] [0] . " | ". $products [2] [1] . " | " .$products [2] [2] ." |
<br>";
```

Приклад 4.7. Використання оператора for в багатовимірних масивах.

```
for($row=0;$row<3;$row++)
{
for($column=0;$column<3;$column++)
{
echo "|".$products[$row][$column];
}
echo "|<br>";
}
```

4.9. Сортування масивів

Приклад 4.8. Використання функції sort().

Наступний код приводить до впорядкування масиву за абеткою:

```
$products=array("zukor","maslo","kuri") ;
sort ($products);
```

```
while($element=each($products))
{
echo $element["key"];
echo"-";
echo $element["value"];
echo"<br>";}
```

Тепер елементи масиву будуть розташовані в наступному порядку: kuri, maslo, zukor. Значення можна впорядковувати також у цифровому порядку. При наявності масиву, який містить ціни на товари, що поставляють в Магазин, можна виконати сортування в порядку зростання чисельних значень.

```
$price=array(100,10,4);
sort($prices);
```

4.10. Використання функцій asort() і ksort() для сортування асоціативних масивів

Якщо для зберігання інформації про товари і їхні ціни використовується асоціативний масив потрібно використати інші функції сортування, що забезпечує спільне збереження ключів і значень при сортуванні.

Наступний код створює асоціативний масив, що містить три товари й пов'язані з ними ціни, а потім сортує масив у порядку збільшення цін

```
$products=array("zukor"=>100, "maclo"=>10, "kuri"=>4 );
asort ($products);.
```

Функція asort() упорядковує масив у відповідності зі значеннями елементів. У масиві значення - це ціни, а ключі - текстові описи. Якщо сортування виконати не за цінами, а по описах, варто використати функцію ***ksort(),*** що виконує сортування не за значеннями, а по ключах. Наступний код приведе до впорядкування ключів масиву за абеткою

```
$products=array("zukor"=>100, "maclo"=>10, "kuri"=>4 );
```

ksort ($products);

Функція arsort() виконує сортування одномірного асоціативного масиву в порядку убування значень елементів. Функція krsort() виконує сортування одномірного асоціативного масиву в порядку убування значень ключів елементів.

4.11. Пошук елемента в масиві

Пошук елемента в масиві здійснюється за допомогою функції **in_array().**

Приклад 4.9. Пошук елемента в масиві.

```
<?
$number=array(0.57,'21.5',3);
if(in_array(21.5,$number)) echo "Значення знайдено";
else
echo "Значення  не знайдено";
?>
```

Приклад 4.10. Виведення випадкового елементу масиву за допомогою функції shuffle().

```
<?
$str="Ми вивчаємо";
$input=array("математику "," інформатику ","Web- програмування ");
shuffle($input);
echo $str.$input[1];
?>
```

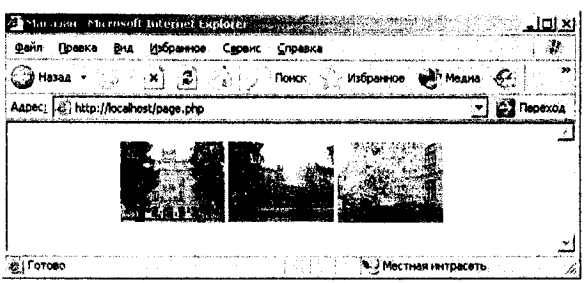


Рис.4.1. Вікно прикладу 4.10

Якщо необхідно представити товари на сайті, можна створити динамічну титульну сторінку.

Приклад 4.11. Створимо динамічну WEB-сторінку за допомогою функції shuffle().

```
<?
$pictures=array("1.jpg","2.jpg","3.jpg","5.jpg","6.jpg","7.jpg");
shuffle($pictures);
?>
<html>
<head><title>КПІ</title></head>
<body><center>
<table>
<tr>
<?
for($i=0;$i<3;$i++)
{
echo"<td aling=center><img src=\"";
echo $pictures[$i];
echo"\"width=100 heidht=100></td>";
}
?>
</tr></table></center></body></html>
```

4.12. Сортування багатовимірних масивів

Сортування масивів, які мають більше одного виміру, або в порядку, що відрізняється від алфавітного або цифрового, більш складна задача. У PHP є можливість порівняння двох чисел або двох текстових рядків, але в багатомірному масиві елемент є масивом. У PHP відсутня можливість порівняння двох масивів, тому для їхнього порівняння необхідно створити метод.

4.13. Визначені користувачем функції сортування

Нижче наведений двовимірний масив. У масиві зберігаються назви товарів, що поставляють у Магазин, їхні коди й ціни.

```
$products = array( array( "бакалія",  "цукор",100),
                   array ( "масла"," масло",10),
                   array ( "м'ясо"," кури",4);
```

Товари можна упорядкувати в алфавітному порядку по назвам або по цінам. В обох випадках слід використовувати функцію usort() та вказати PHP яким чином слід сортувати елементи.

Переміщення усередині масиву: функції each, current(), reset, end(), next(), pos() і prev()

Раніше вже згадувалося, що кожний масив має внутрішній вказівник, що вказує на поточний елемент масиву. Раніше ми побічно задіяли вказівник при використанні функції **each(),** але його можна використати й маніпулювати ним безпосередньо.

При створенні нового масиву поточний вказівник ініціалізується так, щоб вказувати на перший елемент масиву. Виклик функції **current($array_name)** повертає перший елемент.

Виклик функції **next()** або **each()** переміщає вказівник уперед на один елемент. Виклик функції **each($array_name)** повертає поточний елемент, перш ніж перемістити вказівник. Функція **next()** поводиться інакше - виклик функції **next($array_name)** приводить до переміщення вказівника, а потім до повернення нового поточного елемента.

Функція **reset()** повертає вказівник на перший елемент масиву. Аналогічно, виклик функції **end($array_name)** переміщає вказівник в кінець масиву. Функції **reset()** і **end()** повертають, відповідно, перший і останній, елементи масиву.

Підрахунок елементів у масиві: функції count(), sizeof() і array_count_values()

Для підрахунку кількості елементів у масиві використовується функція **count()**. Функція **sizeof()** має аналогічне призначення. Обидві ці функції повертають кількість елементів у переданому їм масиві. Так значення лічильника кількості елементів буде дорівнює 1 для звичайної скалярної змінної та 0 при передачі порожнього масиву, або змінної, котра не встановлена.

Функції **array_count__values()** складніше. Якщо викликати **array_count_values($array),** функція підраховує, скільки разів кожне *унікальне* значення зустрічається в **масиві $array.** Вона повертає асоціативний масив, що містить таблицю частоти використання елементів. Цей масив містить в якості ключів всі унікальні значення масиву **$array.** Кожний ключ має чисельне значення, що вказує, скільки разів відповідний ключ зустрічається в масиві **$array.**

Існують два основних способи зберігання даних: у двовимірних (звичайних) файлах і в базах даних.

Двовимірний файл може мати безліч форматів, але в загальному випадку під *двомірним (flat) файлом* будемо розуміти простий текстовий файл. У розглянутому прикладі замовлення клієнта записуються в текстовий файл, по одному замовленню в рядок.

Цей спосіб настільки ж простий, як і обмежений, що буде показано далі. Якщо доводиться мати справу з досить великим обсягом інформації, імовірно краще скористатися базами даних. Однак, двовимірні файли знаходять досить широке застосування, тому в ряді ситуацій необхідно володіти технологією їх застосування. Давайте візьмемо масив і збільшимо значення кожного його елемента на одиницю.

Приклад 4.12. Робота з масивами.

```
<?
$number = array ("1"=>"15", "2"=>"20", "3"=>"25");
function printarray ($item, $key) {
echo "$key=>$item<br>\n";
}
function add_array (&$item, $key)
{
$item = $item + 1;
  }
echo("Before:<br>");
array_walk ($number, 'printarray');
echo("After:<br>");
array_walk ($number, 'add_array');
array_walk ($number, 'printarray');
?>
```

Питання для самоконтролю

1. Що таке одновимірні та двовимірні масиви?
2. Як описуються двовимірні масиви?
3. Що таке асоціативні масиви?
4. Для чого використовуються функції each() і list()?
5. Які функції використовуються для сортування масивів?

Задачі для розв'язування

1. Використовуючи асоціативні масиви написати прайс лист фірми.
2. Організувати цикл з використанням функцій each() і list().
3. Знайти індекси мінімального елемента масиву.
4. Знайти суму елементів масиву.
5. Обчислити суму діагональних елементів масиву.

РОЗДІЛ 5. ВИКОРИСТАННЯ ФУНКЦІЙ У PHP

Функції існують у багатьох мовах програмування. Вони використовуються для виділення коду, що виконує окрему, певну задачу. Це спрощує читання коду й дозволяє повторно його використовувати при кожному виконанні задачі.

Під функцією розуміють незалежний модуль коду, що встановлює інтерфейс виклику, виконує певну задачу й, необов'язково, повертає результат.

5.1. Базова структура функції

Функція оголошується за допомогою ключового слова function, після якого у фігурних дужках записуються різні оператори, що становлять тіло функції:

```
function MyFunction()
{
 // оператори
}
```

Якщо функція приймає аргументи, то вони записуються як змінні в оголошенні функції. Аргумент функції являє собою змінну, передану в тіло функції для подальшого використання в операціях. У випадку, коли функція приймає більше одного аргументу, ці змінні розділяються комами:

```
function MyFunction($var, $var1, $var2)
```

Якщо функція повертає яке-небудь значення, у тілі функції обов'язково повинен бути присутнім оператор return. Наприклад:

```
function MyFunction()
{
return $ret; // повертається значення змінної $ret
}
```

Приклад 5.1. Приклад простої функції.

```
<?
 function get_sum()
 {
  $var = 5;
  $var1 = 10;
  $sum = $var + $var1;
  return $sum;
 }
 echo(get_sum()); // виводить 15
?>
```

У цьому прикладі показана функція, що обчислює суму двох чисел. Ця функція не приймає жодного аргументу, а просто обчислює суму й повертає отриманий результат. Після цього, вона викликається в тілі оператора echo для виведення результату в браузер. Модифікуємо цю функцію так, щоб вона не повертала отриманий результат, а виводила його в браузер. Для цього досить внести оператор echo у тіло функції:

Приклад 5.2. Використання функцій без аргументу.

```
<?
 function get_sum()
 {
  $var = 5;
  $var1 = 10;
  $sum = $var + $var1;
  echo $sum;
 }
 get_sum();
?>
```

Змінні $var й $var1 ми можемо оголосити як аргументи й у цьому випадку в тілі функції їх визначати не треба:

Приклад 5.3. Використання функцій з аргументами.

```
<?
 function get_sum($var, $var1)
 {
  $sum = $var + $var1;
  echo $sum;
 }
 get_sum(5,2); // виводить 5
?>
```

Змінне, утримуюче значення, передане через аргумент, називається параметром функції.

5.2. Найменування функцій

Під час присвоєння імен функціям найбільше важливо враховувати, що ім'я повинне бути коротким і описовим. Якщо функція створює верхній колонтитул сторінки, що підходить ім'ям може бути pageheader() або page_header().

На імена функцій накладаються наступні обмеження:

1. Функція не може мати те ж ім'я, що в існуючої функції.
2. Ім'я функції може містити тільки букви, цифри й символи підкреслення.
3. Ім'я функції не може починатися із цифри.

Багато мов програмування допускають повторне використання імені функції - це властивість називається *перевантаженням функцій.* Однак PHP не підтримує перевантаження функцій, тому функція не може мати ім'я, що збігається з ім'ям вбудованої або існуючої функції. Майте на увазі, що хоча будь-який PHP-сценарий розпізнає всі вбудовані функції, обумо-

влені користувачем функції існують тільки в тих сценаріях, у яких вони оголошені. Це означає, що ім'я функції можна повторно використати в іншому файлі, але це може приводити до плутанини, тому подібних ситуацій варто уникати.

Наступні імена функцій припустимі:

name(); name2(); name_three(); _namefour();

і неприпустимі: 5name(); name-six(); fopen();(вже існує).

5.3 Виклик функцій

Наступний рядок – найпростіше звертання до функції:
function_name().

Вона викликає функцію з ім'ям function_name(), яка не вимагає параметрів. Цей рядок коду ігнорує будь-яке значення, яке може бути повернено цією функцією.

Більшість функцій викликаються саме в такий спосіб. Функція phpinfo() часто корисна під час тестування, оскільки вона відображає встановлену версію PHP, інформацію про PHP, параметри установки Web-сервера й значення різних змінних PHP і сервера. Ця функція не приймає ніяких параметрів, і в загальному випадку повертає значення, що ігнорується; тому виклик phpinfo() буде мати наступний вид: phpinfo().

Більшість функцій вимагають передачі одного або більше параметрів - інформації, переданої у функцію під час її виклику, яка впливає на результат її виконання. Параметри передаються шляхом передачі даних або імені змінної, у круглі дужки слідом за ім'ям функції. Звертання до функції з параметром має такий вигляд:

function_name("parameter") .

У цьому випадку використаний параметр - рядок, що містить тільки слово parameter, але наступні звернення також є звертаннями до функції:

function_name (2) ; function__name (5.993); function_name($variable) .

В останньому рядку змінна $variable може бути будь-якого типу, у тому числі й масивом. Параметр може бути даним будь-якого типу, але конкретні функції звичайно вимагають передачі конкретних типів даних. Кількість параметрів функції та тип їх даних, можна з'ясувати із *прототипу* функції. Наприклад, як виглядає прототип функції fopen():

int fopen (string filename, string mode, [int use_include_path]).

Прототип повідомляє ряд особливостей і важливо вміти правильно інтерпретувати ці специфікації. У цьому випадку слово int перед ім'ям функції вказує, що ця функція буде повертати цілочисленне значення. Параметри функції виокремлюються у круглі дужки. У цьому випадку в прототипі зазначені три пари параметрів *filename* (ім'я_файлу) і *mode* (режим) є рядками, а *use_inctude_path* - цілочисленним значенням.

Квадратні дужки навколо *use_include_path* показують, що цей параметр необов'язковий. Для необов'язкових параметрів можна передавати значення або ж їх можна ігнорувати - у цьому випадку буде використовуватися значення, за замовчуванням.

Після ознайомлення із прототипом цієї функції зрозуміло, що наступний фрагмент коду буде припустимим викликом функції fopen():

```
$name = "myfile.txt";
$openmode = "r";
$fp = fopen($name, $openmode );
```

Цей код викликає функцію з ім'ям fopen(). Значення, що повертає функція, буде зберігатися в змінній $fp. У функцію передається змінна $name, що містить рядок, ім'я файлу, і змінна $openmode - рядок, що представляє необхідний режим відкриття файлу. Третій параметр не заданий.

5.4. Звернення до невизначеної функції

При спробі викликати функцію, що не існує, відобразиться повідомлення про помилку.

Як правило, виведені PHP повідомлення про помилки дуже корисні. Дане повідомлення точно вказує файл і рядок сценарію, у яких відбулася помилка й ім'я функції, спроба виклику якої привела до помилки. Це повинно істотно спростити пошук і виправлення помилки.

При відображенні повідомлення про помилку необхідно перевірити два моменти:

1. Чи правильно зазначене ім'я функції?
2. Чи існує дана функція у даній версії PHP? Правильне написання імені функції не завжди легко запам'ятати.

Наприклад, деякі імена функцій, що складаються із двох слів, містять символ підкреслення між словами, а в деяких - немає. Так, в імені функції stripslashes() два слова злиті в одне, у той час як в імені функції strip_tags() вони розділені символом підкреслення. Введення імені функції з помилкою у виклику функції приводить до помилки.

Для з'ясування синтаксису конкретної функції, можна звернутися до інтерактивного керівництва на сайті www.php.net. Спроба виклику функції, що не оголошена у даній версії, приведе до генерації повідомлення про помилку.

5.5. Регістр і імена функцій

Зверніть увагу, що імена функцій не залежать від регістра, тому всі звернення finiction_name(), Function_Name() або FUNCTION_NAME() є припустимими й приведуть до того самого результату. Прописні букви можна використати в імені функції будь-яким чином, що на вашу думку, полегшує читання, але при цьому варто прагнути до однаковості.

Важно відзначити, що імена функцій поводяться інакше, чим імена змінних. Імена змінних залежать від регістра, і тому $Name і $name — різні змінні, але Name() і name() — та сама функція.

В попередніх главах приводилося багато прикладів використання деяких вбудованих функцій PHP. Однак реальна сила мови програмування проявляється в можливості створення власних функцій.

Для чого потрібно створювати власні функції?

Функції, вбудовані в PHP, дозволяють взаємодіяти з файлами, використовувати бази даних, створювати графічні зображення й підключатися до інших серверів. Однак у багатьох випадках прийдеться виконувати щонебудь, не передбачене розроблювачами мови.

На щастя, дії програміста не обмежуються вбудованими функціями, оскільки для виконання задачі можна створювати свої власні функції. Імовірно, створюваний код буде являти собою комбінацію існуючих функцій і спеціалізованої логіки виконання тієї або іншої задачі. При створенні блоку коду для виконання задачі, що, швидше за все, прийдеться використати в декількох місцях сценарію або в декількох сценаріях, має сенс оголосити цей блок у вигляді функції.

Оголошення функції дозволяє використати створений код так само, як і вбудовані функції. Досить просто викликати функцію й передати в неї необхідні параметри. Це означає, що ту саму функцію можна багаторазово викликати й використати в сценарії.

5.6. Параметри

Щоб мати можливість виконувати свої задачі, більшість функцій вимагають передачі в них одного й більше параметрів. Параметр дозволяє передавати дані у функцію. Нижче наведений приклад функції, що вимагає передачі в неї параметра. Ця функція приймає одномірний масив і відображає його у вигляді таблиці.

У розглянутих прикладах аргументи функції передаються за значенням, тобто значення параметрів змінюється тільки усередині функції, і ці зміни не впливають на значення змінних за межами функції:

Приклад 5.4. Передача аргумента функції за значенням.

```
<?
 function get_sum($var) // аргумент передається за значенням
 {
  $var = $var + 5;
  return $var;
 }
 $new_var = 20;
 echo(get_sum($new_var)); // виводить 25
 echo("<br>$new_var"); // виводить 20
?>
```

Для того щоб змінні передані функції зберігали своє значення при виході з її, застосовується передача параметрів по посиланню. Для цього перед ім'ям змінної необхідно помістити амперсанд (&):

function get_sum($var, $var1, &$var2).

У цьому випадку змінні **$var** й **$var1** будуть передані за значенням, а змінна **$var2** - по посиланню. У випадку, якщо аргумент передається по посиланню, при будь-якій зміні значення параметра відбувається зміна змінн-аргументу.

Приклад 5.5. Передача аргумента функції по посиланню.

```
<?
 function get_sum(&$var) // аргумент передається по посиланню
 {
  $var = $var + 5;
  return$var;
```

```
}
$new_var = 20;
echo(get_sum($new_var)); // виводить 25
echo("<br>$new_var"); // виводить 25
?>
```

Приклад 5.6. Приклад використання функції reset().

```
function cteate_table($number)
{
echo "<table border=1>";
reset($number);//вказує на початок
$value=current($number);
while($value)
{
echo "<tr><td>$value</td></tr>\n";
$value=next($number);
}
echo "</table>";
}
```

Якщо викликати функцію create_table() у такий спосіб:

```
$number=array("line one",2,3);
cteate_table($number);
```

Як і у випадку убудованих функцій, створені користувачем функції можуть мати кілька параметрів і необов'язкові параметри. Функцію create_table() можна вдосконалити багатьма способами, але одним з них могло б бути надання користувачеві можливості вказувати рамку або інші атрибути. Нижче наведена поліпшена версія функції. Вона досить подібна до попередньої й дозволяє при бажанні визначати ширину рамки, відстань між комірками спосіб заповнення комірок.

Перший параметр функції create_table2() як і раніше, обов'язковий. Наступні три – не обов'язкові, оскільки для них визначені значення за замовчуванням.

Приклад 5.7. Створення таблиці.

```
Function cteate_table2($number,$border=1,$cellpadding=4, $cellspacing=4)
    {
    echo "<table border=$border
    cellpadding=$cellpadding"."cellspacing=$cellspacing>";
    reset($number);
    $value=current($number);
    while($value)
    {
    echo "<tr><td>$value</td></tr>\n";
    $value=next($number);
    }
    echo "</table>";
    }
    $number=array("line one",2,3);
    cteate_table2($number,3,8,8);
    ?>
```

Передача параметра дозволяє передати у функцію дані, які були поза функцією - у цьому випадку, масив $number.

5.7. Область дії змінних

При необхідності використання змінних усередині, включеного файлу ми просто оголошуємо їх у сценарії перед оператором require() або include(), але при використанні функції ми явно передаємо ці змінні у функцію.

Діапазон дії змінних залежить від того, де змінна видима й застосовується. У різних мовах програмування діють різні правила, що встановлюють діапазон дії змінних. У PHP діють дуже прості правила:

- Змінні, які оголошені усередині функції, діють в області від оператора, у якому вони оголошені до закриваючої дужки наприкінці функції. Ця область називається *областю функції,* а такі змінні - *локальними змінними.*
- Змінні, які оголошені поза функцією, діють в області від оператора, у якому вони оголошені до кінця файлу, але не усередині функцій. Ця область називається *глобальною областю,* а такі змінні - *глобальними змінними.*
- Використання операторів require() і include() не впливає на область дії змінних. Якщо оператор використовується усередині функції, застосовується область функції. Якщо він використається не усередині функції, застосовується глобальна область.
- Ключове слово global може використовуватися для вказівки вручну того, що змінна, котру визначено або яка використовується всередині функції, буде мати глобальну область дії.
- Змінні можуть бути вручну вилучені за допомогою функції *unset($variable_name).* Якщо змінна вилучена, вона більше не перебуває в області дії.

Наступні приклади допоможуть розібратися з описаними концепціями.

Наступний код не створює ніякого виведення. У ньому оголошується змінна $var усередині функції fn(). Оскільки ця змінна оголошується усередині функції, вона має область дії від місця її оголошення до кінця функції. При новому звертанні до $var поза функцією, створюється нова змінна $var. Ця нова змінна має глобальну область дії й буде видима до кінця

файлу. На жаль, якщо єдиний оператор, в якому використовується ця нова змінна $var -echo, вона ніколи не буде мати значення.

Приклад 5.8. Змінна оголошена в функції.

```
<?
function fn()
{
$var="contents";
}
echo $var;
?>
```

Виводить помилку.

Наступний приклад протилежний попередньому. Ми оголошуємо змінну поза функцією, а потім намагаємося її використати усередині функції.

Приклад 5.9. Змінна оголошується поза функцією.

```
<?
function fn()
{
$var="contents";
}
echo fn();
?>
```

Виведення немає.

```
<?
function fn($var)
{
echo "inside the function, \$var=".$var."<br>";
$var="contents2";
```

```
echo "inside the function, \$var=".$var."<br>";
}
$var="contents 1";
fn($var);
echo "outside the function, \$var=".$var."<br>";
?>
```

Функція не виконуються доти, поки вона не буде викликана, тому першим виконуваним оператором є $var="contents 1". Він створює змінну $var, що має глобальну область дії й зміст "contents 1". Наступним виконується оператор - звертання до функції fn(). Рядки усередині оператора виконуються по черзі. Перший рядок у функції звертається до змінної $var і виконується. У результаті створюється перший рядок виведення. Наступний рядок, усередині функції, встановлює значення змінної рівним "contents 2". Оскільки дії виконуються усередині функції, це змінює значення локальної змінної $var, а не глобальної. Другий рядок, підтверджує виконання цієї зміни.

На цьому виконання функції завершується, тому виконується заключний рядок сценарію. Оператор echo демонструє, що значення глобальної змінної не змінилося.

Якщо потрібно, щоб змінна, створена усередині функції, була глобальною, можна використати ключове слово global, як показано в наступному прикладі.

Приклад 5.10. Глобальні змінні.

```
<?
function fn()
{
global $var;
$var="contents";
```

```
echo "inside the function, \$var=".$var."<br>";
}
fn();
echo "outside the function, \$var=".$var."<br>";
?>
```

У цьому прикладі змінна $var була явно оголошена як глобальна, тобто після виклику функції змінна буде існувати й поза функцією.

Зверніть увагу, що змінна визначена в області, починаючи з того місця, де виконується рядок global $var. Функцію можна було б оголосити вище *або нижче* того місця, де вона викликається. (Зверніть увагу, що область дії функції значно відрізняється від області дії змінної!). Місце оголошення функції не істотно - важливо лише те, де ми викликаємо функцію, тоді виконується код, що знаходиться всередині неї.

Ключове слово global можна використати також на початку сценарію при першому використанні змінної для оголошення того, що весь сценарій повинен бути областю її дії. Імовірно, це - найпоширеніше використання ключового слова global.

Цілком припустимо повторно використати ім'я змінної усередині й зовні функції без взаємного впливу між ними. Однак, у загальному випадку робити це не рекомендується, оскільки, не вникнувши в код і не взявши до уваги область дії змінних, користувачі можуть вирішити, що це одна змінна.

Розглянемо, ще приклади передачі по посиланню та передачі за значенням. Якщо потрібно створити функцію increment(), яка дозволяє збільшувати значення, програміст може створити її в такий спосіб:

Приклад 5.11. Передача даних за значенням.

```
<?
function increment($value,$amount=1)
```

```
{
$value=$value+$amount;
}
$value=10;
increment($value);
echo $value;
?>
```

Цей код не буде працювати. У результаті виконання наступного тесту виводиться "10". Як бачимо, значення змінної $value не змінилося.

Це пов'язане із правилами області дії. Цей код створює змінну $value, що містить значення 10. Потім програма викликає функцію increment(). Змінна $value створюється у функції при її виклику. До значення додається 1, тому усередині функції значення $value дорівнює 11 доти, поки виконання функції не завершується й не здійснюється повернення до коду, що її викликав. У цьому коді, поза функцією, змінна $value - це інша змінна, створена в глобальній області, і тому вона залишається незмінною.

Один зі способів рішення цієї проблеми - оголошення змінної $value у функції як глобальної, але це означає, що для використання цієї функції змінну, значення якої потрібно збільшити, обов'язково треба назвати $value.

Більш раціональним підходом було б використання *передачі по посиланню.* Звичайний спосіб виклику параметрів функції називається *передачею за значенням.* При передачі параметра створюється нова змінна, котра містить передане значення. Вона є копією вихідної змінної. Це значення можна змінювати будь-яким чином, але при цьому значення вихідної змінної поза функцією залишається незмінним.

Деколи використовується *передача по посиланню.* У цьому випадку при передачі параметра, замість того щоб створювати нове значення, фун-

кція приймає посилання на вихідну змінну. Це посилання має ім'я змінної, що починається зі знака долара, і може використовуватися зовсім так само, як будь-яка інша змінна. Розходження полягає в тому, що замість того, щоб мати власне значення, вона просто посилається на вихідну змінну.

Передача параметра, що використовується, по посиланню вказується шляхом розміщення символу амперсанда (&) перед його ім'ям у визначенні функції. Ніякі зміни у виклику функції не потрібні.

Раніше наведений приклад функції increment() можна змінити, передавши параметр по посиланню, після чого функція буде працювати правильно.

Приклад 5.12. Приклад передачі даних по посиланню.

```
<?
function increment(&$value,$amount=1)
{
$value=$value+$amount;
}
$a=10;
echo $a;
increment($a);
echo"<br>";
echo $a;?>
```

Тепер ми маємо у своєму розпорядженні працюючу функцію й можемо назвати змінну, значення якої потрібно збільшувати як завгодно. Як уже згадувалося, використання того самого імені усередині й зовні функції може привести до плутанини, тому змінній в основному сценарії ми привласнимо нове ім'я. Тепер код буде виводити на екран значення 10 перед звертанням до функції increment() і 11 — після нього.

Ключове слово return припиняє виконання функції. Коли виконання функції завершується або тому, що всі оператори виконані, або через використання ключового слова return, керування повертається до оператора, наступного за викликом функції.

При виклику наступної функції виконується тільки перший оператор echo.

Приклад 5.13. Повернення функції.

```
<?
function test_return()
{
echo "This statement will be executed"; //Це повідомлення виведеться
return;
echo "This statement never be executed"; //Це повідомлення ніколи не виведеться
}
test_return();
echo"<br>";
?>
```

Очевидно, це не дуже корисний спосіб використання оператора return. Повернення з функції буде вимагатися тільки при виконанні певної умови.

Умова виникнення помилки — розповсюджена причина застосування оператора return з метою передчасного припинення виконання функції. Наприклад, була створена функція для визначення більшого із двох чисел, вихід з неї може знадобитися у випадку відсутності одного із чисел.

Приклад 5.14. Визначення більшого з двох чисел.

```
<?
function larger($x,$y)
```

```
{
global $d;
if(!isset($x)||!isset($y))
{echo "Не уведені значення";
return;}
if ($x>=$y)
echo $x;
else
echo $y;
}
$a=1;
$b=2.5;
$c=1.9;
larger($a,$b);
echo"<br>";
larger($c,$a);
echo"<br>";
larger($d,$c);
?>
```

Вбудована функція isset() повідомляє, чи була створена змінна й чи було їй привласнене значення. Цей код повинен генерувати повідомлення про помилку й виконувати повернення, якщо кожний з параметрів немає встановленого значення. Ця перевірка виконується за допомогою виразу !isset(), що означає "НЕ isset()".

5.8. Повернення значень із функцій

Повернення з функції — не єдина причина використання оператора return. У багатьох функціях оператори return використаються для обміну даними із кодом, що їх викликав. Функція була б більш корисною, якби

замість виведення на екран результату порівняння у функції larger() вона повертала б відповідь. У цьому випадку код, що її викликав, міг би приймати рішення, чи потрібно й коли саме відображати або використати відповідь. Еквівалентна вбудована функція max() діє саме так.

Функцію larger() можна переписати в такий спосіб:

Пример 5.15. Повернення значень із функцій.

```
<?
function larger($x,$y)
{
global $d;
if(!isset($x)||!isset($y))
return -1.5E+308;
else if ($x>=$y)
return $x;
else
return $y;
}
$a=10;
$b=2.5;
$c=19;
echo larger($a,$b);
echo"<br>";
echo larger($c,$b);
echo"<br>";
echo larger($d,$b);
?>
```

Ця функція повертає більше із двох переданих їй значень. У випадку помилки функція буде явно повертати інше число. Якщо одне із чисел від-

сутнє, можна нічого не повертати або повернути значення -1.5-E^{308}. Це число дуже мале і його не можна поплутати з реальною відповіддю. Вбудована функція max() нічого не повертає, якщо обидві змінні не встановлені, а якщо тільки одна змінна була встановлена, функція повертає згадане значення.

Часто функції, які виконують певну задачу, але не повинні повертати значення, повертають значення true або false для вказівки на успішне або неуспішне виконання. Значення true і false можуть бути відповідно представлені значенням 1 або 0.

Приклад 5.16. Напишемо програму Транслітерація.

```
<html><head> <title> транслітерація </title></head><body>
 <h1>транслітерація</h1>
 <form action="oll.php" method = "post" >
  <table border=0>
   <tr><td>Введіть слово</td><td><input type=text name="st"
maxlength=13 size=13><br></td></tr>
<tr><td colspan=2><input type=submit value="Register"></td></tr>
  </table> </form></body></html>
```

Файл oll.php

```
<?
$st=$_POST["st"];
$ts=encodestring($st);
echo $ts;
 // функція перекладу тексту с кириллиці в траскрипт
 function encodestring($st)
 {
  // Спочатку поміняємо "односимвольні" фонеми.
  $st=strtr($st,"абвгдеёзийклмнопрстуфхъыэ_",
```

```
"abvgdeeziyklmnoprstufh'iei");
$st=strtr($st,"АБВГДЕЁЗИЙКЛМНОПРСТУФХЪЫЭ_",
"ABVGDEEZIYKLMNOPRSTUFH'IEI");
// а потім - "багатосимвольні".
$st=strtr($st,
        array(
          "ж"=>"zh", "ц"=>"ts", "ч"=>"ch", "ш"=>"sh",
          "щ"=>"shch","ь"=>"", "ю"=>"yu", "я"=>"ya",
          "Ж"=>"ZH", "Ц"=>"TS", "Ч"=>"CH", "Ш"=>"SH",
          "Щ"=>"SHCH","Ь"=>"", "Ю"=>"YU", "Я"=>"YA",
          "ї"=>"i", "Ї"=>"Yi", "є"=>"ie", "Є"=>"Ye"
          )
    );
  // Повертаємо результат.
return $st; }?>
```

5.9. Блоки коду

Група операторів оголошується блоком шляхом оточення їх у фігурні дужки. Це не впливає на дію більшої частини коду, але робить специфічний вплив у тому числі й на спосіб виконання таких керуючих структур, як цикли й умовні оператори.

Наступні два приклади працюють різним чином.

Приклад 5.17.(без блоку коду)

```
<?
for($i=0;$i<3;$i++)
echo "line 1<br>";
echo "line 2<br>";
?>
```

Приклад 5.18. Використання блоків коду.

```
<?
for($i=0;$i<3;$i++)
{
echo "line 1<br>";
echo "line 2<br>";
}
?>
```

В обох прикладах цикл for повторюється три рази. У першому прикладі тільки єдиний рядок, розташований безпосередньо під цим кодом, виконується в циклі for.

У другому прикладі блок кола використається для групування двох рядків. Це означає, що обидві рядки виконуються в циклі for три рази.

5.10. Час життя змінної

Часом життя змінної називається інтервал виконання програми, протягом якого вона існує. Оскільки локальні змінні мають своєю областю видимості функцію, то час життя локальної змінної визначається часом виконання функції, у якій вона оголошена. Це означає, що в різних функціях зовсім незалежно одна від одної можуть використовуватися змінні з однаковими іменами. Локальна змінна при кожному виклику функції ініціалізується заново, тому функція-лічильник, у наведеному нижче прикладі завжди буде повертати значення 1:

```
function counter()
{
 $counter = 0;
 return ++$counter;
}
```

Для того, щоб локальна змінна зберігала своє попереднє значення при нових викликах функції, її можна оголосити статичної за допомогою ключового слова **static**:

```
function counter()
{
 static $counter = 0;
 return ++$counter;
}
```

Часом життя статичних змінних є час виконання сценарію. Тобто, якщо користувач перезавантажує сторінку, це приводить до нового виконання сценарію.

5.11. Що таке рекурсія?

Рекурсією називається така конструкція, при якій функція викликає саму себе. Розрізняють пряму й непряму рекурсії. Функція називається прямо рекурсивною, якщо містить у своєму тілі виклик самої себе. Якщо ж функція викликає іншу функцію, що у свою чергу викликає першу, то така функція називається непрямою рекурсією.

Розглянемо класичні приклади використання рекурсії - реалізацію операції піднесення до степеня й обчислення факторіала числа. Ці приклади є класичними та зручними для пояснення поняття рекурсії. Однак вони не дають виграшу в програмній реалізації в порівнянні з ітераційним способом рішення цих завдань.

Приклад 5.19. Приклад використання рекурсії.

```
<?
 function degree($x,$y)
 {
  if($y)
  {
```

```
    return $x*degree($x,$y-1);
   }
   return 1;
  }
  echo(degree(2,4)); // виводить 16
 ?>
```

Цей приклад заснований на тому, що x^y еквівалентно $x*x^{(y-1)}$. У цьому коді завдання обчислення 2^4 розбивається на обчислення $2*2^3$. Потім $2*2^3$ розбивається на $2*2^2$ і так доти, поки показник не стане рівним нулю.

Приклад 5.20. Ітераційний варіант прикладу.

```
<?
 function degree($x,$y)
 {
  for($result = 1; $y > 0; --$y)
  {
   $result *= $x;
  }
  return $result;
 }
 echo(degree(2,4)); // виводить 16
?>
```

Крім того, що цей код набагато легше зрозуміти, він ще й більше ефективний, оскільки прохід циклу обходиться "дешевше" виклику функції.

Приклад 5.21. Знаходження факторіалу.

```
<?
 function fact($x)
 {
```

```
 if ($x < 0) return 0;
 if ($x == 0) return 1;
 return $x * fact($x - 1);
}
echo (fact(3)); // виводить 6
?>
```

Для негативного аргументу функція повертає нульове значення, тому що факторіал негативного числа не існує по визначенню. Для нульового параметра функція повертає значення 1, оскільки 0! = 1. У деяких випадках викликається та ж функція зі зменшеним на 1 значенням параметра, після чого результат множиться на поточне значення параметра. Тобто, відбувається обчислення добутку:

$$k * (k - 1) * (k - 2) * ... * 3 * 2 * 1 * 1$$

Послідовність рекурсивних викликів переривається тільки при виклику **fact(0).**

Приклад 5.22. Обчислення факторіалу за допомогою ітерацій.

```
<?
 function fact($x)
 {
  for ($result = 1; $x > 1; --$x)
  {
   $result *= $x;
  }
  return $result;
 }
 echo (fact(6)); // виводить 520
?>
```

5.12. Функції з рядковими змінними

Побудуємо просту форму, що буде вводити зауваження й побажання клієнтів. Однак наш додаток буде володіти однією перевагою в порівнянні з багатьма іншими формами, які можна зустріти в Web. Замість того щоб відправляти форму по узагальненій адресі електронної пошти типу reedback@bobsdomain.com, ми постараємося додати процесу деяку інтелектуальність за рахунок пошуку в даних ключових слів, які дають можливість відправити повідомлення електронної пошти відповідному співробітникові компанії. Наприклад, якщо повідомлення електронної пошти містить слово "advertising" ("реклама"), воно може бути відправлено у відділ маркетингу. Якщо повідомлення електронної пошти надходить від важливого клієнта, воно може бути спрямоване безпосередньо керівникові.

Ми почнемо розгляд з простого сценарію, наведеного в лістінгу, змінюючи його по мірі вивчення матеріалу.

Приклад 5.23. Відгук відвідувачів.

```
<html><head><title> повідомлення</title></head><body>
<h1>Відгук відвідувачів</h1>
<p>Скажіть, що Ви думаєте про наші товари</p>
<form method=post action="market.php">
Ваше ім'я: <br>
<input type=text name="name" size=40><br>
Ваш e-mail address: <br>
```

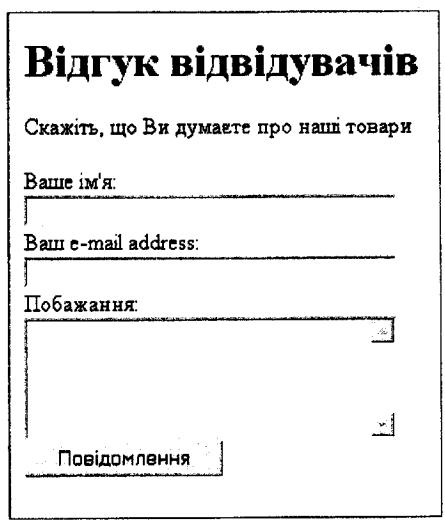


Рис.5.1 **Вікно прикладу 5.23**

```
<input type=text name="email" size=40><br>
Побажання: <br>
<textarea name="feedback" rows=5 cols=30>
</textarea><br>
<input type=submit value="Повідомлення">
</form>
</body>
```

Лістінг market.php — Основний сценарій для створення вмісту повідомлення електронної пошти

```
<html><head><title>повідомлення</title>
</head><body>
<h1>Відгук відвідувачів</h1>
<p>Скажіть, що Ви думаєте про наші товари</p>
<form method=post action="market.php">
Ваше ім'я: <br>
<input type=text name="name" size=40><br>
Ваш e-mail address: <br>
<input type=text name="email" size=40><br>
Побажання: <br>
<textarea name="feedback" rows=5 cols=30>
</textarea><br>
<input type=submit value="Повідомлення">
</form></body>
```

В загальному випадку необхідно перевірити заповнення користувачами всіх обов'язкових полів форми, скориставшись, наприклад, функцію isempty().

Як видно із цього сценарію, поля форми об'єднані, а PHP-функція mail застосовується для відправлення по електронній пошті за адресою

feedback@bobsdomain.com дотепер функція mail() ще не використалася, тому давайте розглянемо, як вона працює. Зовсім очевидно, що ця функція відправляє повідомлення електронної пошти. Її прототип виглядає в такий спосіб:

bool mail (string *кому*, string *тема*, string повідомлення,
string [додаткові_заголовки])

Перших три параметри обов'язкові й представляють, відповідно, адресу, по якій повинне бути відправлене повідомлення, рядок теми й зміст повідомлення. Четвертий параметр може використовуватися для відправлення будь-яких додаткових допустимих заголовків повідомлення електронної пошти. Ці заголовки описані в документі RFC822, що доступний в Internet. (RFC, або Requests For Comment (Запити на коментарі), - джерело багатьох стандартів Internet.) У даному прикладі четвертий параметр був використаний для включення в повідомлення адреси "From:" ("Від:"). Його можна також застосовувати для додавання таких полів, як "Reply-Te:" ("Відповісти:") і "Cc:". Якщо потрібно ввести більше одного додаткового заголовка, їх потрібно просто розділити символами нового рядка (\n), як показано в наступному прикладі:

$additional_headers="From:webserver@bobsdomain.com\n"."Reply -To: bob@bobsdomain.com";

Щоб можна було скористатися функцією email(), в установці PHP варто вказати програму електронної пошти, яка застосовується.

5.13. Форматування рядків

Часто рядки, що вводять користувачі (як правило, з інтерфейсу HTML-форми), треба упорядковувати, перш ніж їх можна буде використати.

Усікання рядків: *функції chop(), Itrim() і trim()*

Перший крок по приведенню рядків у порядок - видалення з них будь-яких зайвих пропусків. Хоча це й не є обов'язковим, але може виявитися корисним, якщо передбачається збереження рядка у файлі або базі даних або його порівняння з іншими рядками. Для цієї мети в PHP існують три корисних функції. Ми використаємо функцію trim().

$name=trim($name); $email=trim($email) ; $feedback=trim($feedback) ;

Функція trim() видаляє пробіли на початку й кінці рядка й повертає відформатований рядок. При цьому символами пробілів уважаються символи нового рядка (\n), повернення каретки (\r), символи горизонтальної (\t) і вертикальної табуляції (\v), кінця рядка (\0) і звичайні пробіли.

Залежно від конкретної мети замість цієї функції можна використати функції Itrim() або chop(). Обидві ці функції аналогічні функції trim. Вони приймають рядок, як параметр і повертають відформатований рядок. Розходження між ними полягає в тому, що функція trim() видаляє пробіли на початку й кінці рядка, Itrim() видаляє пробіли тільки на початку; рядка (або лівої її частини), а chop() — тільки наприкінці (або правій частини) рядка.

5.14. Використання функцій explode(), implode() і join()

Щоб будь-який відгук надходив безпосередньо до дирекції Магазину, ми розділимо адресу пошти на частині, для з'ясування, чи не відповідають вони важливому клієнтові.

Перша функція, explode(), яку можна було б задіяти для цього має наступний прототип:

array explode(string separator, string input).

Для одержання імені домену з адреси електронної пошти в сценарії можна використати наступний код:

$email_array = explode("@", $email).

У результаті цього виклику функції explode() адреса електронної пошти ділиться на дві частини: ім'я користувача, що зберігається в $email_array[0], і ім'я домену, що зберігається в $email_array[l]. Тепер можна перевірити ім'я домену для визначення джерела повідомлення клієнта й для його переадресації відповідній особі. Файл market.php.

Приклад 5.24. Електронний лист.

```
<html>
<head><title>Untitled</title></head>
<body>
<?
$toaddress="feedback@domain.com";
$subject="Повідомлення с Web-сайту";
$mailcontent="customer name: ".$name."\n"
. "Customer email:" . $email. "\n"
."Customer comments: \n".$feedback."\n";
$fromaddress = "webserver@domain.com" ;
$email_array = explode("@", $email);
if ($email_array[1]=="bigcustomer.com") $toaddress = "director@domain.com";
else
$toaddress = "feedback@domain,com";
$token =strtok ($feedback,"");
echo $token."<br>";
while ($token!=""){
$token=strtok("");
echo $token."<br>"; };
$toaddress = "feedback@domain.com"; // значення по замовчуванню
// Змінювання змінної $toaddreas при відповідності Критериям
```

```
if (strstr($feedback, "менеджер"))
$toaddress = "m@domain.com";
else if (strstr($feedback, "маркетинг"))
$toaddress = "market@domain.com" ;
else if (strstr($feedback, "директор") )
$toaddress = "director@domain.com" ;
mail($toaddress,$subject,$mailcontent,$fromaddress);
echo nl2br($mailcontent);
?>
<html> <head>
<title>повідомлення</title> </head> <body>
<hl>Ваше повідомлення відправлено</hl>
</body></html>
```

Зверніть увагу, що якщо ім'я домену містить прописні букви, цей код не буде працювати. Цієї проблеми можна було б уникнути, перетворивши всі букви імені домену в прописні або рядкові, а потім виконавши перевірку

```
$email__array [1] = strtoupper($email_array [ 1] ) ;
```

Ефекту, протилежного дії функції explode(), можна домогтися використовуючи функції implode() або join(), які ідентичні. Наприклад:

```
$new_email = implode("@", $email_array);
```

Ця функція приймає елементи з масиву $emal_array і об'єднує їх в рядок. Виклик цієї функції багато в чому подібний до виклику функції explode(), але її дія протилежна.

5.15. Використання функції strtok()

На відміну від функції explode(), що розділяє на частини відразу весь рядок, функція strlok() дає можливість одержати фрагменти (лексемами) з

рядка по одному. Ця функція - корисна альтернатива використанню функції explode(), при обробці слів з рядка по одному.

Прототип функції strtok() має такий вигляд:

srting strtok(string input, string *separator).*

Як роздільник *separator* можна використати символ або рядок символів, але варто мати на увазі, що рядок уведення буде розділятися по кожному із символів роздільника, а не по всьому рядку роздільника (як має місце у функції explode). Для одержання наступних лексем з рядка досить передати функції єдиний параметр - роздільник. Функція зберігає власний внутрішній покажчик на позицію в рядку. Якщо потрібно переустановити покажчик, можна знову передати рядок у функцію.

Як правило, функція strtok() використається в такий спосіб:

```
$token =strtok ($feedback,"");
echo    $token."<br>";
while    ($token!=""){
$token  =  strtok("");
echo   $token."<br>"; };
```

Звичайно, має сенс перевірити, наприклад, за допомогою функції emty(), чи дійсно користувач заповнив які-небудь поля у формі.

Наведений фрагмент коду виводить кожну лексему відгуку клієнта в окремому рядку й виконує цикли аж до вичерпання лексем. Зверніть увагу, що в PHP- функція strtok() працює не зовсім так, як у С. При наявності у вихідному рядку двох розташованих *підряд* роздільників (у даному прикладі — двох наступних підряд пробілів) функція strtok() повертає порожній рядок. Його не можна відрізнити від порожнього рядка, що повертається по досягненні кінця вихідного рядка. Крім того, якщо однієї з лексем є 0, функція також поверне порожній рядок. У зв'язку з цим PHP- функція

strtok() менш корисна, ніж у С. Часто краще просто застосовувати функцію explode().

5.16. Використання функції substr()

Функція substr() дозволяє одержати доступ до підрядка між заданими початковою й кінцевою позиціями в рядку. Для нашого приклада вона не підходить, але може виявитися корисною, якщо потрібно розділити на частини фіксовані відформатовані рядки. Функція substr() має наступний прототип:

string substr(string рядок, int *початок*, int [довжина]).

Ця функція повертає підрядок з рядка.

Давайте розглянемо приклад використання функції для наступного тестового рядка:

$test = "Як тебе не любити, Києве мій";.

Якщо викликати функцію з позитивним числом як параметр *початок*, ми одержимо рядок, починаючи з позиції *початок* і до кінця рядка. Наприклад: echo substr($test, 3); повертає рядок " тебе не любити, Києве мій". Зверніть увагу, що нумерація позицій рядка починається з 0, як у масивах.

Якщо викликати функцію substr() тільки з негативним параметром *початок,* ми одержимо рядок, що складається із символів наприкінці рядка, кількість яких визначена параметром *початок.* Наприклад, echo substr($test,-9);повертає підрядок "Києве мій ".

Параметр *довжина* може використовуватися для вказівки або для кількості символів, які повинні бути повернуті (якщо він позитивний), або останнього символу, що повертає послідовності (якщо він негативний). Наприклад,

echo substr($test,0,5);

повертає перші чотири символи рядка, а саме "Як тебе".

Наступний код: echo substr($test,3,-10); повертає символи, розташовані між четвертим символом від початку й тринадцятим символи від кінця рядка, тобто "тебе не любити".

5.17. Порівняння рядків

Дотепер ми використали тільки операцію == для порівняння двох рядків. У PHP можна виконувати деякі більш складні операції порівняння. Ми розділили ці операції на дві категорії: частковий збіг і всі інші. На початку ми розглянемо інші функції, а потім функції встановлення часткового збігу, які будуть потрібні для подальшої розробки прикладу інтелектуальної форми Відгук відвідувачів.

5.18. Упорядкування рядків: функції strcmp(), strcasecmp() і strnatcmp()

Ці функції можуть використовуватися для впорядкування рядків, що корисно при сортуванні даних. Прототип функції strcmp() має вигляд

int strcmp(string str1, string str2).

Функція приймає два рядки, які й буде порівнювати. Якщо вони рівні функція поверне значення 0. Якщо в лексикографічному порядку рядок str1 розташований за рядком str2 (або більше нього), функція strcmp() поверне позитивне число. Якщо рядок str1 менше рядка str2, функція strcmp() поверне негативне число. Ця функція є чутливою до регістра.

Функція strcasecmp() ідентична попередній за винятком того, що не чутлива до регістра.

Функція strnatcmp() і її нечутливий до регістра аналог srtnatcasecmp(). Ці функції порівнюють рядки відповідно до "природного впорядкування", що більш звично для людини. Наприклад, strcmp() розташувала б рядок "2" після рядка "12", оскільки лексикографічно вона більше. Функція strnatcm() розташувала б ці рядки у зворотному порядку.

5.19. Перевірка довжини рядка за допомогою функції strlen()

Довжину рядка можна перевірити за допомогою функції slrlen(). Якщо передати цій функції рядок, вона поверне її довжину. Наприклад, strlen("hello") повертає 5.

Це можна задіяти при перевірці правильності даних, що вводять, розглянемо адресу електронної пошти, що зберігається в змінній $email. Один з основних способів перевірки правильності адреси електронної пошти, яка зберігається в змінній email - перевірка її довжини. Мінімальна довжина адреси електронної пошти дорівнює шести символам – наприклад, адреса містить код країни без якого-небудь домену другого рівня, однобуквене ім'я домену й однобуквену адресу електронної пошти, вона може мати вигляд a@a.to. Отже, програма могла б генерувати повідомлення про помилку, якщо довжина адреси виявляється меншою:

```
(strlen($email) < 6){
echo "Адреса не вірна";
exit; // завершення виконання сценарію PHP}
```

Звичайно, це дуже спрощений спосіб перевірки правильності інформації.

Зіставлення й заміна підрядків за допомогою строкових функцій

Часто потрібно перевірити наявність конкретного підрядка в більш довгому рядку. Звичайно, це часткове зіставлення більш корисно, ніж перевірка рядків на предмет рівності.

У нашому прикладі, у файлі market.php потрібно виконати пошук на предмет наявності ключових фраз у відгуку користувача й відправити повідомлення електронної пошти у відповідний підрозділ. Якщо повідомлення електронної пошти, у яких мова йде про магазини, потрібно направляти менеджерові роздрібного продажу, потрібно знати, чи є присутнім в повідомленні слово "Магазин" (або його похідні). Маючи у своєму розпоря-

дженні вже розглянуті функції, можна було б використати функції explode() або slrtok() для одержання окремих слів повідомлення, а потім порівняти використовуючи операцію == або функцію strcmp().

Однак, це ж можна зробити за допомогою єдиного виклику однієї з функцій зіставлення рядків або регулярних виразів. Ці функції використовуються для пошуку певної послідовності усередині рядка. Ми розглянемо кожний з наборів функцій.

5.20. Пошук рядків у рядках: функції strstr(), strchr(), strrchr(), stristr()

Для пошуку рядка усередині іншого рядка можна використати кожну з функцій strchr(), strrchr() або stristr().

Функція strstr() є найбільш загальною й може використовуватися для пошуку відповідного рядка або символу усередині більше довгого рядка. Слід зазначити, що в PHP функція strchr() повністю збігається з функцією strstr(), хоча її ім'я припускає, що вона застосовується для пошуку символу в рядку. Функція strslr() має наступний прототип:

string strstr(string *haystack,* string needle).

Як параметр функції передається рядок *haystack,* у якому потрібно виконати пошук, і рядок *needle,* що потрібно знайти. У випадку виявлення повної відповідності з рядком *needle* функція повертає частину рядка *haystack, що* починається з рядка *needle;* у протилежному випадку вона повертає значення false.

Наприклад, у файлі market.php рішення про адресацію повідомлень електронної пошти здійснюється в такий спосіб:

```
$toaddress = "feedback@domain.com;
 if (strstr($feedback, "менеджер"))
$toaddress = "m@domain.nim";
else if (strstr($feedback, "маркетинг"))
```

```
$toaddress = "market@domain.com" ;
 else if (strstr(Sfeedback, "директор") )
$toaddress = "director@domain.com" ;
```

Цей код виконує перевірку відгуку на предмет наявності певних слів і відправляє повідомлення електронної пошти відповідній особі. Наприклад, відгук клієнта звучить як "менеджерові компанії", у ньому буде знайдений рядок "менеджер", і відгук буде відправлено за адресою m@domain.com.

Функція strstr() використається для пошуку підрядка або символу усередині іншого рядка. Існує два варіанти функції strstr(). Перший - функція slristr(), що практично ідентична попередній, але не чутлива до регістра. Це зручно в даному додатку, оскільки клієнт міг би ввести "delivery", "Delivery" або "DELIVERY".

Другий варіант — функція strrchr(), що також майже ідентична strstr(), але буде повертати частину рядка *haystack,* починаючи з останнього входження рядка *needle.*

5.21. Визначення позиції підрядка: функції strpos(), strrpos()

Функції strpos() і strrpos() діють аналогічно функції strstr() за винятком того, що замість підрядка вони повертають числову позицію рядка усередині рядка *Haystack.* Функція strpos() має наступний прототип:

string strpos(string haystack, string, int [ofsset]).

Функція повертає цілочислене значення, що, представляє *перше* входження рядка *needle* усередині рядка *haystack.* Як правило, перший символ розміщається в нульовій позиції. Наприклад, код поверне значення 4 у вікні браузера:

```
$test="Hallo world";
echo strpos($test,"o");
```

У цьому випадку як рядок *needle* був переданий єдиний символ, але може бути рядок будь-якої довжини.

Необов'язковий параметр offset використовується для вказування позиції усередині *haystack,* з якої повинен починатися пошук. Наприклад,

```
echo  strpos($test,"про",5).
```

Цей код повторить значення 5 у вікні браузера, оскільки PHP починає шукати символ з 5 позиції і не бачити цей символ у позиції 4.

Функція strrpos() діє майже так само, але буде повертати позицію останнього входження рядка *needle* у рядку *haystack.* На відміну від функції strpos функція працює тільки з односимвольним рядком *needle.* Тому, якщо дати рядок як *needle,* для зіставлення функція буде використовуватися тільки перший символ рядка.

Для запобігання цієї проблеми можна використати операцію == з метою перевірки можливих значень:

```
$result=strpos($test, "H") ;
if ($result===false)
echo "Not found"
else
echo  "Found at position 0";
```

Заміна підрядків, функції: *str_replace(), substr_replace()*

Функціональні можливості пошуку й заміни можуть виявитися винятково корисними й при роботі з рядками. Раніше ми вже використали пошук і заміну для персоніфікації документів згенерованих PHP, наприклад, заміняючи *«name»* конкретної особи, а *«address»* — його адресою. Ці функції можна використати також для цензури певних термінів, наприклад, у додатку дискусійного форуму або навіть у додатку "Відгук відвідувачів".

Для виконання цієї задачі можна знову скористатися рядковими функціями обробки регулярних виразів. Рядкова функція, що найчастіше використовується для заміни — str_replace(), має наступний прототип:

string substr_replace(string needle, string replacement, int start,int [length]);

Ця функція буде заміщати частину рядка string рядком *replacement.* Яку саме - залежить від значень параметра *start* і необов'язкового параметра *length.* Значення *start* представляє зсув позиції в рядку, з якого повинна починатися заміна. Якщо воно дорівнює 0 або позитивне, то визначає зсув від початку рядка, якщо негативне - зсув від кінця рядка. Наприклад, рядок що випливає, буде заміняти останній символ у змінної $test символом "X":

$test=substr_replace($test,"X",-1) .

Параметр *length* необов'язковий і представляє позицію, у якій заміна повинна бути припинена. Якщо це значення не зазначене, заміна буде виконуватися від позиції певної параметром *start,* і до кінця рядка.

Запитання для самоконтролю

1. При яких умовах доцільно використання функції?
2. Базова структура функції?
3. Поясніть область дії змінної?
4. Що таке рекурсія?
5. Які рядкові функції Ви знаєте?
6. Якою функцією здійснюється блокування файлу?

Задачі для розв'язування

1. Ввести в текстове поле Прізвище Ім'я по-Батькові. За допомогою рядкових змінних виокремити ім'я та записати його на іншій сторінці.

2. Обчислити

$$R=\frac{\log_2 x-\log_b y}{2\log_{(b+2)}(x+y)}$$

3. Використовуючи функції знайти найбільші елементи і їхні порядкові номери масивів X(P), A(T).

4. Використовуючи функції, знайти найменші елементи і номери рядків і стовпчиків, в яких вони розміщені, для матриць A(5,8) і B(3,5).

5. Обчислити

$$H=(\sum_{i=1}^{10}\sin x_i+\sum_{i=1}^{10}\cos x_i)/\sum_{i=1}^{10}\cos x_i,$$

де змінна x задана масивом. Суми обчислювати в функції.

6. Використовуючи функції, обчислити і запам'ятати кількість від'ємних елементів кожного стовпчика для матриць A(5,5), B(3,6).

7. Вказати які сполучення із двох літер зустрічаються в заданому тексті і скільки разів кожне з них.

8. Відредагувати речення, вилучаючи з нього зайві пробіли, і залишаючи тільки по одному пробілу між словами.

9. В заданому реченні вказати слово, в якому кількість голосних (A,I,E,O) максимальна.

10. Для кожного символу заданого тексту вказати, скільки разів він зустрічається в тексті. Повідомлення по кожному символу повинно друкуватися не більше одного разу.

РОЗДІЛ 6. РЕГУЛЯРНІ ВИРАЗИ

Регулярні вирази – міні-мова описів для пошуку в рядках інформації із заданого шаблона. За допомогою регулярних виразів можна знайти в рядку підрядок, що задовольняє заданому шаблону, перевірити чи існує заданий рядок.

Алгоритм пошуку з використанням регулярних виразів був уперше розроблений одним із творців UNIX Кеном Томпсоном. Цікаво, що споконвічно регулярні вирази з'явилися не в теорії обчислювальних систем, а в нейрофізіології. Основу теорії регулярних виразів заклали нейрофізіологи У. Мак-Каллох й У. Піттс, що працювали над способами математичного опису нервових процесів. Пізніше математик С. Кліні, ґрунтуючись на цих дослідженнях, опублікував роботу "Подання подій у нейронних мережах", у якій і було уведене поняття регулярних виразів. Кен Томпсон, ґрунтуючись на цих роботах, адаптував теорію регулярних виразів для алгоритмів пошуку інформації. Саме починаючи з його робіт, регулярні вирази стали використовуватися в текстових редакторах і ввійшли в більшість мов програмування.

6.1. Базовий синтаксис і створення регулярних виразів

Найпростіший регулярний вираз можна записати так: "abc"
Цей вираз відповідає будь-якому рядку, що містить підрядок "abc".
Існує таке поняття, як **вирази у квадратних дужках**. Квадратні дужки обмежують пошук тими символами, які в них вкладені:

"[abc]"

Цьому регулярному виразу відповідає будь-який рядок, що містить abc або разом, або кожну букву окремо.

Допустимо, нам потрібно створити регулярний вираз, що відповідає всім буквам українського алфавіту. У цьому випадку ми можемо, звичай-

но, перелічити всі ці букви в регулярному виразі. Це припустимо, але не раціонально. Більш коротко такий регулярний вирази можна записати в такий спосіб:

"[а-Я]"

Це вираз відповідає всім буквам українського алфавіту, оскільки будь-які два символи, між якими стоїть дефіс, задають відповідність діапазону символів, що перебувають між ними. Зауважте, що регулярне вирази "[а-Я]" описує символи як нижнього, так і верхнього регістрів, тому більш докладно цей вираз можна записати так:

"[а-яА-Я]"

Точно в такий же спосіб задаються регулярні вирази, що відповідають числам:

"[0-9]"

або

"[0123456789]"

Обидва вирази еквівалентні й відповідають будь-якій цифрі.

6.2. Груповий символ

При створенні регулярних виразів часто зручно користуватися груповим символом крапки ".", що поєднує два одиночних символи, за винятком символу “\n”. Наприклад:

.ок

Цей вираз, зокрема відповідає рядкам "кок", "док", "сок".

Вираз

"х.[0-9]"

відповідає рядку, що містить символ х, за яким іде будь-який інший символ і цифри від 0 до 9. Цьому критерію, наприклад, задовольняють рядка "ху1", "хz2".

Гілки

У регулярному виразі може бути кілька гілок, які розділяються символом |, що діє як оператор OR (АБО). Тобто, якщо у виразі використовуються гілки, то для відповідності регулярного виразу якому-небудь рядку, досить, щоб тільки одна з гілок відповідала цьому рядку:

"abc|абв".

Цьому регулярному виразу відповідає будь-який рядок, що містить підрядок "abc" або "абв". Розгалуження зручно застосовувати при перевірці розширень й імен файлів, зон доменних імен. Наприклад, якщо регулярний вирази перевіряє, чи існують в рядку підрядки "ru", "com" або "net":

"ru|com|net".

Для виключення послідовності символів з пошуку перед нею ставиться символ "^":

"[^а-я]"

Цей регулярний вираз відповідає будь-якому символу, що не знаходиться в діапазоні а-я. Зверніть увагу, що символ ^ перебуває усередині квадратних дужок, тому що тільки в цьому випадку він має значення "не". При використанні символу ^ поза квадратними дужками, він має зовсім інше значення.

6.3. Кваліфікатори

Регулярний вираз можна уточнити за допомогою **кваліфікаторів** - так називаються символи +, ?, *. Кваліфікатори говорять про те, скільки разів послідовність символів може зустрітися в рядку й вказуються безпосередньо після тієї частини виразу, до якої вони застосовуються:

- "а+" - хоча б один а (рядка "абв" й "абва" відповідають цьому виразу, а рядок "укр" - ні);
- "а?" - нуль або один а (рядка "абв" й "укр" відповідають цьому виразу, а рядок "абва" - ні);

- "a*" - нуль або більше а (рядка "абв" й "абва" й "укр" відповідають цьому виразу).

6.4. Границі

Границі - це числа у фігурних дужках, що вказують кількість входжень у рядок фрагмента виразу, що стоїть перед границею:

- "ху{2}" відповідає рядку, у якому за х стоять два у;
- "ху{2,}" відповідає рядку, у якому за х стоїть не менш двох у (може бути й більше);
- "ху{2,6}" відповідає рядку, у якому за х стоїть від двох до шести у;

Для вказівки кількості входжень не одного символу, а їхньої послідовності, використаються круглі дужки:

- "х(уz){2,6}" відповідає рядку, у якому за х слідує від двох до шести послідовностей уz;
- "х(уz)*" відповідає рядку, у якому за х слідує нуль і більше послідовностей уz.

6.5. Підвирази

Іноді буває зручно створювати регулярні вирази таким чином, щоб можна було, наприклад, сказати, що за одним з рядків "морська", слідує рядок "хвиля". Для цього регулярний вираз розбивають на **підвирази** за допомогою круглих дужок:

(морська)*хвиля.

Цей вирази відповідає рядкам "хвиля", "морська хвиля", "морська морська хвиля".

У регулярному виразі можна вказати, де конкретно цей підвираз повинен зустрічатися: на початку, наприкінці рядка або й на початку й наприкінці рядка.

Символ ^ відповідає початку рядка:"^ху". Такий вирази відповідає будь-якому рядку, що починається з ху. Зверніть увагу, що в цьому випадку символ ^ ставиться за межами виразу в дужках, наприклад:"^[a-z]".

Знак долара $ відповідає кінцю рядка:"ху$". Цей регулярний вираз відповідає будь-якому рядку, що закінчується на ху.

У тих випадках, коли потрібно порівнювати вирази в рядку, у якому зустрічаються спецсимволи, такі як $, ^, { , перед ними ставиться символ зворотної косої риски (\). Наприклад, для того, щоб знайти в рядку символ $, у регулярному виразі потрібно написати "\$".

У тих випадках, коли перед символом стоїть зворотна коса риска, говорять, що символ записаний у вигляді ***escape-послідовності.***

Те ж саме ставиться й до самого символу зворотної косої риски. Якщо потрібно провести зіставлення із символом зворотної косої риски, то в цьому випадку ставиться дві зворотних косих риски, тобто \\.

6.6. Класи символів

Класами символів називаються скорочені позначення для визначених символів.

- Клас [[:alnum:]] - буквено-цифрові символи .
- Клас [[:digit:]] - десяткові цифрові символи.
- Клас [[:xdigit:]] - шістнадцяткові цифрові символи.
- Клас [[:alpha:]] - буквені символи.
- Клас [[:upper:]] - прописні буквені символи.
- Клас [[:lower:]] - рядкові буквені символи.
- Клас [[:punct:]] - знаки пунктуації.
- Клас [[:space:]] - символи пробілу.
- Клас [[:blanc:]] - символи табуляції й пробілу.
- Клас [[:print:]] - друковані символи.
- Клас [[:cntrl:]] - керуючі символи.

- Клас [[:graph:]] - друковані символи.

Ви можете використовувати класи символів у регулярних виразах як інші символи. Наприклад:

- Еквівалентом виразу "[a-zA-Z_0-9]" є вираз "[[:alnum:]]".
- Вираз "[0-9]" еквівалентний виразу "[[:digit:]]".
- Вираз "[a-Z]" еквівалентний регулярному виразу "[[:alpha:]]".

6.7. Функції для роботи з регулярними виразами

PHP підтримує два види запису регулярних виразів: POSIX й Perl. POSIX розшифровується як Portable Operating System Interface (інтерфейс переносної операційної системи) і є стандартом для інтерфейсів додатків. *У загальному випадку, функції для роботи з регулярними виразами виконуються більш повільно, ніж строкові функції, що надають аналогічні можливості. Тому більш ефективно використовувати строкові функції.*

Функція ereg()

bool ereg(string pattern, string string [, array regs]).

Дана функція шукає в рядку **string** відповідність регулярному виразу, заданому в шаблоні **pattern**. Якщо відповідності підвиразу із шаблоном будуть знайдені, то вони зберігаються в масиві відповідностей **regs**. При цьому **$regs**[0] містить копію рядка string, **$regs**[1] містить підрядок, що починається з першої лівої дужки, **$regs**[2] зберігає підрядок, що починається із другої лівої дужки.

Нижче наведений код, що перетворить дату з формату YYYY-MM-DD у формат DD.MM.YYYY.

Приклад 6.1. Приклад використання функції ereg().

```
<?
 $date = "2003-03-21";
if (ereg ("([0-9]{4})-([0-9]{1,2})-([0-9]{1,2})", $date, $regs))
{
```

```
echo "$regs[3].$regs[2].$regs[1]";
}
else
{
echo "Невірний формат дати: $date";
}
?>
```

Функція ereg_replace()

string ereg_replace(string pattern, string replacement, string string).

Ця функція заміняє знайдений у рядку **string** шаблон **pattern** на рядок **replacement** й, якщо відповідність була знайдена, повертає модифікований рядок.

Однією із частих помилок при заміні числових значень є представлення числа як типу, відмінного від рядкового, що приводить до невірного результату. Синтаксис функції такий, що число обов'язково повинне бути записане як рядок.

Приклад 6.2. Приклад використання функції ereg_replace().

```
<?
 $number = "1952";
 $str = "Він народився в п'ятдесят другому.";
 echo("до заміни:$str");
 $str = ereg_replace("п'ятдесят другому", $number, $str);
 echo("<br> після заміни: $str");
?>
```

Результат:

до заміни: Він народився в п'ятдесят другому.

після заміни: Він народився в 1952.

Функція eregi()

bool eregi (string pattern, string string[, array regs]).

Ця функція ідентична функції **ereg**, за винятком того, що вона ігнорує регістр.

Функція eregi_replace()

string eregi_replace (string pattern, string replacement, string string).

Функція аналогічна функції **ereg_replace**, за винятком того, що вона є нечутливою до регістра.

Функція split()

array split (string pattern, string string [, int limit]).

Ця функція повертає масив рядків, які являють собою підрядки рядка **string**, утворені в результаті поділу рядка string на підрядки відповідно до регулярного виразу **pattern**. Якщо зазначено необов'язковий параметр **limit**, то в масиві, що повертається буде не більше limit елементів, останній з яких містить нерозділену частину рядка.

Ця функція корисна при поділі дат, доменних імен і т.д.

Приклад 6.3. Використання функції split().

```
<?
 $url = "www.softtime.ru";
 $array = split ("\.", $url);
 foreach($array as $index => $val)
 {
 echo("$index -> $val <br />");
 }
?>
```

Результат:

```
0 -> www
1 -> softtime
```

2 -> ru

Те ж саме можна проробити з датою:

Приклад 6.4. Використання функції split() з датою.

```
<?
  $date = "10-12-2003";
  $array = split ("-", $date);
  foreach($array as $index => $val)
  {
    echo("$index -> $val <br />");
  }
?>
```

Результат:

0 -> 10

1 -> 12

2 -> 2003

Функція spliti()

array spliti (string pattern, string string [, int limit]).

Ця функція аналогічна функції **split**, за винятком того, що є нечутливою до регістра.

Запитання для самоконтролю

1. Що таке регулярні вирази?
2. Що таке кваліфікатор?
3. Які Ви знаєте функції для роботи з регулярними виразами?

РОЗДІЛ 7. РОБОТА З ФАЙЛАМИ ТА КАТАЛОГАМИ

7.1 Запис даних у файл та зчитування файла

Запис даних у файл реалізується в три кроки:

1. Відкриття файла. Якщо файл ще не існує, його буде потрібно створити.
2. Запис даних у файл.
3. Закриття файла.

Аналогічно, зчитування даних з файлу також пов'язане з виконанням трьох кроків:

1. Відкриття файла. Якщо файл не може бути відкритий (наприклад, він не існує), ця ситуація повинна бути розпізнана й варто передбачити коректний вихід з неї.
2. Зчитування даних з файла.
3. Закриття файла.

При необхідності зчитування даних з файлу можна вибирати, яка частина файлу повинна зчитуватися за один раз. Пізніше будуть докладно розглядатися всі доступні можливості.

7.1.1. Відкриття файла

Для відкриття файла в середовищі PHP використається функція fopen(). При відкритті файла необхідно вказати, як його передбачається використати. Це називається режимом файла.

Серверна операційна система повинна знати, що потрібно робити з відкритим файлом. Їй потрібно знати, чи може файл бути відкритий і оброблений іншим сценарієм у той час, коли він є відкритим, якщо власник сценарію має право на подібне його використання. Власне кажучи, режими файлу надають операційній системі механізм для визначення способу обробки запитів на доступ, що надходять від інших користувачів або сценарі-

їв, а також метод перевірки наявності доступу й прав для роботи з конкретним файлом.

Таблиця 7.1. **Режими відкриття файла**

Режим	Значення
r	Режим читання
w	Режим запису
a	Режим додавання
b	Режим двійковий

При відкритті файлу варто прийняти три рішення:

1. Файл можна відкрити тільки для читання, тільки для запису або для читання й запису.
2. При виконанні запису у файл можна перезаписати будь-який існуючий вміст файлу або ж дописати нові дані в кінець файлу.
3. При спробі виконання запису у файл у системі, що розрізняє двійкові й текстові файли, може знадобитися вказати тип файлу.

Функція fopen() підтримує будь-які комбінації цих трьох варіантів.

7.1.2. Використання функції fopen() для відкриття файлу

Приклад 7.1. Запишемо "Hello, kibernetik!"в файл file.txt.

```
<?
 $file = fopen ("file.txt","w");
 $str = "Hello, kibernetik!";
 if ( !$file )
 {
  echo("Помилка відкриття файлу ");
 }
 else
 {
```

```
    fputs ( $file, $str);
  }
  fclose ($file);
?>
```

Давайте припустимо, що потрібно записати замовлення клієнта у файл замовлень Магазину, цей файл можна відкрити для запису за допомогою наступного оператора:

```
$fp=fopen("$D0CUMENT_ROOT/../orders/otders.txt","w");
```

Запис у файл у PHP виконується порівняно просто. Для цього можна скористатися однією з функцій **fwrite()** (file write — запис у файл) або **fputs()** (file put string — запис рядка у файл); **fputs()** — це псевдонім функції **fwrite().** Функцію **fwrite()** можна викликати в такий спосіб:

fwrite($fp,$outputstring).

Це вказує PHP на необхідність запису рядка зі змінної **$outputstring** у файл, зазначений $fp. Розглянемо функцію **fwrite()** більш докладно, перш ніж приступати до дослідження змінної **$outputstring.**

У дійсності функція **fwrite()** приймає три параметри, однак третій з них не є обов'язковим. Прототип функції **fwrite()** має такий вигляд

int fputs (int fp, string str, int [length]).

Третій параметр *length* являє собою максимальну кількість байтів, які потрібно записати. При передачі цього параметра функція **fwrite()** буде записувати рядок ***str*** у файл, зазначений параметром fp, поки не буде досягнутий кінець рядка або не запише ***length*** байтів, залежно від того, що відбудеться раніше.

7.1.3. Формати файлів

При створенні файлу даних, формат зберігання даних повністю залежить від програміста. Давайте створимо рядок, що представляє один запис у файлі даних. Це можна зробити в такий спосіб:

```
$outputstring = $date."\t".$cukor." цукор \t".$maclo." масло\t"
 .$kuri." кури\t\$".$Tovar ."\t". "\n";
@ $fp = fopen("orders.txt", "a");
```

У цьому простому прикладі кожний запис замовлення зберігається в окремому рядку. Подібне рішення обумовлене тим, що дозволяє як простий роздільник рядків використати символ нового рядка. Оскільки символи нового рядка невидимі, вони представляються за допомогою "\n".

```
<form action="processorder1.php" method=post>
<table border=0>
<tr bgcolor=#66ff33>
<td width=170>Товари</td>
<td width=17>Вартість</td>
</tr><tr>
<td>Цукор</td>
<td align=center><input type="text" name="cukor" size=3 max-
length=3></td>
</tr><tr>
<td>масло</td>
 <td align=center><input type="text" name="maclo" size=3
maxlength=3> </td></tr><tr>
<td>Кури</td>
<td align=center><input type="text" name="kuri" size=3 maxlength=3>
</td></tr><tr>
<td colspan=2 align=center><input type=submit value="Підрахунок">
</td></tr>
</table></form>
```

Processorder1.php

```
<html><head><title>Магазин</title></head>
```

```
<body>
<h2>Магазин</h2>
<?
$cukor =$_POST["cukor"];
$maclo =$_POST["maclo"];
$kuri =$_POST["kuri"];
$suma = 0;
$suma  += $cukor;
$suma += $maclo;
$suma += $kuri;
$Tovar = 0.00;
define("CinaCuk",7);
define("CinaMas",4);
define("CinaKur",18);
$date = date("H:i, jS F");
echo "<p>Ваше замовлення";
echo $date;
echo "<br>";
echo "<p>Ваш вибір:";
echo "<br>";
if($suma == 0 )
{
echo "Ви нічого не замовили на попередній сторінці!<br>";
}
else
{
if ( $cukor>0 )
echo $cukor." цукор<br>";
```

```
if ( $maclo>0 )
echo $maclo." масло<br>";
if ( $kuri>0 )
echo $kuri." кури<br>";
 }
echo "Всього  товару на суму<br>";
$Tovar=$cukor*CinaCuk+$maclo*CinaMas+$kuri*CinaKur;
echo $Tovar."   гривні";
$Tovar=number_format($Tovar, 2, ".", " ");
echo "<P>Всього".$Tovar."</p>";
$outputstring = $date."\t".$cukor." цукор \t".$maclo." масло\t".
$kuri." кури\t\$".$Tovar."\t". "\n";
// відкриття файлу для  запису даних
@ $fp = fopen("orders1.txt", "a");
flock($fp, 2);
if (!$fp)
{
echo "<p><strong> Файл відсутній. "."Спробуйте ще
раз"."</strong></p></body></html>";
exit;
}
fwrite($fp, $outputstring);
flock($fp, 3);
fclose($fp);
echo "<p>Дякуємо!<br></p>";
?>
</body></html>
```

Поля даних будуть записуватися в тому самому порядку, а як роздільник полів буде використатися символ табуляції. Знов-таки, оскільки ці символи невидимі, він представляється керуючою послідовністю "\t". Як роздільник можна використати будь-який символ, що легко читається.

Роздільником, повинен бути будь-який символ, що не буде зустрічатися у вихідних даних, інакше прийдеться корегувати вихідні дані з метою видалення або скасування всіх входжень роздільника.

Використання спеціального роздільника полів спрощує поділ даних на окремі змінні під час зчитування. Кожне замовлення буде оброблятися як окремий рядок.

7.1.4. Зчитування з файла

В прикладі 7.1. ми записали речення в файл, тепер прочитаємо, що записано в файлі.

Приклад 7.2. Зчитування даних з файла.

```
<?
$file = fopen ("file.txt","r");
if ($file){
while(!feof($file)){
$str = fgets($file);
echo $str;
echo ("<br>");  }
fclose ( $file);  }
else
{
echo("Похибка при відкритті файлу");
}
?>
```

Уже зараз клієнти можуть відправляти свої замовлення через Web, однак якщо співробітники Магазину захочуть глянути на замовлення, їм доведеться відкривати файли самостійно.

Давайте створимо Web-інтерфейс, що дозволить працівникам Магазину легко читати файли. У цьому сценарії виконується раніше описана послідовність дій: відкриття файлу, зчитування з файлу, закриття файлу.

```
<?
$fp = fopen("orders1.txt", "r");
flock($fp, 1); //блокування файлу для читання
if (!$fp)
{
echo "<p><strong> Немає доступу
до файлу."
."Спробуйте пізні-
ше.</strong></p></body></html>";
exit;
}
while (!feof($fp))
{
$order= fgets($fp, 100);
$rezult=strpos($order,"львів");
if($rezult>0)
echo $order."<br>";}
flock($fp, 3); //зняття блокування запису
fclose($fp);
?>
```

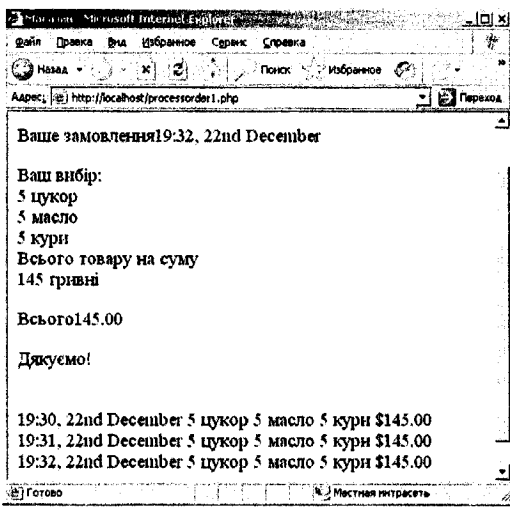


Рис.7.1. **Вікно прикладу 7.2**

Давайте докладніше розглянемо функції, які використовуються в цьому сценарії. Як і раніше, ми відкриваємо файл за допомогою функції

fopen(). На цей раз відкривається тільки для читання, тому використається режим файлу "r":

$fp = fopen("orders.txt", "r");

У цьому прикладі використовується цикл **while** для зчитування з файлу доти, доки не буде досягнутий кінець файлу. Перевірка на наявність кінця файлу здійснюється за допомогою функції feof():

while(!feof($fp)).

Функція feof() приймає в єдиному параметрі вказівник файлу. Вона повертати значення **true,** якщо вказівник файлу перебуває наприкінці файлу. Ім'я функції легко запам'ятати, якщо знати, що **feof** означає File End Of File (Файл: кінець файлу).

В даному випадку (і взагалі при зчитуванні) зчитування з файлу виконується тих пір, поки не буде досягнутий EOF.

7.1.5. Порядкове зчитування: fgets(), fgetss() і fgetcsv()

У розглянутому прикладі для зчитування з файлу використовується функція

$order= fgets($fp, 100).

Ця функція використовується для зчитування з файлу по одному рядку за один раз. У цьому випадку зчитування буде виконуватися доти, доки не зустрінете символ нового рядка (\n), **EOF** або з файлу не буде прочитано 99 байт. Максимальна довжина зчитувального рядка дорівнює зазначеній довжині мінус один байт.

Існує багато різних функцій, які використовують для зчитування з файлів. Функція **fgets()** корисна при роботі з файлами, що містять звичайний текст, який потрібно оброблювати по частинах.

Цікавим варіантом функції **fgets()** є функція **fgetss(), що** має наступний прототип:

string fgetss(int fp, int length, string [allowably_tags]).

Ця функція багато в чому подібна функції **fgets()** за винятком того, що вона вилучає дескриптори PHP і HTML. Функцію **fgetss()** варто використовувати для забезпечення безпеки при зчитуванні файлу, записаного ким-небудь іншим або утримуючі дані, введені користувачем. Відсутність обмежень на наявність у файлі PHP-коду може надати зловмисному користувачеві майже повну свободу дій на сервері.

Функція **fgetcsv()** — ще одна варіація функції **fgets().** Вона має наступний тип: **array fgetcsv(int fp, int length, string [delimiter]).**

Ця функція використовується для поділу рядків файлів. Якщо потрібно відновити змінні замовлення окремо одна від іншої, а не у вигляді рядка тексту, варто вдатися до функції **fgetcsv().** Вона викликається подібно функції fgets(), але їй необхідно передати роздільник, що використовується для поділу полів. Наприклад: $order=fgetcav($fp,100"\t") .

В результаті одержуємо рядок з файлу й розбиваємо його при кожному виявленні символу табуляції (\t). Отримані дані заносяться в масив (у цьому прикладі — в **$order).**

Параметр ***length*** повинен бути більше довжини самого довгого рядка, який зчитується з файлу, в символах.

7.1.6. Зчитування всього файлу: readfile(), fpassthru(), file()

Замість зчитування по одному рядку з файлу за один прохід можна зчитувати весь файл. Існують три різних способи.

Перший полягає у використанні функції **readfile().** Весь раніше створений сценарій можна замінити одним рядком:

readfile("$D0CUMENT_R0OT/../orders/orders.txt").

Функція **readfile()** відкриває файл, повторює його вміст у стандартному вікні браузера, а потім закриває файл. Прототип цієї функції має вигляд:

int readfile(string ім'я_файлу, int [use_include_path]).

Необов'язковий другий параметр вказує, чи потрібно PHP шукати файл у шляху use_include_path, і діє так само, як у функції **fopen().** Функція повертає загальну кількість байтів, зчитаних з файлу.

По-друге, можна використати функцію **fpassthru().** Спочатку необхідно відкрити файл за допомогою функції **fopen().** Потім вказівник файлу можна передати у функцію fpassthru(), що завантажить зміст файлу, починаючи з позиції, заданої вказівником, у стандартне виведення. По завершенні цього процесу функція закриває файл. Раніше наведений сценарій можна замінити функцією **fpassthru()** у такий спосіб:

```
$fp = fopen("orders.txt", "r");
fpassthru ($fp).
```

Функція **fpassthru()** повертає значення **true,** якщо зчитування було виконано успішно, і **false** у протилежному випадку.

Приклад 7.3. Приклад використання функції fpassthru().

```
<?
$file = fopen("12.php","rb");
if(!file)
{
echo("Помилка відкриття файлу ");
}
else
{
fpassthru($file);
 }
?>
```

Третя можливість зчитування всього файлу використання функції **file().** Ця функція ідентична функції **readfile()** за винятком того, що замість

повторення файлу в стандартному вікні виведення, вона перетворить його в масив.

Приклад 7.4. Приклад використання функції file().

```
<?
$file_array = file("gb.txt");
if(!$file_array){
echo("Помилка відкриття файлу ");}
else{
for($i=0; $i < count($file_array); $i++)
{
printf("%s<br>", $file_array[$i]);
}
}
?>
```

Цей рядок приведе до зчитування всього файлу в масив, який називається **$file_array.** Кожний рядок файлу зберігається в окремому елементі масиву.

Формат CSV є одним з форматів, в якому можуть зберігатися файли MSExcel. В наступному прикладі буде прочитано файл створений в MSExcel file.csv.

Приклад 7.5. Зчитування файлу, створеного в MSExcel.

```
<?
$count = 1;
$file =  fopen ("file.csv","r");
while ($data = fgetcsv ($file, 1000, ","))
{
$num = count ($data);
$count++;
```

```
for ($i=0; $i < $num; $i++)
{
print "$data[$i]<br>";
}
}
fclose ( $file );
?>
```

7.1.7. Зчитування символу fgetc()

Використовуючи функцію fgetc(), код зчитує з файлу по одному символу й зберігає його в змінній $char, поки не буде досягнутий кінець файлу. Потім виконується невелика додаткова обробка з метою заміщення текстових символів кінця рядка "\n", HTML-роздільниками рядків "
". Це робиться лише для упорядкування форматування. Оскільки без цього коду браузери не розпізнають нові рядки, весь файл був би виведений у вигляді єдиного рядка.

Побічний ефект використання функції fgetc() замість функції fgets() полягає в тому, що вона буде повертати символ EOF, у той час як fgets() не робить цього. Після зчитування символу доводиться знову виконувати перевірку за допомогою функції feof(), оскільки символ EOF не повинен відображатися у вікні браузера.

У загальному випадку зчитування файлу символ за символом не знаходить особливого застосування, якщо тільки з якої-небудь причини потрібна посимвольна обробка файла.

7.1.8. Зчитування рядків довільної довжини: fread()

Останній спосіб зчитування з файлу, що ми розглянемо — використання функції **fread()** для зчитування з файлу довільної кількості байтів. Ця функція має наступний прототип:

string fread(int fp, int lenqth).

Функція зчитує *length* байтів або всі байти до кінця файлу. Існує ряд інших файлових функцій, які часом можуть виявитися корисними.

7.1.9. Перевірка існування файлу: file_exists()

Якщо необхідно перевірити файл на предмет існування без його відкриття можна скористатися функцією **file_exists(),** як показано в наступному прикладі:

```
If (file_exists("orders.txt") )
echo "There art orders waiting to be processed.",
else
echo "There are currently no orders.";
```

Розмір файлу можна перевірити за допомогою функції **filesize().** Вона повертає розмір файлу, визначений у байтах:

```
echo  filesize("$DOCUMENT_ROOT/. . /orders/orders . txt") .
```

Ця функція може застосовуватися в сполученні з функцією **fread()** для одночасного зчитування всього файлу (або певної його частини). Весь початковий сценарій можна замінити наступним кодом:

```
$fp = fopen("orders.txt", "r");
echo fread( $fp, filesize("orders.txt" )) ;
fclose( $fp ).
```

7.1.10. Копіювання, переіменування та видалення файла

Копіювання файлів здійснюється функцією **copy**:

int copy (string file1, string file2).

Переіменування файла відбувається за допомогою функції **rename**:

int rename (string old, string new).

Якщо після обробки замовлень файл замовлень необхідно видалити, це виконується за допомогою функції unlink() (Немає жодної функції з ім'ям **delete.)** Наприклад:

unlink("orders.txt").

Ця функція повертає значення **false,** якщо файл не може бути вилучений. Як правило, це буде відбуватися при недостатньому рівні прав доступу до файлу або якщо файл не існує.

7.1.11. Переміщення усередині файлу: rewind(), fseek() і ftell()

З'ясовувати позицію вказівника файлу усередині файлу й змінювати її можна за допомогою функцій **rewind(), fseek()** і **ftell().**

Функція **rewind()** переустановлює вказівник файлу на початок файлу. Функція повідомляє в байтах позицію вказівника відносно початку файлу.

Функція **fseek()** може використовуватися для установки вказівника файлу в точку усередині файлу. Її прототип має вигляд:

int fseek(int fp, int offset).

У результаті виклику функції **fseek()** вказівник файлу *fp* встановлюється у файлі в точці, що має зсув ***offset*** байтів відносно початку файлу. Виклик функції **rewind()** еквівалентний виклику функції fseek() зі зсувом, рівним нулю. Функцію fseek() можна використати для знаходження середнього запису у файлі або для виконання бінарного пошуку. Часто, коли подібні завдання потрібно вирішувати відносно досить складного файлу даних, має сенс використовувати базу даних.

7.1.12. Блокування файлів

Уявіть собі ситуацію, коли два клієнти одночасно намагаються замовити товар. Ця ситуація виникає не настільки рідко, особливо коли Web-сайт починає обробляти значні інформаційні потоки. Що відбудеться, якщо один клієнт викличе функцію **fopen()** і починає запис, а потім другий клієнт також викличе функцію **fopen()** і теж спробує виконати запис? Яким у результаті буде вміст файлу? Чи буде спочатку записане перше замовлення, а потім друге, і навпаки? Або ж зміст буде являти собою щось менш корисне, на зразок двох довільних послідовних замовлень? Відповідь на ці

питання залежить від конкретної операційної системи, але часто точно відповісти на них неможливо.

Щоб уникнути подібних проблем, використовується блокування файлів. У PHP блокування реалізується за допомогою функції **flock().** Ця функція повинна викликатися після відкриття файлу, але перед зчитуванням даних з файлу або їхнім записом у файл.

Прототип функції **flock()** виглядає так:

bool flock (int fp, int operation).

У функцію необхідно передати вказівник на відкритий файл і число, що характеризує вид необхідного блокування. Функція повертає значення **true,** якщо блокування було успішно виконане, і **false** у протилежному випадку.

Якщо вирішено використовувати функцію flock(), її варто включити в усі сценарії, у яких використовується даний файл; у противному випадку її застосування позбавлене змісту. Для використання блокування в розглянутому прикладі програму processorder.php необхідно змінити в такий спосіб:

```
$fp=fopen("$DOCUMENT__ROOT/. ./orders/orders . txf ,"a",1);
flock($fp,2); //блокування файлу для запису
fwrite($fp,$outputstring) ;
flock($fp,3); //зняття блокування запису
fclose($fp);
```

Варто також додати блокування у файл vieworders.php:

```
$fp=fopen("orders.txf, "r") ;
flock($fp,1) ; //блокування файлу для читання
//зчитування з файлу
flock($fp,3); //зняття блокування запису
fclose($fp) ;
```

Тепер код більш надійний, але усе ще не ідеальний. Що відбудеться, якщо два сценарії спробують одночасно виконати блокування? Це привело б до конфлікту, коли процеси суперничають за установку блокування, але не відомо, якому з них це вдасться, що могло б викликати нові проблеми. Завдання можна вирішити значно краще, використовуючи СУБД.

7.1.13. Простий лічильник відвідувачів

Наступний сценарій, підраховує кількість звертань до сторінки, у якій він перебуває. Перш ніж переходити до програмного коду, перегляньте алгоритм, написаний на псевдокоді.

1. Привласнити змінній $access ім'я файлу, у якому буде зберігатися значення лічильника.
2. Використати функцію file() для зчитування вмісту $access у масив $visits. Префікс @ перед ім'ям функції подавлює можливі помилки (наприклад, відсутність файлу із заданим ім'ям).
3. Привласнити змінній $current_visitors значення першого (і єдиного) елемента масиву $visits.
4. Збільшити значення $current_visitors на 1.
5. Відкрити файл $access для запису й встановити покажчик поточної позиції в початок файлу.
6. Записати значення $current_visitors у файл $access.
7. Закрити маніпулятор, що посилається на файл $access.

Приклад 7.6. Простий лічильник відвідувачів.

```
<?
// Сценарій: простий лічильник відвідувачів
// Призначення: збереження кількості відвідувачів у файлі
$access = "hits.txt"; // Ім'я файлу вибирається довільно
$visits = @file($access); // Прочитати вміст файлу в масив
$current_visitors = $visits[0]; // Витягти перший (і єдиний) елемент
```

```
++$current_visitors; // Збільшити лічильник відвідувань
$fh = fopen($access, "w"); // Відкрити файл hits.txt і встановити
// покажчик поточної позиції в початок файлу
@fwrite($fh, $current_visitors);// Записати нове значення лічильника
// у файл "hits.txt"
fclose($fh);  // Закрити маніпулятор файлу "hits.txt"
?>
```

Лічильник відвідувань можна написати наступним чином.

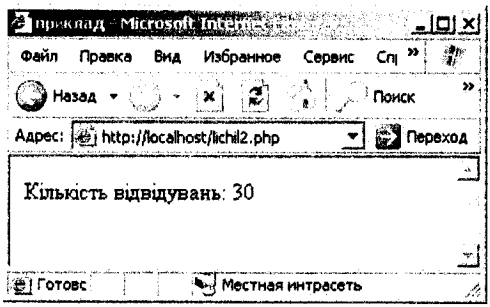


Рис.7.2. **Вікно прикладу 7.6**

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=windows-1271">
<TITLE>приклад</TITLE>
<?
$f = fopen("cunter.txt", "a+");
$count = fread($f, 100);
@$count=$count+1;
ftruncate($f, 0);
fwrite($f, $count);
fclose($f);
?>
</HEAD><BODY>
Кількість відвідувань:
<?
echo $count;
```

```
?>
</BODY></HTML>
```

Приклад 7.7. Успішність групи.

```
<html><head>
<title>Успішність </title>
</head><body>
<h1>УСПІШНІСТЬ ГРУПИ УК-41</h1>
<form action="uk41p.php" method="post">
<table border=0>
<tr><td>НОМЕР</td>
<td><input type=text name=n maxlength=13 size=13><br></td></tr>
<tr><td>ПРІЗВИЩЕ</td>
<td> <input type=text name=f maxlength=30 size=30><br></td></tr>
<tr><td>МАКРОЕКОНОМІКА</td><td> <input type=text name="mak"
maxlength=60 size=30><br></td></tr>
<tr><td>ФІЛОСОФІЯ</td><td><input type=text name="fil"
maxlength=5 size=5><br></td></tr>
<tr><td>ІНФОРМАТИКА</td><td><input type=text name="infor"
maxlength=5 size=5><br></td></tr>
<tr><td colspan=2><input type=submit value="Register"></td></tr>
<tr><td colspan=2><INPUT TYPE="RESET" VALUE= "СКАСУВА-
ТИ"> </td></tr>
<tr><td>Пошук по прізвищу</td></tr>
<tr><td>ПРІЗВИЩЕ</td><td> <input type=text name="ff"
maxlength=30 size=30><br></td></tr>
```

```
<tr><td colspan=2><input type=submit value="пошук"></td></tr>
</table>
</form></body></html>
```

Файл "uk41p.php"

```
<html><head>
<title>успішність</title>
</head><body>
<?
$ n=$_POST[“n “];
$ f=$_POST[“f “];
$ mak =$_POST[“mak “];
$ fil =$_POST["fil "];
$ff =$_POST["ff"];
```

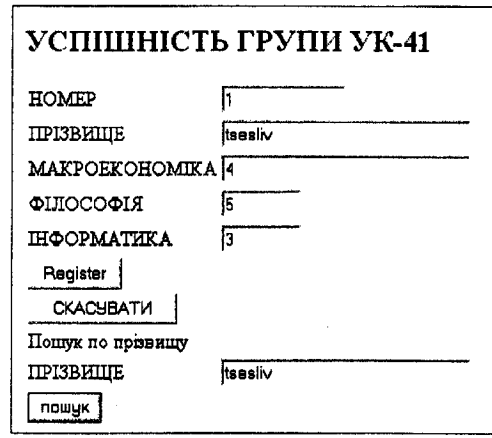


Рис.7.3. **Вікно прикладу 7.7**

```
$uspishnist = $n."номер \t".$f." прізвище\t".$mak. "макроекономі-
ка\t\$".$fil."філософія\t". $infor."інформатика\n";
// відкрити файл для запису даних
@ $fp = fopen("uspish.txt", "a");
flock($fp, 2);//блокування файлу для запису
if (!$fp)
{
echo "<p><strong> Процес не можна виконати зараз. "
."Спробуйте пізніше. </strong></p></body></html>";
exit;
}
//записуємо у файл
fwrite($fp, $uspishnist);
flock($fp, 3); //зняття блокування запису
fclose($fp);
```

```
//зчитуємо дані з файлу
echo"<center></center><i>зчитуємо дані з файлу</i></center> <br>";
$fp = fopen("uspish.txt", "r");
flock($fp, 1); //блокування файлу для читання
if (!$fp)
{
echo "<p><strong>No orders pending."
."Спробуйте знову.</strong></p></body></html>";
exit;}
while (!feof($fp))
{
$order= fgets($fp, 100);
if ($ff=="" ) echo $order."<br>";
}
flock($fp, 3);//зняття блокування запису
fclose($fp);
$fp = fopen("uspish.txt", "r");
if($ff!=""){
while (!feof($fp))
{
$order= fgets($fp, 100);
 $rezult= strpos($order,$ff);
 if($rezult>0)
echo $order."<br>";
}
flock($fp, 3);} //зняття блокування запису
fclose($fp);
```

?></body></html>

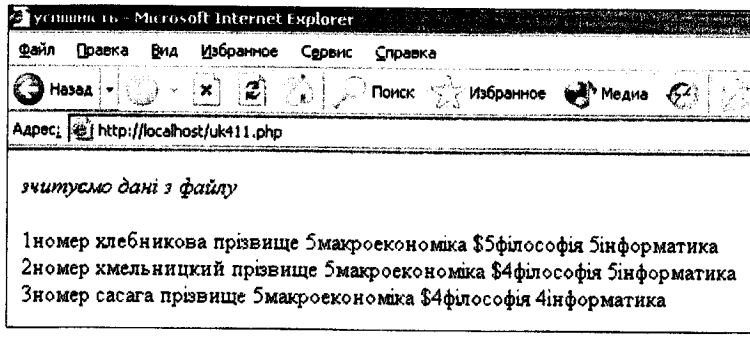
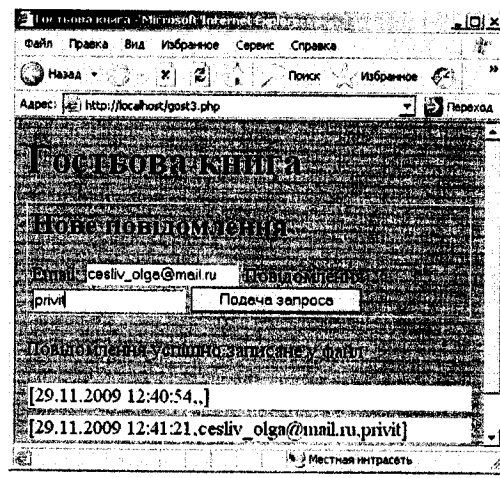


Рис.7.4. **Вікно прикладу 7.7**

7.1.14. Гостьова книга



Приклад 7.8. Гостьову книгу можна написати й наступним чином

```
<?
echo "<html>
<head>
<title>Гостьова
книга</title></head>
<body bgcolor=#00ffff>";
echo "<h1>
```

Рис.7.5. **Вікно прикладу 7.8.**

```
Гостьова книга</h1>";
    $file_gb="olga.txt"; // це файл гостьової книги
    $file_tmp="gb_tmp.txt"; // це тимчасовий файл, що використовується
сценарієм
    $Max=50; // максимальне число повідомлень, які виводяться
```

```
// Функція для виведення файлу gb.txt
function view()
{
$Messages=file('olga.txt');
echo "<p><table width=100%>"; $i=0;
foreach($Messages as $v)
{
$i++;$Max=50;
if ($i % 2 == 0) echo "<tr><td>$v</td></tr>";
else echo "<tr><td bgcolor=white>$v</td></tr>";
// Якщо номер повідомлення = 50 (тобто вже виведено 50 повідом-
лень), перериваємо цикл
if ($i==$Max) break;}}
if (!isset($_Post['Post'])) {
// Виводимо форму для уведення нового повідомлення
echo "<table width=100% border=1>";
echo "<tr><td><h2>Нове повідомлення</h2>
<form method=post action=$SCRIPT_NAME>";
echo " Email: <input type=text name=email>";
echo " Повідомлення <input type=text name=mes>";
echo " <input type=submit name=Post></td></tr>";
echo "</table></form>";
// Виводимо повідомлення
view();
}
else
{
if(file_exists($file_tmp)) die("fatal error, call administrator!");}
```

```
// Додаємо нове повідомлення в початок файлу
if(copy($file_gb, $file_tmp))
{
 if($w=fopen($file_gb,"w"))
 {
flock($w,2); // блокуємо доступ до основного файлу
$email=$_POST['email'];
$mes=$_POST['mes'];
// $mes - це повідомлення користувача
// перед внесенням повідомлення у файл видалимо з нього HTML-
теги
$o=strip_tags($mes);
fwrite($w,"[".date("d.m.Y H:i:s").",".$email.",".$o."] \n");
if(!$r=fopen($file_tmp,"r")) die("can't open file");
flock($r,1);          // блокуємо доступ до тимчасового файлу
while($mes=fgets($r,250))
{
fputs($w,$mes);
}
// Знімаємо блокування й закриваємо файли
flock($r,3);
fclose($r);
flock($w,3);
fclose($w);
// Видаляємо тимчасовий файл
unlink($file_tmp);
}}
echo "Повідомлення успішно записане у файл";
```

```
// Виводимо гостьову книгу
view();
?>
```

7.1.15. Динамічне формування сторінки

Приклад 7.9. Динамічне формування сторінки

1. Створіть 4 файли з іменами: *1.html*, *2.html* и *3.html*,4.html. В кожний з них запишіть невеликий текст.

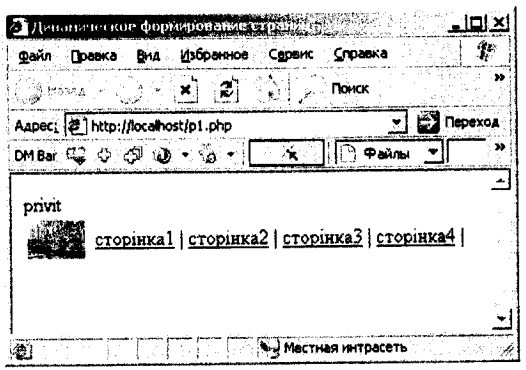


Рис.7.6. **Вікно прикладу 7.9.**

2. Створіть файл *p1.php*, запишіть в нього наступний код:

```
<html><head>
<title> Динамічне формування сторінки </title>
</head>
<body >
<?
    if(!isset($_GET['page'])){      include('1.html')   ;
    //якщо файлу з такою назвою не існує відкриваємо файл
1.html
    } else
    //відкриваємо потрібний файл
    {
        if(file_exists($_GET['page'].'.html')){
        include($_GET['page'].'.html');
    }
```

```
else {        include('1.html');
        }}
        ?>
        <TABLE width="100%">
    <TR>
<TD width=46 align="left"><IMG height=30  src="2.jpg" width=46></TD>
    <TD width=180><BR>
     </TD>
    <TD align="middle">
     <A href="p1.php?page=1">сторінка1</A> |
     <A href="p1.php?page=2">сторінка2</A> |
     <A href="p1.php?page=3">сторінка3</A> |
     <A href="p1.php?page=4">сторінка4</A> |
  </TD>

</TR></TABLE><BR>
    </body>
  </html>
```

Приклад 7.10. Напишемо програму опитування, результати зберігаємо в текстовому файлі.

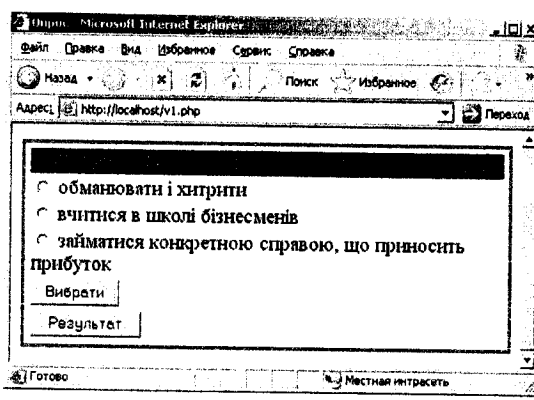


Рис.7.7 **Вікно прикладу 7.10.**

Система обробки опитування складається з двох файлів:

Index.php - це форма, з варіантами відповідей;

v2.php – файл обробки відповідей, зберігання в текстовому файлі.

Index.php

```
<html><head><title>Опрос</title></head>
<body>
<form method ="post" action="v2.php">
<table border="2" cellspacing="0" cellpadding="3"
bordercolor="#336699">
<tr><td><table border=0>
<tr><td  bgcolor="#FF00FF" font color="white">Що означає вислів
"Робити бізнес"?</td></tr>
<tr><td><input type="Radio" name="a" value="1"  > обманювати і хит-
рити</td></tr>
<tr><td><input type="Radio" name="a"value="2"> вчитися в школі бі-
знесменів</td></tr>
<tr><td><input type="Radio" name="a" value="3"> займатися конкрет-
ною справою, що приносить прибуток</td></tr>
</tr>
<tr><td><input type=submit Name=vote value="Вибрати"> </td></tr>
<tr><td><input type=submit Name=result value="Результат">
</td></tr>
</table></td></tr></table>
</form>
</body>
</html>
```

Файл v2.php .

```
<?
$a=$HTTP_POST_FILES['a'];
echo $a;
$file="vote.txt";
$data=file($file);
```

```
$i=1;
$qty=count($data);
$n=0;
while($i<$qty):
$data[$i]=trim(str_replace("\n","",$data[$i]));
$n=$n+$data[$i];
$i++;
endwhile;
$a=$_POST['a'];
if($a!=" "){
echo "<br>Дякую";
$data[$a]++;$n++;
$res="Результат\n".$data[1]."\n".$data[2]."\n".$data[3]."\n".$data[4];
$fp=@fopen($file,"w");
if($fp){ $counter=fputs($fp,$res);
fclose($fp);}
else{echo "помилка запису в файл";}}
else {echo "<br>Результат";}
echo "<br>обманювати і хитрити-<b>".$data[1]."</b>";
echo "<br>вчитися в школі бізнесменів-<b>".$data[2]."</b>";
echo "<br>займатися конкретною справою, що приносить прибуток-
<b>".$data[3]."</b>";
echo "<br>Всього:".$n;
?>
```

7.2. Робота з каталогами

Для установки поточного каталогу застосовується функція chdir:

int chdir (string directory).

Працювати із цією функцією можна в такий спосіб:

- chdir("/tmp/data"); // перехід по абсолютному шляху ;
- chdir("./js"); // перехід у підкаталог поточного каталогу;
- chdir(".."); // перехід у батьківський каталог ;
- chdir("~/data"); // переходимо в /home/користувач/data (для Unix).

Щоб дізнатися шлях до поточного каталогу можна скористатися функцією getcwd:

string getcwd (string path).

Для того щоб відкрити каталог використовується функція opendir, що відкриває каталог, заданий параметром path:

int opendir (string path).

Після того, як каталог відкритий, прочитати його можна функцією readdir:

string readdir (int dir).

Ця функція повертає імена елементів, які знаходяться в каталозі. Крім файлів і папок у каталогах перебувають також елементи "." й "..". Перший елемент вказує на поточний каталог, а другий - на батьківський. Поточний каталог, до речі, можна відкрити, указавши його ім'я як ".":

$dir = opendir (".").

Після того, як робота з каталогом закінчена, його потрібно закрити. Закриття каталогу виконується за допомогою функції closedir:

void closedir ($dir).

Приклад 7.11. Здійснимо читання й виведення файлів, які перебувають у поточному каталозі.

```
<?
 $dir = opendir (".");
 echo "Files:\n";
 while ($file = readdir ($dir))
 {
```

```
  echo "$file<br>";
 }
 closedir ($dir);
?>
```

Помітимо, що ця функція повертає також "." й "..". Якщо цього робити не потрібно, то виключити ці значення можна в такий спосіб:

```
<?
$dir = opendir (".");
while ( $file = readdir ($dir))
{
if (( $file != ".") && ($file != ".."))
{
echo "$file<br>";
}
}
closedir ($dir);
?>
```

Приклад 7.12. Код, що видаляє всі файли з каталогу "c:/temp".

```
<?
function delTemporaryFiles ($directory)
{
$dir = opendir ($directory);
while (( $file = readdir ($dir)))
{
if( is_file ($directory."/".$file))
{
$acc_time = fileatime ($directory."/".$file);
$time =  time();
```

```
if (($time - $acc_time) > 24*60*60)
{
if ( unlink ($directory."/".$file))
{
echo ("Файли успішно вилучені");
}} }
else if ( is_dir ($directory."/".$file) && ($file != ".") && ($file != ".."))
{
delTemporaryFiles ($directory."/".$file);
}}
closedir ($dir);
}
delTemporaryFiles ("c:/temp");
?>
```

Створення каталогів здійнюється за допомогою функції mkdir:

bool mkdir (string dirname, int mode).

Ця функція створює каталог з ім'ям dirname і правами доступу mode. У випадку невдачі повертає false. Права доступу задаються тільки для каталогів UNIX, оскільки в Windows цей аргумент ігнорується.

Приклад 7.13. Нижче наведений приклад створення каталогу test у директорії "c:/temp".

```
<?
$flag = mkdir ("c:/temp/test", 0700);
if($flag)
{
echo("Каталог успішно створений");
}
else
```

```
{
echo("Помилка створення каталогу");
}
?>
```

Видалити каталог можна за допомогою функції rmdir:

bool rmdir (string dirname).

Приклад 7.14. Видалимо тільки що створений каталог /test:

```
<?
$flag = rmdir ("c:/temp/test");
if($flag)
{
echo("Каталог успішно вилучений");
}
else
{
echo("Помилка видалення каталогу");
}
?>
```

Функція rmdir видаляє тільки порожні каталоги. Для того щоб видалити непусті каталоги, давайте напишемо функцію й видалимо каталог "c:/temp" з усіма вкладеними папками й файлами:

Приклад 7.15. Приклад функції, яка видаляє каталог "c:/temp" з усіма вкладеними папками й файлами.

```
<?
function full_del_dir ($directory)
{
$dir = opendir($directory);
while(($file = readdir($dir)))
```

```
{
if ( is_file ($directory."/".$file))
{
unlink ($directory."/".$file);
}
else if ( is_dir ($directory."/".$file) &&
($file != ".") && ($file != ".."))
{
full_del_dir ($directory."/".$file);
}
}
closedir ($dir);
rmdir ($directory);
echo("Каталог успішно вилучений");
}
full_del_dir ("c:/temp")
?>
```

При рекурсивному виклику функції не передавайте як аргументи записи "." й "..", що вказують на поточні й батьківський каталоги, тому що в цьому випадку ви можете втратити ваші дані.

Прииклад 7.16. Дерево каталогів.

```
<?
// Роздрук дерева каталогів файлової системи.
// Функція роздруковує імена всіх підкаталогів у поточному каталозі,
// виконуючи рекурсивний обхід. Параметр $level задає поточну
// глибину рекурсії.
function printTree($level=1) {
// Відкриваємо каталог і виходимо у випадку помилки.
```

```
$d = @opendir(".");
if (!$d) return;
while (($e=readdir($d)) !== false) {
// Ігноруємо елементи.
if ($e=='.' || $e=='..') continue;
// Нам потрібні тільки підкаталоги.
if (!@is_dir($e)) continue;
// Друкуємо пробіли, щоб змістити друк.
for ($i=0; $i<$level; $i++) echo "  ";
// Виводимо поточний елемент.
echo "$e\n";
// Входимо в поточний підкаталог і друкуємо його.
if (!chdir($e)) continue;
printTree($level+1);
// Повертаємося назад.
chdir("..");
// Відправляємо дані в браузер, щоб уникнути видимості зависання
// для більших роздруківок.
flush();  }  closedir($d);}
// Виводимо інший текст фіксованим шрифтом
echo "<pre>";echo "/\n";
// Входимо в кореневий каталог і друкуємо
echo chdir($_SERVER['DOCUMENT_ROOT']);
PrintTree();
echo "</pre>";
?>
```

Функція readdir () повертає черговий елемент каталога. Синтаксис:

string readdir (int resource_handle).

Приклад 7.17. Приклад використання функції readdir().

```
<?
$dp = opendir('records');
while ($file = readdir($dp) ) {
echo $file."
";
}
closedir($dp);
?>
```

7.3. Методи PUT, POST, GET

Протокол HTTP надає три методи для роботи з інформацією, що розміщена на Web-сервері: GET, PUT й POST. Метод GET застосовується для одержання Web-сторінок, при цьому всі змінні форми передаються в URL. Оскільки на багатьох Web-серверах установлене обмеження на максимальну довжину URL (як правило, не більше 1024), не варто застосовувати метод GET, якщо потрібна передача даних більшого обсягу.

Метод PUT застосовується для відновлення інформації на сервері, і вимагає, щоб зміст запиту HTTP PUT зберігалося на сервері. Запит виглядає в такий спосіб:

PUT /path/filename.html HTTP/1.1.

У цьому випадку Web-сервер повинен зберегти вміст цього запиту у вигляді /path/filename.html у просторі імен URL Web-сервера. За замовчуванням сам Web-сервер не виконує такі запити, а задає CGI-сценарій для їхньої обробки. В Apache призначити сценарій для обробки PUT-запитів, можна змінивши директиву script, що перебуває у файлі httpd.conf, приміром, так:

script PUT /cgi-bin/put.cgi.

Це означає, що обробляти PUT-запити буде CGI-скрипт put.cgi.

Як правило, для завантаження файлів на сервер використовують метод HTTP POST. Цей метод дозволяє передавати великі обсяги даних з форми й зберігає всі змінні форми в тілі запиту.

7.3.1. Більш раціональний спосіб обробки: системи керування базами даних

Дотепер у всіх розглянутих прикладах використалися двовимірні файли. У наступному розділі книги буде розглядатися застосування MySQL - системи керування реляційними базами даних. У читачів може виникнути питання: "Для чого це потрібно?"

При роботі із двовимірними файлами виникає ряд проблем:

- Коли двовимірні файли стають великими, робота з ними істотно повільнішає.
- Здійснити пошук конкретного запису або групи записів у двовимірному файлі важко. Якщо записи впорядковані, для пошуку в ключовому полі можна використати який-небудь із видів бінарного пошуку в сполученні із застосуванням записів фіксованої довжини. Якщо потрібно знайти інформацію, що відповідає певному шаблону (наприклад, знайти всіх клієнтів, що проживають у Києві), прийдеться прочитати й перевірити кожний із записів окремо.
- Конкуруючий доступ може породжувати проблеми. Уже було показано, як блокуються файли, але це може привести до виникнення описаної раніше конфліктної ситуації. Крім того, це може привести до утворення "вузького місця" у мережі. При досить інтенсивному інформаційному потоці в сайті великій групі користувачів може знадобитися очікувати розблокування файлу, перш ніж вони зможуть розмістити свої замовлення. Якщо очікування протриває занадто довго, люди звернуться за покупкою кудись в інше місце.
- Вся дотепер розглянута обробка файлів зводилася до послідовної

обробки — тобто зчитування починалося з початку файлу й виконувалося до його кінця. При необхідності вставити записи або видалити їх із середини файлу (при необхідності довільного доступу), це може виявитися скрутним зрештою, прийдеться скопіювати весь файл, виконати зміни й знову записати весь файл. При роботі з великими файлами цей процес сполучений зі значним перевантаженням системи.

Системи управління реляційними базами даних (СУБД) вирішують всі проблеми:

- СУБД можуть забезпечити більш швидкий доступ до даних, чим двовимірні файли. А MySQL, система керування базами даних, володіє одними з найвищих показників продуктивності із СУБД.
- В СУБД можна легко відправляти запит для добування наборів даних за відповідними певними критеріями.
- СУБД мають убудовані механізми обробки конкуруючих запитів, що дозволяє програмістові не турбуватися про це.
- СУБД забезпечують довільний доступ до даних.
- СУБД мають убудовані системи визначення прав доступу. MySQL має суттєво більші можливості в цій області.

Імовірно, головна причина використання СУБД полягає в тому, що всі функціональні можливості, необхідні для системи зберігання даних, у ній вже реалізовані.

Запитання для самоконтролю

1. Яка послідовність запису даних у файл та зчитування файлу?
2. Які функції для зчитування даних з файлу Ви знаєте?
3. З якими проблемами, пов'язано з використання двовимірних файлів?
4. Які проблеми вирішуються за допомогою СУБД?

Задачі для розв'язування

1. Написати програму тестування, результати зберігти в текстовому файлі.
2. Написати код, який видаляє файли з каталогу.
3. Написати код, який створює каталог.
4. Створити масив та записати його в файл, а потім прочитати.
5. Написати код , щоб можна було прочитати файл в Excel.
6. Написати код, який виводить дерево каталогів.
7. Написати простий лічильник відвідувань.
8. Файл містить відомості про співробітників фірми. Структура запису: прізвище співробітника, посада, рік народження. Виконати вибірку інформації про співробітників, що займають посаду менеджера. Кількість записів довільна.
9. З файлу, що містить відомості про абітурієнтів, зарахованих в інститут, виконати вибірку зарахованих на спеціальність N і підрахувати їхню кількість. Структура запису: найменування спеціальності, прізвище студента. Кількість записів довільна.
10. Файл містить відомості про телефони абонентів-підприємств. Кожний запис має поля: найменування абонента-підприємства, рік встановлення телефону, номер телефону. Виконати вибірку абонентів, номера телефонів, яких містять цифру 7. Кількість записів довільна.

РОЗДІЛ 8. СЕСІЇ І COOKIES В PHP

8.1. Призначення Сесії й cookies

Сесії й cookies призначені для зберігання відомостей про користувачів при переходах між декількома сторінками. При використанні сесій дані зберігаються в тимчасових файлах на сервері. Файли з cookies зберігаються на комп'ютері користувача, і по запиту відсилаються браузером серверу.

Використання сесій й cookies дуже зручно й виправдано в таких додатках як Інтернет-магазини, форуми, дошки оголошень, коли, по-перше, необхідно зберігати інформацію про користувачів протягом декількох сторінок, а, по-друге, вчасно надавати користувачеві нову інформацію.

Протокол HTTP є протоколом "без збереження стану". Це означає, що даний протокол не має вбудованого способу збереження стану між двома транзакціями. Необхідний метод, за допомогою якого було б зручно відслідковувати інформацію про користувача протягом одного сеансу зв'язку з Web-сайтом. Одним з таких методів є керування сеансами за допомогою призначених для цього функцій. Сеанс по суті, являє собою групу змінних, які зберігаються й після завершення виконання PHP-сценарію.

При роботі із сесіями розрізняють наступні етапи:

- відкриття сесії;
- реєстрація змінних сесії і їхнє використання;
- закриття сесії.

8.2. Відкриття сесії

Найпростіший спосіб відкриття сесії полягає у використанні функції **session_start**, що викликається на початку PHP-сценарію:

Синтаксис: session_start().

Ця функція перевіряє, чи існує ідентифікатор сесії, і, якщо ні, то створює його. Якщо ідентифікатор поточної сесії вже існує, то завантажуються зареєстровані змінні сесії.

8.3. Реєстрація змінної сесії

Для того, щоб зареєструвати змінну, у якій зберігається ім'я користувача, зазначене їм при реєстрації, ми повинні викликати цю функцію таким чином:

$_SESSION['username'] = "username";

або

$HTTP_SESSION_VARS['username'] = "username".

Таку перевірку можна зробити за допомогою функції **session_is_registered()**:

$result = session_is_registerd("username");

При використанні асоціативних масивів **$HTTP_SESSION_VARS** й **$_SESSION** застосовувати цю функцію не треба, а потрібно прямо перевіряти елементи цих масивів, наприклад, так:

if(isset($_SESSION['username'])).

8.4. Закриття сесії

Після завершення роботи із сесією, спочатку потрібно зняти реєстрацію всіх змінних сесії, а потім викликати функцію **session_destroy**:

Синтаксис: **session_destroy**().

Існують різні способи зняття реєстрації сеансових змінних залежно від того, яким способом вони були зареєстровані.

При включеному **register_globals** і використанні функції **session_register()** зняття реєстрації здійснюється за допомогою функції **session_unregister()**:

Синтаксис: **session_unregister("username").**

Якщо ж реєстрація здійснювалася шляхом застосування асоціативного масиву, то використовують функцію **unset():**

Синтаксис: **unset($_SESSION**["username"]).

Приклад 8.1. Проста сесія.

Розглянемо приклад простої сесії, що працює із трьома сторінками. При відвідуванні користувачем першої сторінки відкривається сесія й реєструється змінна **$username**. Відповідний код реалізації наведений у лістингу:

```
<?
 session_start();
 $_SESSION['username'] = "olga";
 echo 'Привіт,
'.$_SESSION['username']."<br>";
?>
```

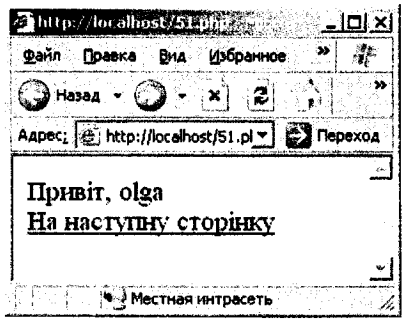


Рис.8.1. **Вікно прикладу 8.1**

```
<a href="page2.php">На наступну сторінку </a>
```

Результат роботи цього сценарію показаний на рисунку.

Після цього, користувач olga натискає на посилання й попадає на сторінку page2.php, код якої наведений у лістингу:

```
<?
 session_start();
 echo $_SESSION['username'].' ,це друга сторінка сайту !';
 echo("<br>");
?>
 <a href="53.php">На наступну сторінку </a>
```

При натисканні на посилання, користувач попадає на сторінку page3.php, при цьому відбувається зняття реєстрації сеансової змінної й знищення сесії. Відповідний код реалізації наведений у лістингу:

```
<?
 session_start();
 unset($_SESSION['username']); // зняти реєстрацію  зі змінної
```

```
echo 'Привіт, '.$_SESSION['username'];
/* тепер им'я користувача не виводиться */
session_destroy(); // вилучаємо сесію
?>
```

Як видно з рисунка, після зняття реєстрації сеансової змінної значення масиву **$_SESSION**['username'] вже не потрібно.

8.5. Використання cookies

Використання cookies зручно як для програмістів, так і для користувачів. Користувачі виграють за рахунок того, що їм не доводиться щоразу заново вводити інформацію про себе, а програмістам cookies допомагають легко й надійно зберігати інформацію про користувачів.

Визначення. *Cookies - це текстові рядки, що зберігаються на стороні клієнта, що утримують пари "ім'я-значення", з якими зв'язаний URL, по якому браузер визначає чи потрібно посилати cookies на сервер.*

Установка cookies здійснюється за допомогою функції **setcookie**:

Синтаксис: **bool setcookie (string name** [, **string value** [, **int expire** [, **string path** [, **string domain** [, **int secure**]]]]])

Ця функція має наступні аргументи:

- **name** - ім'я cookie, що встановлюється;
- **value** - значення, що зберігається в cookie з ім'ям **$name**;
- **expire** - час у секундах з початку сеансу, при закінченні якого поточний cookie стає недійсним;
- **path** - шлях, по якому доступний cookie;
- **domain** - домен, з якого доступний cookie;
- **secure** - директива, що визначає, чи доступний cookie не по запиту HTPPS.

Приклад 8.2. Простий додаток з cookies.

```
<?
```

```
$_COOKIE['counter']++;

setcookie("counter",$_COOKIE['counter']);
        echo 'Ви відвідали цю сторінку
            '.$_COOKIE['counter'].' раз';
```

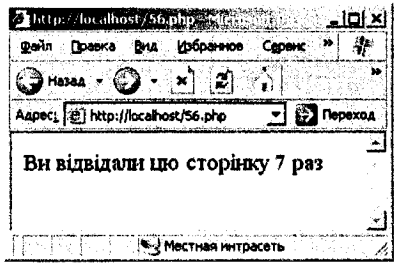


Рис.8.2.**Вікно прикладу 8.2**

```
?>
```

8.6. Встановлення часу придатності cookies

За замовчуванням cookies встановлюються на один сеанс роботи із браузером, однак можна задати для них більше тривалий час існування. Це дуже зручна й корисна властивість, оскільки в цьому випадку користувачеві не потрібно надавати свої дані знову при кожному відвідуванні сайту.

В PHP існують функції **time** й **mktime** для роботи з датою й часом, які дозволяють переводити поточний час у кількість секунд. Функція **time** просто переводить поточний системний час у кількість секунд, що пройшли від початку сеансу **time()**.

Синтаксис: **time().**

Удосконаленим варіантом функції **time** є функція **mktime()**:

Синтаксис: **int mktime ([int hour [, int minute [, int second [, int month [, int day [, int year [, int is_dst]]]]]]])**

Аргумент **is_dst** цієї функції визначає, чи попадає ця дата в період літнього часу й може приймати наступні значення:

- -1 (За замовчуванням. Означає, що властивість не задана);
- 0 (Часовий інтервал не попадає на період літнього часу);
- 1 (Часовий інтервал попадає на період літнього часу).

Приклади встановлення часу придатності cookies:

```
<?
 /* цей cookie дійсний протягом 10 хв після створення */
```

```
setcookie("name", $value, time() + 600);
/* дія цього cookie припиняється опівночі 25 січня 2010 року */
setcookie("name", $value, mktime(0,0,0,01,25,2010));
/* дія цього cookie припиняється в 18.00 25 січня 2010 року */
setcookie("name", $value, mktime(18,0,0,01,25,2010));
?>
```

8.7. Видалення cookies

Видалити cookies просто. Для цього треба викликати функцію **setcookie** і передати їй ім'я того cookies, що підлягає видаленню: **setcookie**("name"). Інші встановлені cookie при цьому не видаляються.

8.8. Проблеми безпеки, пов'язані з cookies

Іноді в cookies доводиться зберігати конфіденційні дані, і в цьому випадку розроблювач повинен подбати про те, щоб інформація, яка зберігається в cookie не була передана третім особам. Існує декілька методів захисту інформації, що зберігається в cookie:

- встановлення області видимості cookies;
- шифрування;
- обмеження доступу для доменів;
- відправлення cookies по захищеному запиті.

Найкращим рішенням є комплексне застосування всіх цих способів.

Запитання для самоконтролю

1. Для чого призначені Сесії й cookies?
2. Якою функцією здійснюється відкриття сесії?
3. Що таке cookies?
4. В чому полягають проблеми безпеки, пов'язані з cookies?

Задачі для розв'язування

1. Написати програму - приклад простої сесії.
2. Написати програму - приклад cookies.

РОЗДІЛ 9. РОБОТА ІЗ ГРАФІКОЮ. БІБЛІОТЕКА GD

9.1. Робота з графічними функціями

Бібліотека GDLib розширює можливості мови PHP і призначена для роботи із зображеннями й файлами графічних форматів, таких як GIF, PNG, WBMP, XPM. За допомогою бібліотеки GDLib можна простим способом створювати власні зображення, зберігати їх у різних форматах або виводити безпосередньо у вікно браузера.

Функція *getimagesize()*

Функція getimagesize() повертає розмір зображення в пікселях і інформацію про зображення.

Синтаксис: array getimagesize (string *filename* [, array *imageinfo*]).

filename - ім'я файлу із зображенням;

imageinfo - при вказівці цього параметра в нього заноситься розширена інформація про зображення.

Функція *imagecreatefromgif()*

Створює зображення в пам'яті з файлу *filename* формату GIF.

Синтаксис: resource imagecreatefromjpeg(string *filename).*

Функція *imagegif()*

Функція imagegif() записує зображення *image* на диск під ім'ям *filename* у форматі GIF.

Синтаксис: int imagegif (resource image [, string *filename]).*

Якщо параметр *filename* не зазначений, то зображення буде виведено у вихідний потік браузера. При виведенні зображення безпосередньо в браузер необхідно передати оглядачеві MIME-тип виведених даних. Це варто зробити за допомогою функції Header ().

Header("Content-type: image/gif").

Приклад 9.1. Рисунок з видимою картинкою.

```
<?php
if (!$image = @imagecreatefromgif('image.gif'))
{
//якщо не було створене зображення на основі image.gif, ми створюємо нову картинку розміром 88 на 31 пикселей
$image = imagecreate(88, 31);
//визначаємо кольори тла:
$backgroundcolor = imagecolorallocate($image, 255, 0, 255);
//заливаємо отримане зображення обраними кольорами:
imagefill($image, 0, 0, $backgroundcolor);
}
//визначаємо кольори шрифту:
$fontcolor = imagecolorallocate($image, 0, 0, 0);
//створюємо випадкове чотиризначне число:
$text = mt_rand(1000, 9999);//виводимо число на картинку:
imagestring($image, 5, 0, 0, 'привіт', $fontcolor);//і видаємо отриманий результат користувачеві:
header("Content-type: image/gif");
imagegif($image);
?>
```

Приклад 9.2.Роботи з графічними функціями.

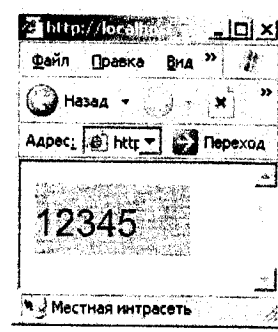


Рис.9.1. **Вікно прикладу 9.2**

```
<?
$ix = 110;
$iy = 50;
$NewImage = ImageCreate($ix, $iy);
$color[0] = ImageColorAllocate($NewImage, 255, 0, 0);
$color[1] = ImageColorAllocate($NewImage, 155, 155, 155);
```

```
ImageFilledRectangle($NewImage, 0, 0, 110, 50, $color[1]);
$size = 20;
$angle = 0;
$x = 0;
$y = 37;
$font[0] = "fonts/antiquai.ttf";
$font[1] ="C:\WINDOWS\Fonts\arial.ttf ";
$font[2] = "fonts/tahoma.ttf";
$font[3] = "fonts/verdana.ttf";
$str = "12345";
ImageTTFtext($NewImage, $size, $angle, $x, $y, $color[0],$font[1] ,
$str);
Header("Content-type: image/gif");
Imagegif($NewImage);
ImageDestroy($NewImage);
?>
```

9.2.Побудова гістограми

Скрипт побудови гістограми – діаграми, яка складається зі стовпців. Розглянемо функцію imagefilledrectangle() , що використовується для побудови гістограм.

Функція *imagefilledrectangle().*

Функція imagefilledrectangle() малює зафарбований прямокутник на зображенні *image*. Прямокутник визначається координатами лівого верхньому й правого нижнього кутів - x1, y1 і x2, y2 відповідно. Кольори прямокутника задаються ідентифікатором кольорів *color*.

int imagefilledrectangle(resource image, int xl, int yl, int x2, int y2, int color).

Приклад 9.3. Гістограма.

Рис.9.2.**Гістограма**

```
<?
global $x1,$y1,$x2,$y2;
// Створення порожнього зображення, розміром 200x200 пікселів
$img = imagecreate (500, 500) ;
// Якщо зображення не створене, виконання скрипта зупиняєте;
if (!$img) exit();
// Визначення білих кольорів на зображенні
$white = imagecolorallocate($img, 255, 255, 255);
// Заливання зображення білими кольорами
imagefill($img, 1, 1, $white);
$sectors = array(80,15,54,16);
// Визначення кольорів тла діаграми
$color = imagecolorallocate($img, 240, 240, 240);
// Змінні $cy й $cy визначають центр діаграми
$cx = $cy = 100;
// Нижня координата стовпців діаграми
// Значення координат відраховують від верхнього лівого кута
$y1 = 200;
// Максимальний розмір зображення по висоті
$max_y = 200;
// Координата x, з якої почнеться побудова діаграми
$x1 = 10;
for($i=0;$i<count($sectors);$i++)
{
```

```
// Формування кольорів для кожного стовпця
$color = imagecolorallocate($img,rand(0, 255), rand(0, 255), rand(0, 255)); // Нормування висоти стовпця. Переклад відсотків у пикселі
$y2 =$y1-$sectors[$i]*$max_y /100; // Визначення другої координати прямокутника
$x2 =$x1-10; // Малювання прямокутника
imagefilledrectangle($img, $x1, $y1, $x2, $y2, $color); // Визначення початкової x-кооринати для наступного стовпця
$x1 = $x2 + 30;}
// Виводимо зображення у браузер у форматі GIF
header ("Content-type: image/gif");
imagegif($img) ;
?>
```

Масив $sectors, який визначається на початку скрипта, містить висоту стовпців у відсотках.

9.3. Робота з шаблонами

Веб-майстрам часто буває необхідно динамічно створювати або змінювати малюнки на своїх сторінках. Це потрібно у тих випадках, коли зображення несуть не винятково декоративну функцію, а містять якусь корисну інформацію. Для цього досить підключити модуль розширення GD.

Завантажити модуль GD можна за адресою www.boutell.com/gd . Для його підключення необхідно забрати знак коментарю в рядку extension=php_gd.dll (для сервера з ОС Windows; у випадку Unix-систем розширення файлу може бути іншим) в php.ini і запустити знову веб-сервер. Різні версії GD можуть працювати з різними форматами графічних файлів. Так, при використанні бібліотеки версії 1.6 і нижче можна створювати зображення у форматах JPEG, GIF й SWF, але не PNG. Більше нові версії дозволяють використати PNG, але відмовляються підтримувати фо-

рмат GIF з ліцензійних міркувань. Всі наведені нижче приклади будуть працювати при використанні GD версії 2.0.1 або вище.

9.4. Створення графічного лічильника відвідувань

Важлива особливість роботи з модулем GD полягає в тому, що скрипт, що формує новий рисунок, не повинен виводити що-небудь крім самого рисунка (тобто в ньому не повинно бути викликів echo, printf і подібних їм функцій).

Створення нового малюнка в PHP починається або зі створення нової чистої "сторінки" (canvas) для малювання, або із завантаження й модифікації вже існуючого зображення. Але перед тим як почати процес виведення графічної інформації, необхідно вибрати його формат (тип MIME) за допомогою виклику функції header(str). Наприклад, для формату PNG необхідно використати наступний код:

header("Content-type: image/png").

Далі для створення області для малювання необхідно викликати функцію int imagecreate (int x_size, int y_size), що передати як параметри x_size й y_size, відповідно, ширину й висоту (у пікселах) формованої картинки; при цьому функція поверне ідентифікатор створеної області для малювання. Якщо ж ми хочемо взяти за основу вже наявну картинку, не залежно від її формату, потрібно викликати функцію imagepng, або imagejpeg, або imagegif, передавши як параметр ім'я файлу-картинки. Для виведення тексту існує функція int imagestring(int im, int font, int x, int y, string s, int col), якій потрібно передати: ідентифікатор області малювання, розмір шрифту (1-5), координату X початку тексту, координату Y початку тексту, сам текст і кольори тексту відповідно. Для визначення кольорів використовується конструкція вигляду:

$white=ImageColorAllocate($im, 255, 255, 255).

Останні три числових параметри - RGB-складові необхідних кольорів. Щоразу вказувати кольори нераціонально, варто створити include-файл із визначенням основних кольорів colors.inc.

Приклад 9.4. Приклад використання шаблону.

```
<?php
$im = imagecreate(88, 31);
$white = ImageColorAllocate ($im, 255, 255, 255);
$black = ImageColorAllocate ($im, 0, 0, 0);
$red = ImageColorAllocate ($im, 255, 0, 0);
$green = ImageColorAllocate ($im, 0, 255, 0);
$blue = ImageColorAllocate ($im, 0, 0, 255);
$yellow = ImageColorAllocate ($im, 255, 255, 0);
$magenta = ImageColorAllocate ($im, 255, 0, 255);
$cyan = ImageColorAllocate ($im, 0, 255, 255);
$l_grey = ImageColorAllocate ($im, 221, 221, 221);
header("Content-type: image/gif");
imagegif($im);
?>
```

Після того як ми намалювали засобами GD нашу картинку, її необхідно вивести в браузер. Для цього залежно від формату малюнка необхідно викликати одну з функцій: imagepng(int im [, string filename]) або imagejpeg (int im [, string filename [, int quality]]), - передавши їй як параметр ідентифікатор картинки. Якщо крім ідентифікатора області малювання вказати ім'я файлу, то зображення буде збережено на диску під цим ім'ям). Після того як ми завершили роботу з малюнком, необхідно звільнити пам'ять, яка займалася малюнком. Для цього служить функція imagedestroy (int im). Розглянемо роботу із цими функціями на прикладі.

Для початку створимо шаблони заголовка й "підвалу" HTML-документа, які будемо використовувати для того, щоб не засмічувати PHP-код конструкціями HTML:

header.tpl:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//RU">
<html><head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<title>Робота із графікою в php</title> </head>
<body>
```

footer.tpl:

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

Допустимо, ми хочемо створити графічний лічильник відвідувань й у нас є готовий файл, у який потрібно "дописати" кількості відвідувань. Він може виглядати так.

Тоді скрипт, що завантажує в готовий файл лічильника дані про відвідування, може виглядати так: (сам підрахунок відвідувань опущений):

counter.php:

```
<?php
  header("Content-type: image/gif");
$im =ImageCreateFromGIf("image.gif");
//include "colors.inc";
$l_grey = ImageColorAllocate ($im, 221, 221, 221);
//визначаємо розмір шрифту для виведення тексту
$fontSize=1;
// задаємо координати для виведення першого рядка
$x1=40; $y1=22;
```

```
// задаємо координати для виведення другого  рядка
$x2=49; $y2=33;
//пишемо кількість відвідувачів
ImageString($im, $fontSize, $x1, $y1, "1 000 000", $blue);
//пишемо кількість відвідувачів сьогодні
ImageString($im, $fontSize, $x2, $y2, "10", $blue);
//виводимо картинку
ImageGIf($im);
// звільняємо пам'ять
imagedestroy($im); ?>
```

Скрипт для відображення лічильника може бути таким:

```
<?php
include "header.tpl";
echo "<center><img
src=\"counter.php\"></center>";
?>
```

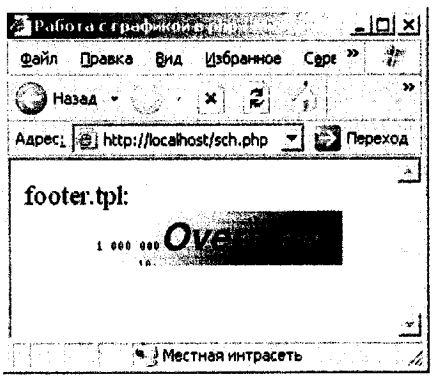


Рис.9.4 **Вікно програми 9.4**

Як бачите, у використанні GD немає нічого складного

Запитання для самоконтролю

1. Для чого використовується бібліотека GDLib ?
2. Яка функція створює картинку?
3. Яка функція визначає колір тла?
4. За допомогою яких функцій зображення виводиться в браузер?

Задачі для розв'язування

1. Створити картинку та написати на ній текст.
2. Побудувати кругову діаграму.

РОЗДІЛ 10. РОБОТА З СЕРВЕРОМ БАЗ ДАНИХ MYSQL

10.1. Архітектура Web-баз даних

Основна операція WEB сервера показана на рисунку 10.1.

Ця система складається з 2 об'єктів: WEB браузера та WEB сервер. Між ними має існувати канал зв'язку. WEB браузер посилає запит на сервер, сервер відсилає відповідь.

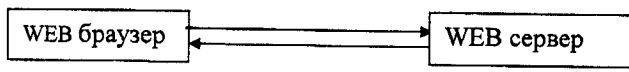


Рис.10.1.**Схема роботи сервера**

Архітектура сайту, який включає базу даних складніше. Розглянемо один з можливих варіантів реалізації. Трансакція складається з наступних етапів

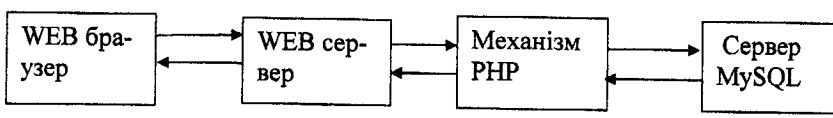


Рис.10.2. **Архітектура сайту, який включає базу даних**

1. WEB браузер відправляє HTTP запит, що визначає WEB сторінку, використовуючи HTML форму.

2. WEB сервер приймає файл та передає його механізму PHP на обробку.

3. Механізм PHP починає синтаксичний аналіз сценарію. В сценарії присутні команди підключення до бази даних і виконання запиту до неї. PHP відкриває з єднання з сервером MySQL та відправляє необхідний запит.

4. Сервер приймає запит в базу даних, опрацьовує його, а потім відправляє результати в PHP.

5. Механізм PHP завершує виконання сценарію, формуючи результати у вигляді HTML, після чого відправляє результати в HTML форматі WEB серверу.

6. WEB сервер пересилає HTML в браузер. За допомогою якого користувач переглядає результати.

Використання мови PHP та MySQL дозволяє робити сайти динамічними, які містять інформацію в реальному часі. MySQL СУБД швидка та надійна. Крім MySQL можна використовувати Postgre SQL, Oracle, dbm, Hyperware, Informix, InterBase, SyBase.

10.2. Мова SQL. Загальні відомості

Запити – це основний інструмент вибірки й обробки даних у СУБД. Для створення і реалізації запитів розроблена спеціальна мова SQL (*Structured Query Language* – мова структурованих запитів).

Перший прототип мови SQL з'явився наприкінці 70-х років і одержав через якийсь час широке поширення. Він став застосовуватися у всіх комерційних СУБД і поступово став стандартом де-факто для мов маніпулювання даними в реляційних БД.

Перша версія стандарту називалася SQL-86 і була прийнята ANSI (Американським національним інститутом стандартів) і ISO (Міжнародним інститутом стандартів).

Наприкінці 1992 р. був прийнятий новий міжнародний стандарт мови – це SQL-92. Цей стандарт підтримується всіма сучасними СУБД, у тому числі і MS Access.

Останній стандарт з'явився в 1999 р., у якому були введені нові типи даних, і ряд інших нововведень, важливих для розроблювачів нових версій

сучасних СУБД. Ми коротко ознайомимося з деякими найбільш важливими елементами мови, що відповідають стандарту SQL-92.

Всі оператори мови можна розділити на такі три категорії:

1. **Оператори контролю даних** – використовуються для перевірки повноважень користувача при звертанні до БД. Це оператори GRANT і REVOKE.
2. **Оператори визначення даних** – використовуються для створення об'єктів БД і визначення їхньої структури. До них відносяться оператори CREATE SCHEMA, CREATE TABLE, CREATE VIEW, CREATE DOMAIN.
3. **Оператори керування даними** – використовуються для пошуку, видалення, зміни і збереження даних. Це оператори SELECT, UPDATE, INSERT, DELETE.

Найважливішим оператором мови SQL є оператор SELECT, призначений для вибірки даних з таблиць БД відповідно до заданого критерію і перетворення отриманих результатів до потрібного виду.

Оператор SELECT. Добір записів з однієї таблиці

Загальний формат оператора SELECT наступний:

SELECT [DISTINCT] {* | <Список полів>}

FROM <Список таблиць >

[WHERE <Умова добору записів >]

[GROUP BY <Список полів для групування>]

[HAVING <Умови добору для груп>]

[ORDER BY <Список полів для сортування>]

Результатом виконання оператора SELECT є набір даних, який складається з записів, що відповідають заданим умовам добору. В операторі обов'язково повинні бути присутніми інструкції SELECT і FROM. Інші інструкції (вказані у квадратних дужках) можуть бути відсутніми.

Інструкція SELECT повідомляє СУБД, що це команда – запит. В інструкції SELECT указується список полів, які будуть включатися в записи, що відбираються. У списку полів повинне бути задане хоча б одне поле. Якщо в список полів потрібно включити всі поля з таблиці (таблиць), то замість перерахування полів можна вказати символ *. Якщо в список полів включаються поля з різних таблиць, то для вказівки належності поля до тієї чи іншої таблиці використовують складене ім'я, що складається з імені таблиці й імені поля, розділених крапкою. Необов'язкова інструкція DISTINCT забороняє включення в результуючий набір даних повторюваних записів.

В інструкції FROM перелічуються імена таблиць, з яких відбираються записи. Список повинний містити хоча б одну таблицю.

В інструкції WHERE задається умова (критерій) добору записів, представлена логічним виразом. Логічний вираз складається з операндів, операцій порівняння і логічних операцій. У якості операндів можуть використовуватися імена полів і константи.

У вираженнях умов добору можуть використовуватися такі операції порівняння і логічні оператори і операції:

=, <, >, <>, <=, >= - операції порівняння;

Between – предикат, що перевіряє приналежність значення поля заданому діапазону значень;

In – предикат, що перевіряє приналежність значення поля заданій множині;

Like – предикат, що перевіряє відповідність значення поля заданому шаблону;

And, Or, Not – логічні операції.

Інструкція GROUP BY призначена для вказівки полів, по яких визначаються групи записів. В одну групу включаються записи з однаковими значеннями в полях, перерахованих в інструкції GROUP BY. Для груп записів можна застосовувати групові операції (їх ще називають агрегатними функціями). У мові SQL визначені такі групові операції:

- *Max*( ) – вибирає максимальне значення поля;
- *Min*( ) – вибирає мінімальне значення поля;
- *Count*( ) – визначає число значень у групі;
- *Avg*( ) – обчислює середнє значення;
- *Sum*( ) – обчислює суму значень полів у групі.

Інструкція HAVING застосовується разом з інструкцією GROUP BY і використовується для завдання умов добору для згрупованих даних. Правила запису умов добору аналогічні правилам завдання умов в інструкції WHERE.

В інструкції ORDER BY указується список полів, по яких потрібно сортування записів у результуючому наборі даних. За замовчуванням сортування по кожному полю виконується в порядку зростання значень. Якщо необхідно зробити сортування по спаданню, то після імені відповідного поля потрібно записати покажчик DESC.

10.3.Створення таблиць баз даних

Наступний етап настроювання бази даних - створення таблиць. Це робиться за допомогою SQL-команди **CREATE TABLE.** Загальна форма оператора **CREATE TABLE** виглядає у такий спосіб:

CREATE TABLE *tablename(columns)*

create table books

(customerid int unsigned not null auto_increment primary key,

name char (30) not null,

title char(40) not null,

price char(20) not null

);

Кожна із таблиць створюється окремим оператором **CREATE TABLE.**

NOT NULL означає, що всі рядки таблиці повинні мати значення в цьому атрибуті. Якщо NOT NULL не зазначене, поле може бути порожнім (NULL).

AUTO_INCREMENT — спеціальна можливість MySQL, яку можна задіяти у числових стовпцях. Якщо при вставці рядків у таблицю залишати таке поле порожнім, MySQL автоматично генерує унікальне значення ідентифікатора. Це значення буде на одиницю більше максимального значення, що вже існує в стовпці. У кожній таблиці може бути не більше одного такого поля. Стовпці з AUTO_INCREMENT повинні бути проіндексованими.

PRIMARY KEY після імені стовпця визначає, що цей стовпець є первинним ключем для таблиці. Дані в цьому стовпці повинні бути унікальними. MySQL автоматично індексує цей стовпець. Помітьте, що раніше, при використанні його с **customerid** у таблиці **customers, без AUTO_INCREMENT** не обійшлося. Автоматичний індекс по первинному ключі зберігає індекс, необхідний **AUTOJNCREMENT.**

Указувати **PRIMARY KEY** після назви стовпця треба лише тоді, коли ми маємо справу з первинним ключем у вигляді одиночного стовпця.

UNSIGNED після цілочисленого типу означає, що його значення може бути або позитивним, або нульовим.

При створенні будь-якої таблиці необхідно прийняти рішення відносно типів стовпців. У таблиці **customers,** як позначено в схемі, існує чотири стовпці. Перший, **customerid,** — це первинний ключ, що визначений безпосередньо. Згідно нашому рішенню, він буде представлятися цілим числом (тип даних int), причому unsigned. Всі інші стовпці будуть містити строкові типи даних. Для них обраний тип char. Він визначає поля фіксованої ширини. Ширина вказується в дужках, тому, наприклад, *ім'я* складається з 30 символів.

Цей тип даних завжди буде призначати 30 символів для імені, навіть якщо не всі символи будуть використовуватися. Для дотримання необхідного розміру MySQL додасть до даних відповідну кількість пропусків.

Для реальних клієнтів, з реальними іменами й адресами, ширина цих стовпців напевно виявиться недостатньою. Зверніть увагу, що всі стовпці оголошені **NOT NULL.** Це мінімальна оптимізація, у результаті якої система буде працювати небагато швидше.

10.4. Робота з базою даних MySQL. Створення бази даних починаємо відкривши сторінку за адресою http://localhost/Tools/phpMyAdmin/.

Створемо першу базу даних firstdb, таблиця в якій буде називатися firma.

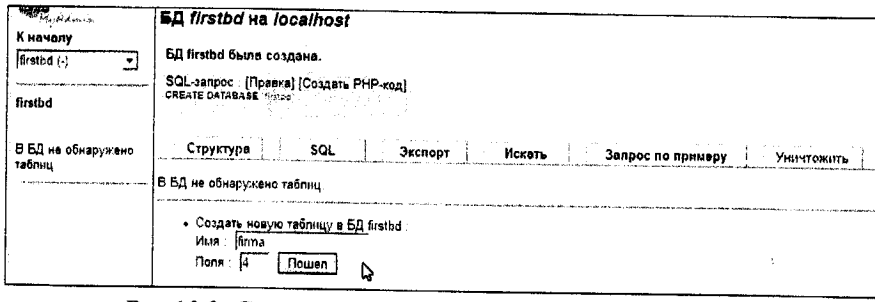


Рис.10.3. **Створення таблиці даних firma в phpMyAdm**

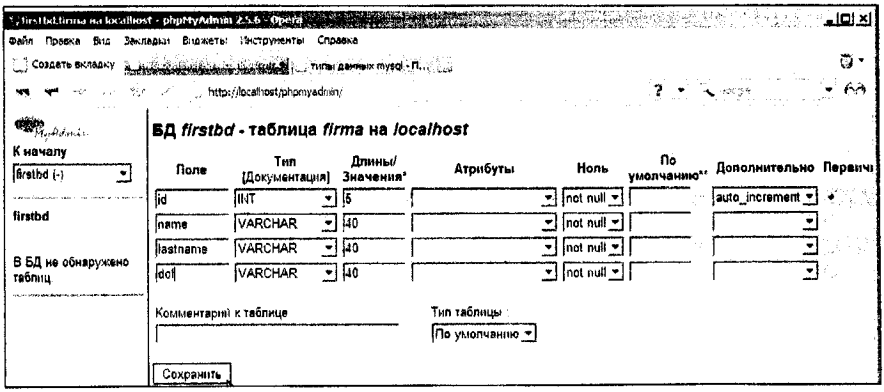


Рис.10.4. **Конструювання таблиці**

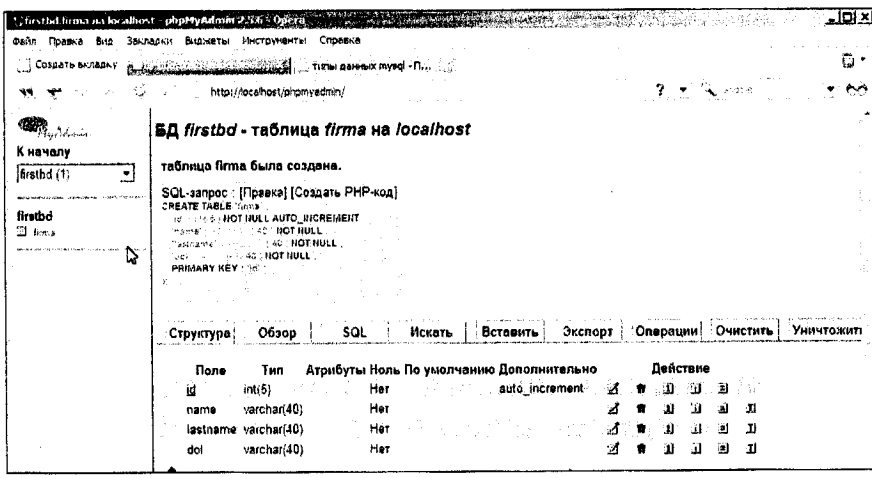


Рис.10.5. **SQL запит на створення таблиці firma**

Заповнюємо таблицю даними.

З'єднання з базою даних.

1.З'єднюємося з MySQL сервером і отримуємо ідентифікатор - mysql_connect("сервер","користувач","пароль").

2. Вибираємо базу, з якою будемо працювати mysql_select_db("і'мя бази", ідентифікатор).

Приклад 10.1.Виведемо і'мя першого співробітника.

```
<?
//З'єднуємося з MySql сервером
$db=mysql_connect("localhost","root");
// З'єднуємося з базою
mysql_select_db("firstdb",$db);
// Створюємо запит, результат зберігаємо в змінній $result
$result=mysql_query("select * from firma",$db);
//Перший рядок БД описуємо, як масив і записуємо в $myrow
$myrow=mysql_fetch_array($result);
// Виводимо з цього масиву ім'я
echo $myrow["name"];
?>
```

Приклад 10.2. Виводимо всіх співробітників в циклі

```
<?
$db=mysql_connect("localhost","root");
mysql_select_db("firstdb",$db);
$result=mysql_query("select * from firma",$db);
$myrow=mysql_fetch_array($result);
do
{
echo "Сотрудник №".$myrow["id"]."<br>";
echo $myrow["name"]."<br>";
echo $myrow["lastname"]."<br>";
echo $myrow["dol"]."<br>";
}
while($myrow=mysql_fetch_array($result));
```

?>

10.5. Створення бази даних Бібліотека

Базу даних називаємо books. Створити. На мові SQL: CREATE DATABASE `books` ;

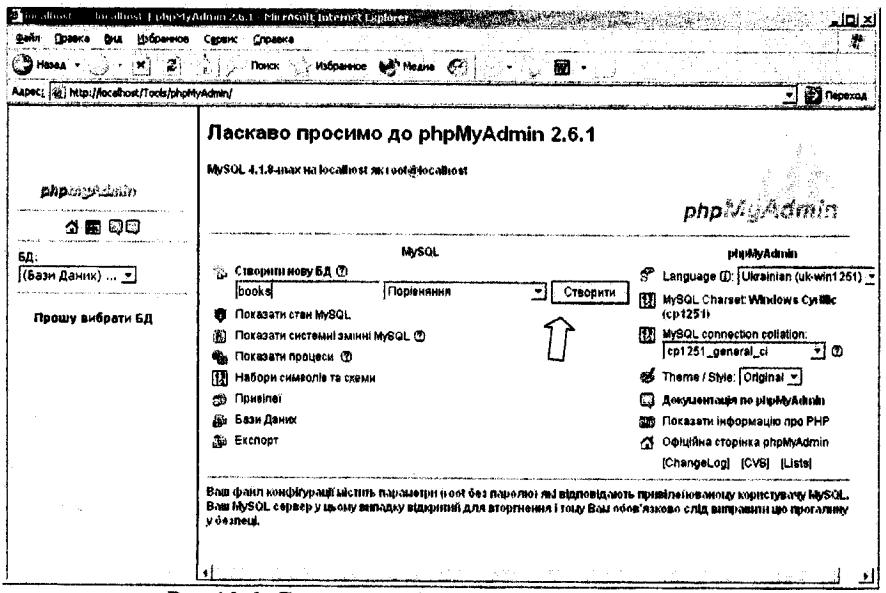


Рис.10.6. **Створення бази даних в phpMyAdmin**

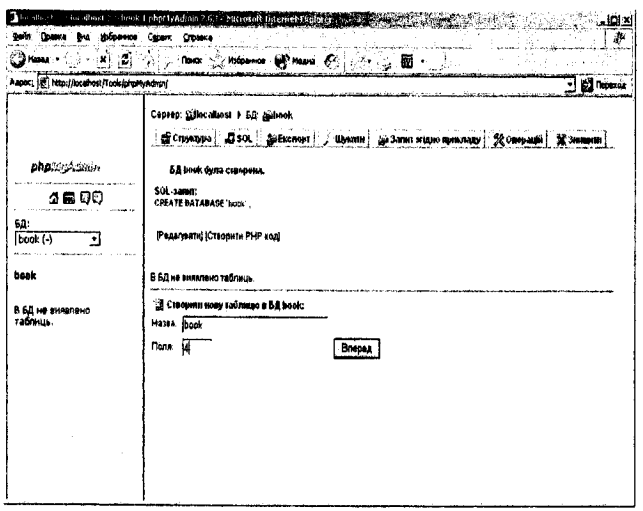


Рис.10.7. **Створення таблиці даних в phpMyAdmin**

Зробимо однотабличну базу даних. Створимо таблицю, яку назвемо book. Ця таблиця матиме 4 поля.

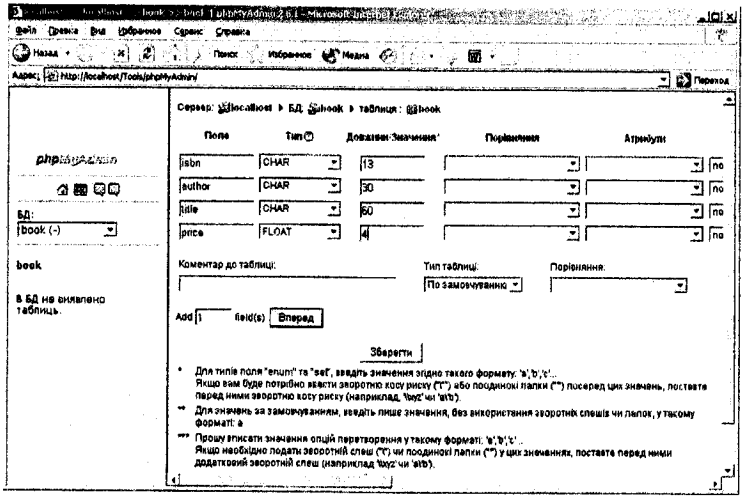


Рис.10.8. **Конструювання таблиці**

Конструюємо таблицю: опишемо кожне поле.

```
CREATE TABLE `books` ( `isbn` CHAR( 13 ) NOT NULL ,`author` CHAR( 30 ) NOT NULL ,`title` CHAR( 60 ) NOT NULL ,`price` FLOAT( 4 ) NOT NULL ,PRIMARY KEY ( `isbn` ) );
```

Заповнюємо таблицю. Після заповнення таблиця має наступний вигляд.

Приклад 10.3. Створимо сайт Бібліотека.

```
<html><head>
        <title>БІБЛІОТЕКА</title>
</head>
    <body>
    <h1>БІБЛІОТЕКА</h1>
```

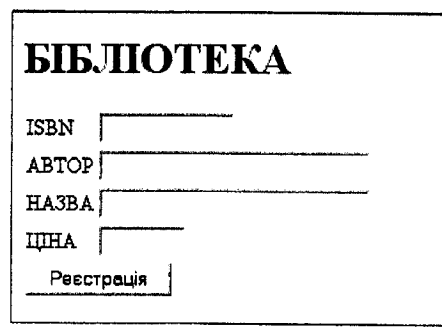


Рис.10.9. **Вікно прикладу 10.1**

```
<form action="insert.php" method="post">
<table border=0>
<tr><td>ISBN</td><td><input type=text name=isbn maxlength=13 size=13><br></td></tr>
<tr><td>АВТОР</td><td> <input type=text name=author maxlength=30 size=30><br></td></tr>
<tr><td>НАЗВА</td><td> <input type=text name=title maxlength=60 size=30><br></td></tr>
<tr><td>ЦІНА </td><td><input type=text name=price maxlength=7 size=7><br></td></tr>
<tr><td colspan=2><input type=submit value="Реєстрація"></td></tr>
</table> </form></body></html>
```

Файл "insert.php"

```
<html><head><title> БІБЛІОТЕКА </title></head>
<body><h1>БІБЛІОТЕКА</h1>
<?
$isbn=$_POST["isbn"];
$author=$_POST["author"];
$title=$_POST["title"];
$price=$_POST["price"];
if (!$isbn || !$author || !$title || !$price)
{echo "Не всі поля введені"."Повторіть ще раз.";
exit; }
$isbn = addslashes($isbn);//форматування полів перед внесеним в базу даних
$author = addslashes($author);
$title = addslashes($title);
$price = doubleval($price);
```

```
@ $db = mysql_pconnect("localhost","root");//з'єднаємося с базою даних
if (!$db)//є така база
{ echo "Error: Не можливо під'єднатися до БД";
exit; }
mysql_select_db("books");//вибрана база даних
$query = "insert into books values ('".$isbn."', '".$author."', '".$title."', '".$price."')";
// запит на вставку значень
$result = mysql_query($query);//виконуємо запит
if ($result)
echo mysql_affected_rows()." КНИГА ДОБАВЛЕНА";
//якщо все успішно з'являється повідомлення
?></body></html>
```

Пошук книги

```
<title>БІБЛІОТЕКА</title>
</head>
<body>
<h2>ПОШУК ІНФОРМАЦІЇ</h2>
<form action="results.php" method = "post">
КРІТЕРІЙ Пошуку: <br>
<select name="searchtype">
<option value="author">АВТОР
<option value="title">НАЗВА
<option value="isbn">ISBN
</select><br>
ВВЕДІТЬ Значення: <br>
```

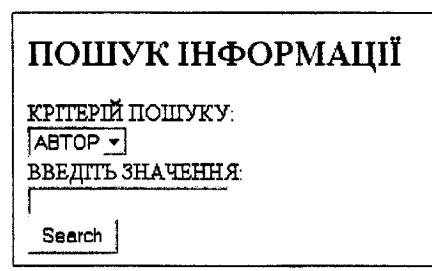


Рис.10.10. **Пошук книги**

```
<input name="searchterm" type=text><br>
<input type=submit value="Search"> </form></body></html>
```

Файл "results.php"

```
<html><head><title>БІБЛІОТЕКА</title></head>
<body><h1>РЕЗУЛЬТАТИ ПОШУКУ</h1>
<?
$searchtype =$_POST["searchtype "];
$searchterm =$_POST["searchterm "];
if (!$searchtype || !$searchterm)
{echo "You have not entered search details.  Please go back and try
again.";
exit;}
$searchtype = addslashes($searchtype);
$searchterm = addslashes($searchterm);
$db = mysql_pconnect("localhost","root");
if (!$db)
{echo "Error: Немає з'єднання з базою даних.";
exit;}
mysql_select_db("books");
$query = "select * from books where ".$searchtype." like
'%".$searchterm."%'";
$result = mysql_query($query);
$num_results = mysql_num_rows($result);
for ($i=0; $i <$num_results; $i++)
{ $row = mysql_fetch_array($result);
echo "<p><strong>".($i+1).". Title: ";
echo stripslashes($row["title"]);
echo "</strong><br>Author: ";
```

```
echo stripslashes($row["author"]);
echo "<br>ISBN: ";
echo stripslashes($row["isbn"]);
echo "<br>Price: ";
echo stripslashes($row["price"]);
echo "</p>";}
?></body></html>
```

10.5. Аутентифікація за допомогою PHP й MySQL

Розглянемо, як використати різні можливості PHP й MySQL для аутентифікації користувачів.

Web - це досить анонімне середовище, проте корисно знати, хто відвідав ваш сайт. На щастя, для конфіденційності відвідувачів, без їхньої допомоги можна одержати тільки дуже незначну інформацію.

Сервери можуть отримати інформації про комп'ютери й мережі, з якими з'єднуються, оскільки Web-браузер ідентифікує себе, вказуючи назву браузера, версію браузера й операційну систему, розміри вікна браузера.

Кожен підключений до Internet комп'ютер має унікальну IP-адресу. Можна довідатися, хто володіє цією адресою, та з певною імовірністю припустити географічне положення відвідувача. В основному, люди з постійним підключенням до Internet мають постійні IP-адреси. А клієнти, які додзвонюються до Internet-провайдерів, у більшості випадків одержують у тимчасове користування один з IP-адрес провайдера. Коли наступного разу ви побачите цю адресу вона вже може використовуватися іншим комп'ютером, а коли ви побачите попереднього користувача, у нього, можливо, буде інша IP-адреса.

Інформація, яку видає браузер, не дозволяє повністю ідентифікувати користувача. Якщо хочете знати ім'я відвідувача й інші деталі, варто запитати це безпосередньо у нього.

Прохання до користувача довести свою особистість називається *аутентифікацією.* Звичайний метод аутентифікації в Web - це вимога до відвідувачів надати унікальне ім'я користувача й пароль. Аутентифікація звичайно використовується для дозволу або заборони доступу до певних сторінок або ресурсів. Аутентифікація може бути необов'язкової або використовуватися для інших цілей, наприклад, для персоналізації.

Найпростіший спосіб реалізації контролюю доступу. Існує тільки одне значення логіна та пароля.

Приклад 10.4. Приклад організації контролю доступу.

```
<?
if(!isset($name)&&!isset($password))
{
//Введіть логін та пароль
?>
<h1>Please Log In</h1>
This page is secret.
<form method = post action = "secret.php">
<table border = 1>
<tr>
<th> Username </th>
<td> <input type = text name = name> </td></tr> <tr>
<th> Password </th>
<td> <input type = password name = password> </td>
</tr><tr>
<td colspan =2 align = center>
```

```
<input type = submit value = "Log In">
</td>
</tr></form>
<?
}
else if($name=="user"&&$password=="pass")
{
// якщо логін та пароль співпадають
echo "<h1>Here it is!</h1>";
echo "I bet you are glad you can see this secret page.";
} else
{
// якщо логін та пароль не співпадають
echo "<h1>Go Away!</h1>";
echo "You are not authorized to view this resource.";
}
?>
```

10.6. Зберігання паролів

Зберігання паролів в окремому файлі на сервері дозволить дуже просто написати програму для додавання й видалення користувачів, а також для зміни паролів.

Всередині сценарію або іншого файлу даних існує обмеження на кількість користувачів, яких можна обслуговувати. Якщо планується зберігати велику кількість елементів у файлі або робити пошук у рамках великої кількості елементів, варто розглянути можливість використання бази даних замість двовимірного файлу. Практичний метод вибору між файлом і базою даних говорить: якщо ви збираєтеся зберігати й робити пошук у більш ніж 100 елементах, варто віддати перевагу базі даних.

Використання бази даних для зберігання імен і паролів відвідувачів не сильно ускладнить сценарій, але дозволить швидко проводити аутентифікацією безлічі користувачів. Це також спростить створення сценарію для додавання й видалення користувачів, а також дасть можливість користувачам змінювати свої паролі.

Сценарій для аутентифікації відвідувачів сторінки з використанням бази даних наведений далі.

Приклад 10.5. Аутентифікації відвідувачів.Створемо базу даних.

```
create database auth;
use auth;
create table auth (
name varchar(10) not null,
pass varchar(30) not null,
primary key (name)
);
insert into auth values
('user', 'pass');
insert into auth values
('testuser', password('test123')); grant select, insert, update, delete on
auth.*
to webauth@localhost
identified by 'webauth';
```

Файл secretdb.php

```
<?
$name =$_POST["name"];
$password =$_POST["password "];
if(! isset($name)&&! isset($password))
{
```

```
//Відвідувач має ввести логін та пароль
?>
<hl>Введіть логін та пароль</hl>
<form method = post action = "se-
cretdb.php">
<table border = 1 >
<tr>
<th> Username </th>
<td> <input type = text name = name> </td>
</tr>  <tr>
<th> Password </th>
<td> <input type = password name = password> </td> </tr> <tr>
<td colspan =2 align = center> <input type = submit value = "Log In">
</td> </tr> </form>
<?
} else
{
// підключення до бази даних
$mysql = mysql_connect( 'localhost', 'webauth', 'webauth');
if(!$mysql)
{ echo 'Cannot connect to database.';
exit;
}
// вибір нашої бази даних
$mysql = mysql_select_db( 'auth' );
if(!$mysql)
{
echo 'Cannot select database.'; exit;
```

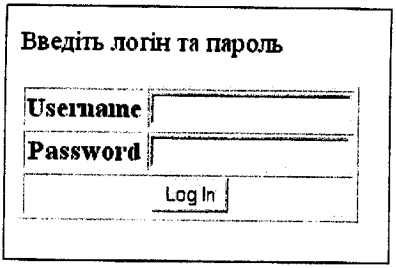


Рис.10.11. **Вікно прикладу 10.3**

```
}
// запит до бази даних
$query = "select count(*) from auth where
name = '$name' and
pass = '$password'"; $result = mysql_query( $query ); if(!$result)
{ echo 'Cannot run query.';
exit;
}
$count = mysql_result( $result, 0, 0 );
if($count>0)
{
// якщо логін та пароль існують в базі даних
echo "<hl>Привіт!</hl>";
} else
{
// якщо логін та пароль не існують в базі даних
 echo "<hl> До побачення! </hl>"; } }
```

10.7. Опитування з використанням MySQL та побудова гістограми

Ваш SQL-запрос был успешно выполнен

SQL-запрос:

```
USE poll; # MySQL вернула пустой результат (т.е. ноль рядов).
CREATE TABLE poll_results(
   candidate [illegible](30),
   num_votes [illegible]
); # MySQL вернула пустой результат (т.е. ноль рядов).
INSERT INTO poll_results
VALUES (
   'John Smith', 0
), (
   'Mary Jones', 0
), (
   'Fred Bloggs', 0
); # Затронутые ряды: 3
GRANT ALL PRIVILEGES ON poll . * TO poll@localhost IDENTIFIED BY 'poll'; # MySQL вернула пустой результат (т.е. ноль рядов).
```

[Правка] [Создать PHP-код]

Рис.10.12. **SQL код для створення бази даних прикладу 10.6**

Приклад 10.6. Опитування.

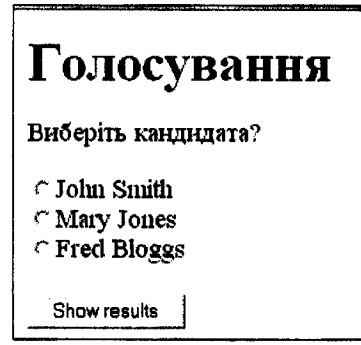


Рис.10.13. **Вікно прикладу 10.6.**

```
<html><head> <title>Polling</title>
</head><body>
<h1>Голосування</h1>
<p>Виберіть кандидата?</p>
<form method=post
action ="show_poll.php">
<input type=radio name=vote value="John Smith">John Smith<br>
<input type=radio name=vote value="Mary Jones">Mary Jones<br>
<input type=radio name=vote value="Fred Bloggs">Fred
Bloggs<br><br>
<input type=submit value="Show results">
</form>
</body></html>
```

Файл show_poll.php

```
<?
$vote=$_POST["vote"];
// Запит з БД
if (!$db_conn = @mysql_connect("localhost", "poll", "poll"))
{
 echo "Could not connect to db<br>";
 exit;
};
@mysql_select_db("poll");
if (!empty($vote))  // if they filled the form out, add their vote
{
$vote = addslashes($vote);
```

```
$query = "update poll_results
set num_votes = num_votes + 1
where candidate = '$vote'";
if(!($result = @mysql_query($query, $db_conn)))
{
echo "Could not connect to db<br>";exit;}};
// запросити поточні дані
$query = "select * from poll_results";
if(!($result = @mysql_query($query, $db_conn)))
{
 echo "Could not connect to db<br>";
 exit;
}
$num_candidates = mysql_num_rows($result);
// підрахувати загальну кількість голосів
$total_votes=0;
while ($row = mysql_fetch_object ($result))
{
$total_votes +=  $row->num_votes;
}
mysql_data_seek($result, 0);  // обнулить покажчик на результати
/*****************************************
Початкові розрахунки для побудови графіка
*****************************************/
// константи
$width=500;       // width of image in pixels - this will fit in 640x480
$left_margin = 50; // простір ліворуч від зображення
$right_margin= 50; // праворуч
```

```
$bar_height = 40;
$bar_spacing = $bar_height/2;
$font = "arial.ttf";
$title_size= 16; // у точках
$main_size= 12; // у точках
$small_size= 12; // у  точках
$text_indent = 10; // положення текстових міток ліворуч
// початкова точка для малювання
$x = $left_margin + 60;  // базова лінія малюнку
$y = 50;
$bar_unit = ($width-($x+$right_margin)) / 100;   // точка  на гістограмі
// висота прямокутника
$height = $num_candidates * ($bar_height + $bar_spacing) + 50;
/*****************************************
 Настроювання опорного зображення
*****************************************/
// створити полотно
$im = imagecreate($width,$height);
// вибрати  кольори
$white=ImageColorAllocate($im,255,255,255);
$blue=ImageColorAllocate($im,0,64,128);
$black=ImageColorAllocate($im,0,0,0);
$pink = ImageColorAllocate($im,255,78,243);
$text_color = $black;
$percent_color = $black;
$bg_color = $white;
$line_color = $black;
$bar_color = $blue;
```

```
$number_color = $pink;
// залити полотно кольорами тла
ImageFilledRectangle($im,0,0,$width,$height,$bg_color);
// Контур полотна
ImageRectangle($im,0,0,$width-1,$height-1,$line_color);
// заголовок
$title = "Результати голосування";
$title_dimensions = ImageTTFBBox($title_size, 0, $font, $title);
$title_length = $title_dimensions[2] - $title_dimensions[0];
$title_height = abs($title_dimensions[7] - $title_dimensions[1]);
$title_above_line = abs($title_dimensions[7]);
$title_x = ($width-$title_length)/2;  // center it in x
$title_y = ($y - $title_height)/2 + $title_above_line; // center in y gap
ImageTTFText($im, $title_size, 0, $title_x, $title_y,
$text_color, $font, $title);
// базова лінія
ImageLine($im, $x, $y-5, $x, $height-15, $line_color);

/*****************************************
 Виведення даних на графік
*****************************************/
// Запросити дані на кожного кандидата
while ($row = mysql_fetch_object ($result))
{
if ($total_votes > 0)
$percent = intval(round(($row->num_votes/$total_votes)*100));
else
$percent = 0;
```

```
        // вивести відсотки
        ImageTTFText($im, $main_size, 0, $width-30, $y+($bar_height/2),
        $percent_color, $font, $percent." %");
        if ($total_votes > 0)
        $right_value = intval(round(($row->num_votes/$total_votes)*100));
        else
        $right_value = 0;
        // довжина доріжки
        $bar_length = $x + ($right_value * $bar_unit);
        // намалювати доріжку для даного значення
        ImageFilledRectangle($im, $x, $y-2, $bar_length, $y+$bar_height,
$bar_color);
        // намалювати  доріжку для даного значення
        ImageTTFText($im, $main_size, 0, $text_indent, $y+($bar_height/2),
        $text_color, $font, "$row->candidate");
        // намалювати доріжку для даного значення
        ImageRectangle($im, $bar_length+1, $y-2,
        ($x+(100*$bar_unit)), $y+$bar_height, $line_color);
        // виведення  числа
         ImageTTFText($im, $small_size, 0, $x+(100*$bar_unit)-50,
$y+($bar_height/2),
       $number_color, $font, $row->num_votes."/".$total_votes);
       // наступна доріжка
       $y=$y+($bar_height+$bar_spacing);
       }

       /*****************************************
       Вивести зображення
```

```
**********************************************/
Header("Content-type:  image/png");
ImagePng($im);
/**********************************************
Очистити ресурси
**********************************************/
ImageDestroy($im);
?>
```

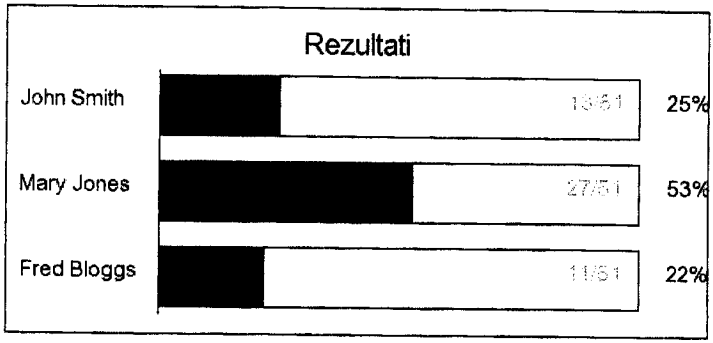


Рис.10.14. **Гістограма прикладу 10.7**

10.8. Голосування з використанням бази даних

Система обробки голосуванн складається з трьох файлів:

Index.php - це форма, з варіантами відповідей;

poll.php – файл обробки відповідей;

poll_results.php - файл обробки запитів до бази даних;

Приклад 10.7. Приклад опитування.

```
<html> <head>
<meta http-equiv="Content-Type" content="text/html;
charset=windows-1251">
<title>Отримані данні форми</title>
</head> <body>
```

```
<form method="POST" action="poll.php">
<p><h1>Конкуренція в перекладі з латинської:</h1></p>
<input type="radio" name="test" value="a1">зіткнення<br>
<input type="radio" name="test" value="a2">змагання<br>
<input type="radio" name="test" value="a3">конкурс<br>
<input type="radio" name="test" value="a4">співпраця<br></p>
<p><input type="submit" value="Відповісти"></p> </form>
<a href="poll_results.php">Результати</a>
</body></html>
```

SQL-запит створення бази даних

```
CREATE DATABASE poll;
USE poll;
CREATE TABLE `poll_results` (`answers` varchar( 30 ) default NULL ,
`voices` int( 11 ) default NULL) ENGINE = MYISAM DEFAULT
CHARSET = cp1251
INSERT INTO `poll_results`
VALUES ('зіткнення', 0);
INSERT INTO `poll_results`
VALUES ('змагання', 1);
INSERT INTO `poll_results`
VALUES ('конкурс', 0);
INSERT INTO `poll_results`
VALUES ('співпраця', 0);
```

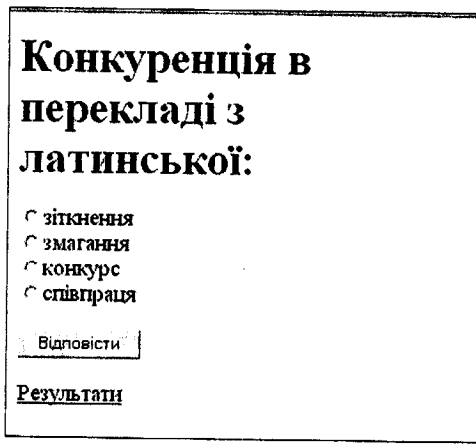


Рис.10.15. **Вікно прикладу 10.7**

Файл Poll.php

```
<?php
$db_conn = @mysql_connect('localhost', 'root', '');
```

```
@mysql_select_db('poll');
 $answer = $_POST["test"];
//echo "<b>  $answer </b><br>";
if ($answer == "a1") $query = "update poll_results set voices = voices + 1
where answers = 'зіткнення'";
if ($answer == "a2") $query = "update poll_results set voices = voices + 1
where answers = 'змагання'";
if ($answer == "a3") $query = "update poll_results set voices = voices + 1
where answers = 'конкурс'";
if ($answer == "a4") $query = "update poll_results set voices = voices + 1
where answers = 'співпраця'";
if(!($result = @mysql_query($query, $db_conn)))
{echo 'Помилка доступу до бази даних';exit;};
echo '<b>Дякуємо! Ваш голос враховано!</b>';
?><br><br>
<a href="poll_results.php">Результати</a><br>
<a href="javascript:history.back(1)">Назад</a>
```

Файл poll_results.php

```
<?
MYSQL_CONNECT('localhost', 'root', '')
or die ("Неможливо створити з'єднання!");
@mysql_select_db('poll')
or die("Неможливо вибрати базу даних!");
echo "<table width=260 border=0 cellspacing=2 cellpadding=0><tr>";
$zapros = mysql_query ('SELECT answers, voices FROM poll_results')
or die ('<p>Помилка доступу до бази даних!');
echo "<tr>";
while ($record = mysql_fetch_array ($zapros))
```

```
{
echo '<td bgcolor=#f0f0f0>', $record['answers'],'</td>
<Td bgcolor=#f0f0f0>', $record['voices'],"</td></tr>";
}
echo "</table>";
?>
<a href="index.html">Назад</a>
```

10.9. Система перевірки знань

Напишемо Систему перевірки знань(СПЗ) (SmartTest), що дозволяє публікувати тести й відповідати на них віддаленно. Це дуже проста СПЗ. У ній передбачені наступні функції:

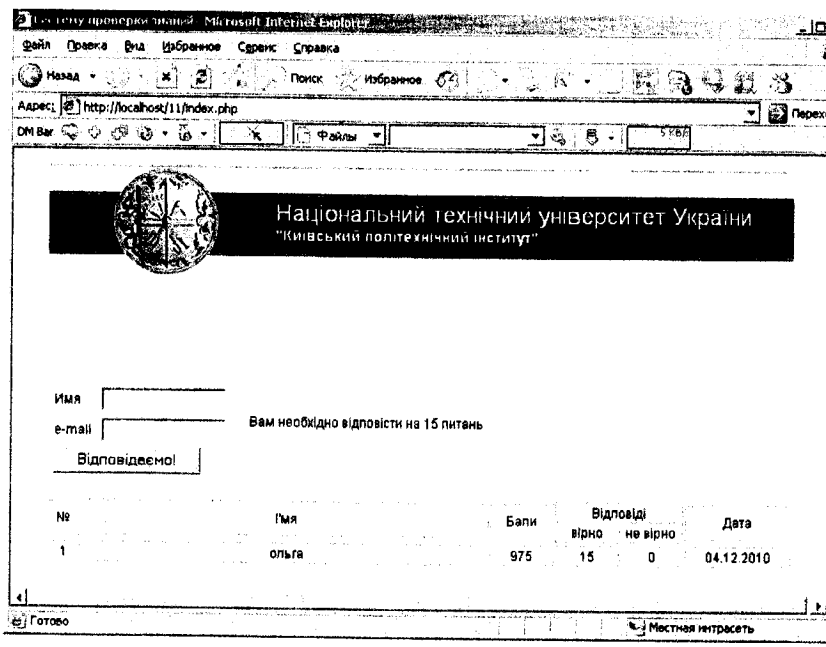


Рис.10.16. **Вікно прикладу Система перевірки знань**

- **Захист від несанкціонованого** доступу. Користувач, перед тим як пройти тест, повинен бути зареєстрований у таблиці користува-

чів SmartTest. Додати запис у цю таблицю може адміністратор СПЗ. Паролі передаються по мережі в зашифрованому виді, ваш пароль не побачить жоден мережний сніффер.

- **Кілька варіантів відповідей.** Максимальна кількість варіантів відповіді - 4.
- **Для кожного тесту ведеться таблиця відповідей.** У неї заноситься ім'я користувача і його оцінка.
- **Номера питань генеруються випадковим чином.**

Користувачеві надаються запитання й 4 варіанти відповідей на питання, він повинен відповісти на них.

Залежно від часу й правильності відповідей він отримує певну кількість балів, виведення результату наприкінці опитування, всього необхідно відповісти на 15 питань.

Для роботи цього додатку необхідно.

1) Скрипти необхідно розмістити на сервері .

2) Відкрийте install.php та index.php блокнотом, відредагуйте дані підключення до MySQL.

3) Запустіть install.php, після створення таблиць вам буде виведене "ok install", тепер необхідно видалити файли install.php та 1.txt.

4) Додаток готовий до роботи.

Файли:

index.php – головний файл;

install.php - скрипт створення таблиць;

header.html – шаблон;

1.txt - база з запитаннями.

Папки: imgs - папка з картинками .Вимоги: Будь-яка платформа, PHP+MySQL.

Сценарій install.php створює таблиці:

- vika_users - таблиця реєстрації відповідей;
- vika - таблиця реєстрації тестів.

install.php

```
<?php
/// Конект до MySQL бази
$sql_host="localhost";  // Хост
$sql_id=" root"; //Логін
$sql_pass="";   // Пароль
$sql_db="cesliv"; //База
// Створюємо таблиці
        $link = @mysql_connect ("$sql_host", "$sql_id", "$sql_pass") or
die ("Немає з'єднання");
        $link2 = @mysql_select_db("$sql_db") or die ("aaa");
$query = "CREATE TABLE vika (
 num int(10) NOT NULL auto_increment,
 vopros text,
 otvet int(1) default NULL,
 otvet1 varchar(200) default NULL,
 otvet2 varchar(200) default NULL,
 otvet3 varchar(200) default NULL,
 otvet4 varchar(200) default NULL,
 PRIMARY KEY  (num)
);";
$sort=@mysql_query($query);
$query = "CREATE TABLE vika_users (
 num int(10) NOT NULL auto_increment,
 user varchar(200) default NULL,
 email varchar(200) default NULL,
```

ip varchar(40) default NULL,
ball int(10) default NULL,
date_a int(20) default NULL,
date_b int(20) default NULL,
vopros int(6) default NULL,
v1 int(1) default NULL,
v2 int(1) default NULL,
v3 int(1) default NULL,
v4 int(1) default NULL,
v5 int(1) default NULL,
v6 int(1) default NULL,
v7 int(1) default NULL,
v8 int(1) default NULL,
v9 int(1) default NULL,
v10 int(1) default NULL,
v11 int(1) default NULL,
v12 int(1) default NULL,
v13 int(1) default NULL,
v14 int(1) default NULL,
v15 int(1) default NULL,
o1 int(1) default NULL,
o2 int(1) default NULL,
o3 int(1) default NULL,
o4 int(1) default NULL,
o5 int(1) default NULL,
o6 int(1) default NULL,
o7 int(1) default NULL,
o8 int(1) default NULL,

```
  o9 int(1) default NULL,
  o10 int(1) default NULL,
  o11 int(1) default NULL,
  o12 int(1) default NULL,
  o13 int(1) default NULL,
  o14 int(1) default NULL,
  o15 int(1) default NULL,
  y1 int(1) default NULL,
  y2 int(1) default NULL,
  y3 int(1) default NULL,
  y4 int(1) default NULL,
  y5 int(1) default NULL,
  y6 int(1) default NULL,
  y7 int(1) default NULL,
  y8 int(1) default NULL,
  y9 int(1) default NULL,
  y10 int(1) default NULL,
  y11 int(1) default NULL,
  y12 int(1) default NULL,
  y13 int(1) default NULL,
  y14 int(1) default NULL,
  y15 int(1) default NULL,
  yes int(4) default NULL,
  no int(4) default NULL,
  session varchar(34) NOT NULL default '',
  PRIMARY KEY  (num)
);";
$sort=@mysql_query($query);
```

```
$ff=file("1.txt");
while (list($key, $value) = each($ff)) {
list($a, $b, $c, $d, $e, $f) = explode('|', $value);
echo "$a, $b, $c, $d, $e, $f<br>";
$f=preg_replace(array("'\|'", "'[\r|\n]'", "'\"'"),array("\|", "", """), $f);
$a=preg_replace(array("'\|'", "'[\r|\n]'", "'\"'"),array("\|", "", """), $a);
$b=preg_replace(array("'\|'", "'[\r|\n]'", "'\"'"),array("\|", "", """), $b);
$c=preg_replace(array("'\|'", "'[\r|\n]'", "'\"'"),array("\|", "", """), $c);
$d=preg_replace(array("'\|'", "'[\r|\n]'", "'\"'"),array("\|", "", """), $d);
$e=preg_replace(array("'\|'", "'[\r|\n]'", "'\"'"),array("\|", "", """), $e);
$query = "INSERT INTO vika (num, vopros, otvet, otvet1, otvet2, otvet3,
otvet4) VALUES (\"\", \"$b\", \"$a\", \"$c\", \"$d\", \"$e\", \"$f\");";
$sort=@mysql_query($query) or die ("$query");
}
echo "ok install<br>видалите файли install.php й 1.txt";
?>
```

Файл index.php

```
<?php
/// Налаштування
$sql_host="localhost";
$sql_id="root";
$sql_pass="";
$sql_db="cesliv";
$vopr=15;
$html="";
if(isset($_GET["page"])) { $page=addslashes($_GET["page"]); } else {
$page=""; }
```

```
if(isset($_GET["session"])) { $session=addslashes($_GET["session"]); }
else { $session=""; }
if(isset($_GET["vs"])) { $vs=addslashes($_GET["vs"]); } else { $vs=""; }
$link = @mysql_connect ("$sql_host", "$sql_id", "$sql_pass") or
die ("Немає конекта");
$link2 = @mysql_select_db("$sql_db") or die ("aaa");
if($page=="end") {
$otvok=0;
$otvno=15;
$ball=0;
$query = "SELECT * from vika_users WHERE session='$session';";
$sort=@mysql_query($query);
$row = @mysql_fetch_array($sort);
$time=$row["date_b"]-$row["date_a"];
$html.="<b>Ваше прізвище:</b> $row[user]<br> <b>Час:</b> $time
секунд<p><hr size=\"1\" color=\"#CFCFCF\">";
for($i=1; $i<=$vopr; $i++) {
$vp="v$i";
$op="o$i";
$sql_o="";
$query = "SELECT * from vika WHERE num='".$row["$vp"]."' LIMIT
1;";
$sort=@mysql_query($query);
$row2 = @mysql_fetch_array($sort);
$tabl1=array(1=>"",2=>"",3=>"",4=>"");
$tabl2=array(1=>"",2=>"",3=>"",4=>"");
$ops=$row["$op"];
$vps=$row2["otvet"];
```

```
if($ops==$vps) {
$tabl1["$ops"]="<table bgcolor=\"#7983C4\" cellspacing=2 cellpad-
ding=2><tr><td width=\"400\" bgcolor=\"#EAF1FF\">";
$tabl2["$ops"]="</td></tr></table>";
$otvok++;
$otvno--;
$ball+=50;
$sql_o.="y$i=1,";
} else {
$tabl1["$vps"]="<table bgcolor=\"#f0f0f0\" cellspacing=1 cellpad-
ding=2><tr><td width=\"400\" bgcolor=\"#ffffff\">";
$tabl2["$vps"]="</td></tr></table>";
$tabl1["$ops"]="<table bgcolor=\"#7983C4\" cellspacing=1 cellpad-
ding=2><tr><td width=\"400\" bgcolor=\"#ffffff\">";
$tabl2["$ops"]="</td></tr></table>";
$ball-=10;
$sql_o.="y$i=0,";
}
$html.=<<<END
<table width="500">
<tr><td align="center"><b>$row2[vopros]</b></td></tr>
<tr><td align="center">
<table width="300">
<tr><td><img src="imgs/1.gif"></td><td>$tabl1[1] $row2[otvet1]
$tabl2[1]</td></tr>
<tr><td><img src="imgs/2.gif"></td><td>$tabl1[2] $row2[otvet2]
$tabl2[2]</td></tr>
```

```
<tr><td><img src="imgs/3.gif"></td><td>$tabl1[3] $row2[otvet3]
$tabl2[3]</td></tr>
<tr><td><img src="imgs/4.gif"></td><td>$tabl1[4] $row2[otvet4]
$tabl2[4]</td></tr>
</table>
</td></tr>
</table>
<hr size="1" color="#CFCFCF">
END;
$v[$i]=$row["v$i"];
}

if(300-$time > 0) { $ball+=300-$time; }
if($ball < 0) { $ball=0; }
$html.=<<<END
<b>Правильні відповіді на</b> $otvok <b>питань</b><br>
<b>Допущено</b> $otvno <b>помилок</b><br>
<b>Всього набрано</b> $ball <b>балів</b>
<div align=center><a href="index.php">Дивитися результат в таблиці
результатів</a></div>
END;
$date_b=date("U");
$sql="UPDATE vika_users SET
yes=$otvok,
no=$otvno,
ball=$ball,
vopros=1,
$sql_o
```

```
date_b=$date_b
WHERE session=\"$session\" AND vopros=0";
$sql_ok=@mysql_query($sql);

} else if(isset($_POST["name"]) AND isset($_POST["email"]) AND
$_GET["session"]) {
$query = "SELECT num from vika";
$sort=@mysql_query($query);
$count=mysql_num_rows($sort);
$countd=round($count/$vopr);
$sql_a="";
for($i=1; $i<=$vopr; $i++) {
$a=$i*$countd-$countd+1;
$b=$i*$countd;
$sql_a.="\"".rand($a, $b)."\", ";
}
$date_a=date("U");
$date_b=date("U");
$user=addslashes($_POST["name"]);
$email=addslashes($_POST["email"]);
$ip="127.0.0.1";
$session=addslashes($_GET["session"]);
$query = "SELECT num from vika_users WHERE session='$session';";
$sort=@mysql_query($query);
$count=mysql_num_rows($sort);
if($count==0) {
if(strlen($user)!=0) {
$sql=<<<END
```

```
INSERT INTO vika_users (num, user, email, ip, ball, date_a, date_b, vo-
pros, v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, v11, v12, v13, v14, v15, o1, o2, o3,
o4, o5, o6, o7, o8, o9, o10, o11, o12, o13, o14, o15, y1, y2, y3, y4, y5, y6, y7,
y8, y9, y10, y11, y12, y13, y14, y15, yes, no, session)
VALUES (
"", "$user", "$email", "$ip", 0, "$date_a", "$date_b", 0, $sql_a 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, "$session"
);
END;
$sql_ok = @mysql_query($sql) or die("fig"); /// создаем таблицы
} else { $html="Заполніть всі поля.<br>Поверніться  назад та
виправте всі помилки "; }
}
else { $sql=1; }
$html = "<a href=\"index.php?session=$session\">играть</a>";
} else if(isset($_GET["session"])) {
if(isset($_POST["vopros"]) AND isset($_POST["vs"])) {
$vopros=addslashes($_POST["vopros"]);
$vs=round(addslashes($_POST["vs"]));
$date_b=date("U");
$sql="UPDATE vika_users SET $vopros=\"$vs\", date_b=$date_b
WHERE session=\"$session\" AND vopros=0";
$sql_ok=@mysql_query($sql);
}
$i=0;
$j=0;
$query = "SELECT * from vika_users WHERE session='$session';";
$sort=@mysql_query($query);
```

```
$row = @mysql_fetch_array($sort);
for($j=1; $j<=$vopr; $j++) {
if($row["o$j"]==0) { $i++;  $v[$i]=$row["v$j"]; $nam[$i]="o$j"; }
}
if(isset($v[1])) {
$query = "SELECT * from vika WHERE num='$v[1]';";
$sort=@mysql_query($query);
$row = @mysql_fetch_array($sort);

$table1="<table bgcolor=\"#f0f0f0\" cellspacing=1 cellpad-
ding=2><tr><td width=\"400\" bgcolor=\"#ffffff\">";
$table2="</td></tr></table>";
$html=<<<END
<table>
<form action="index.php?session=$session" method="post">
<input type="hidden" size="1" name="vopros" value="$nam[1]">
<tr><td><b>$row[vopros]</b></td></tr>
<tr><td>$table1 <img src="imgs/1.gif"><input type="radio" name="vs"
value="1" id="1"> <label for="1">$row[otvet1]</label> $table2</td></tr>
<tr><td>$table1 <img src="imgs/2.gif"><input type="radio" name="vs"
value="2" id="2"> <label for="2">$row[otvet2]</label> $table2</td></tr>
<tr><td>$table1 <img src="imgs/3.gif"><input type="radio" name="vs"
value="3" id="3"> <label for="3">$row[otvet3]</label> $table2</td></tr>
<tr><td>$table1 <img src="imgs/4.gif"><input type="radio" name="vs"
value="4" id="4"> <label for="4">$row[otvet4]</label> $table2</td></tr>
<tr><td><input type="submit" value="Далее"></td></tr>
</from>
</table>
```

```
END;
} else {
$query = "SELECT num from vika_users WHERE session='$session';";
$sort=@mysql_query($query);
$count=mysql_num_rows($sort);
if($count==1) { $html="<meta http-equiv='refresh' content='1;
url=index.php?page=end&session=$session'>Опитування завершено ."; }
else { $html="<meta http-equiv='refresh' content='2;
url=index.php'>Заповніть всі поля.<br> Виправити помилки "; }
}
} else {
$session=md5(date("U")+rand(1,100));
$html=<<<END
<table><tr><td>
<table width=180>
<form action="index.php?session=$session" method="post">
<tr><td>Имя</td><td><input type="text" size="15" name="name" val-
ue=""></td></tr>
<tr><td>e-mail</td><td><input type="text" size="15" name="email"
value=""></td></tr>
<tr><td colspan=2><input type="submit" val-
ue="Відповідаємо!"></td></tr>
</from>
</table>
</td><td>Вам необхідно відповісти на 15
питань<P></td></tr></table>
<hr size="1" color="#CFCFCF">
<table width="100%" cellspacing=1 cellpadding=0>
```

```
END;
$query_as = "SELECT * from vika_users WHERE vopros=1 ORDER BY
ball DESC LIMIT 50";
$sort_as=@mysql_query($query_as) or die ("База пуста 14 $query_as");
$i=0;
$table1="<table bgcolor=\"#99ABD5\" cellspacing=1 cellpadding=1
width=\"100%\">";
$table2="</td></tr></table>";
$html.="<tr>
<td width=30 rowspan=2>$table1 <tr><td height=30 bgco-
lor=\"#EAEAEA\" align=center> № $table2</td>
<td>$table1 <tr><td height=30 bgcolor=\"#EAEAEA\" align=center>
І'мя $table2</td>
<td width=60>$table1 <tr><td height=30 bgcolor=\"#EAEAEA\"
align=center> Бали $table2</td>
<td width=120 colspan=2>$table1 <tr><td bgcolor=\"#EAEAEA\"
align=center colspan=2> Відповіді </td></tr><tr><td width=60 align=center
bgcolor=\"#EAEAEA\"> вірно </td><td width=60 align=center bgco-
lor=\"#EAEAEA\"> не вірно $table2</td>
<td width=100>$table1 <tr><td height=30 bgcolor=\"#EAEAEA\"
align=center> Дата $table2</td>
<tr>
";
$table1="<table bgcolor=\"#99ABD5\" cellspacing=1 cellpadding=1
width=\"100%\"><tr><td bgcolor=\"#F4F4F4\" align=center>";
$table2="</td></tr></table>";
while ($row = @mysql_fetch_object($sort_as)) {
$i++;
```

```
$html.="<tr>
<td width=30>$table1 $i $table2</td>
<td>$table1$row->user$table2</td>
<td width=60>$table1$row->ball$table2</td>
<td width=60>$table1$row->yes$table2</td>
<td width=60>$table1$row->no$table2</td>
<td width=100>$table1".date("d.n.Y", $row->date_b)."$table2</td>
<tr>";
}
$html.="</table>";
}
$template = file ('header.html');
if(!isset($html)) { $html = "Ошибка 404"; $page_name=" : Ошибка
404"; }
if(!isset($page_name)) { $page_name=" : Error Name Page"; }
foreach ($template as $temp) {
$temp = ereg_replace("<!--text-->", "$html", $temp);
echo "$temp";
}
?><?php require_once("include_options.php");?>
```

Файл header.html

```
<HTML>
 <HEAD>
  <TITLE>Систему проверки знаний</TITLE>
<meta http-equiv="Content-Type" content="text/html; charset=windows-
1251">
<LINK REL="stylesheet" type="text/css">
<STYLE>
```

```
BODY { FONT-SIZE: 12px; font-family: Arial, Verdana; text-decoration: none; }
table { font-size : 12; font-family: Arial, Helvetica, sans-serif, Verdana; text-decoration: none; }
a.a { FONT-SIZE: 12px; font-family: Arial, Helvetica, sans-serif, Verdana; COLOR: BLACK; text-decoration: none; }
a.a:hover { FONT-SIZE: 12px; font-family: Arial, Helvetica, sans-serif, Verdana; COLOR: #0B0D6A; text-decoration: underline; }
a { FONT-SIZE: 12px; font-family: Arial, Helvetica, sans-serif, Verdana; COLOR: #0B0D6A; text-decoration: none; }
a.:hover { FONT-SIZE: 12px; font-family: Arial, Helvetica, sans-serif, Verdana; COLOR: #0B0D6A; text-decoration: underline; }
</STYLE>
</HEAD>

<body bgcolor="#FFFFFF">
<div align="center">
<table border=0 cellspacing=0 cellpadding=0 width="550">
<tr>
<td valign="top" width=43><img src="imgs/picture_logo.gif"></td></tr>
<tr><td></td></tr>
<tr><td><br><br><br><br><br><br><!--text--></td></tr>
</table>
<hr size="1" width="750" color="#CFCFCF">
</div></body></html>
```

Файл 1.txt

Текстовий файл створюється в блокноті і має наступний вигляд.

```
1|Що таке тег?|команда мови html|пристрій|засіб|не знаю
```

4|Який тег позначає початок нового абзацу?|B|br|I|p

3|Компютерні мережі виникли|у 30х|у 50х|у 60х|не знаю

10.10.Створення керованого сайту – підключення до Бази Даних

Тексти наших сторінок будемо зберігати в базі даних. Чому не у файлах? Так тому що MySQL набагато зручніше й простіше, ніж текстові файли! Наше перше завдання - створити користувача БД і саму базу даних. Створюємо папку в яку записуємо файл connect.php, який містить налаштування для підключення до БД. Код цього файлу наступний:

```
<?php
$link = mysql_pconnect('localhost', 'root') or die("Не можу з'єднатися");
mysql_select_db('page', $link);
?>
```

Створення Бази Даних (БД)

Для зручності керування базою даних MySQL створена програма PhpMyAdmin. Створимо базу даних "page". Після цього вибираємо пункт меню SQL. У вікні для уведення запитів пишемо:

```
CREATE TABLE `pages` (
`id` INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY ,
`body` TEXT NOT NULL
) TYPE = MyISAM ;
```

Наша таблиця *pages* створена! У перше поле *(id)* будемо записувати ідентифікатори, номера наших сторінок. Тому поле *id* числове (INT), беззнакове, тобто не має негативних значень (UNSIGNED) і, це поле при кожному новому записі самостійно збільшується на одиничку(AUTO_INCREMENT). Друге поле*(body)* у нашій табличці текстове, у нього ми й будемо записувати дані наших сторінок.

Створюємо файл index.php

Створюємо папку 'adm' і переходимо до самого складного й головного завдання - створюємо файл *index.php*, що буде ключовим у нашій панелі адміністратора. Пишемо нашу функцію на php:

```
<?
function show_form(){
      // підключаємо файл  до БД
     require 'inc/connect.php';
      $result = mysql_query("SELECT * FROM pages WHERE id =
'".$_GET['id']."';", $link);
      $row = mysql_fetch_array($result);
?>
<!-- далі йде звичайний HTML -->
<form action="" method="post">
<table cellspacing="1" cellpadding="2" bgcolor="#1F2760">
<tr bgcolor="#B0ADC3">
 <td><p>Текст сторінки</p></td>
</tr>
<tr bgcolor="#ffffff">
 <td>
    <textarea name="body" rows="20" cols="59" class="enter">
          <?// "<?=" те саме, що "<? echo", виведення на екран;-) ?>
          <?=stripslashes($row['body']);?>
    </textarea>
 </td>
</tr>
<tr>
 <td bgcolor="#1F2760" align="right">
    <input type="hidden" name="id" value="<?=$_GET['id'];?>">
    <input type="submit" value="відправити" name="edit">
 </td>
</tr>
</table>
</form>
<?php
} // функция show_form() ззакінчилася
function complete(){
    require 'inc/connect.php';
```

```
    // робимо запит до БД у якому намагаємося витягти сторінку з певним
id
    $result = mysql_query("SELECT * FROM pages WHERE id =
'".$_POST['id']."';", $link);
    // перекидаємо дані з MySQL у асоціативний масив
    $row = mysql_fetch_array($result);
    // перевіряємо чи не порожній елемент масиву id. Якщо порожній, зна-
чить вставляємо наші дані в БД
    if(empty($row['id']))
        $query = "INSERT INTO pages (body) VALUES
('".mysql_real_escape_string($_POST['body'])."'
)";
    // а от якщо не порожній, значить із цим id уже є запис й у цьому випа-
дку ми його просто відредагуємо
    else
        $query = "UPDATE pages SET
                    body = '".mysql_real_escape_string($_POST['body'])."'
            WHERE id = '".$_POST['id']."';";
    // безпосередньо записуємо наші дані в базу (до цього ми просто опису-
вали, що треба зробити, а тепер робимо)
   mysql_query($query, $link);
    // просто виводимо повідомлення, що скрипт відробив
    echo '<h3> Дані оновлені </h3>';
}
function show_pages() {
     require '../inc/connect.php';
     echo '
<table cellspacing="1" cellpadding="2" bgcolor="#1F2760">
<tr bgcolor="#B0ADC3">
 <td>
   <a href="?id=new">Додати сторінку</a>
 </td>
</tr>
</table>';
     echo '
<table cellspacing="1" cellpadding="2" bgcolor="#1F2760">
<tr bgcolor="#B0ADC3">
 <td>
   <b>Номер сторінки</b>
 </td>
</tr>';
```

```
    $result = mysql_query("SELECT * FROM pages ORDER BY id;", $link);
    while($row = mysql_fetch_array($result)){
        echo '
<tr bgcolor="#ffffff">
 <td>
  <a href="?id='.$row['id'].'">'.$row['id'].'</a>
 </td>
</tr>';
    }
    echo '
</table>';
}
if($_POST['edit']) complete(); // якщо натиснути кнопку відправити ", яка називається edit - тоді викликаємо функцію complete()
if($_GET['id']) show_form(); // це новий момент, якщо ми натиснули на посилання в нашій новій функції, то значить ми передали в змінну $_GET['id'] той самий id, що нас цікавить. Тому в цьому випадку викликаємо форму редагування нашої сторінки
else show_pages(); // якщо ми не вибрали певний id - запускаємо нашу останню функцію вибору id.
?>
```

Виведення даних нашого сайту з админкой

Створюємо файл *index.php* у корінь нашого сайту .

```
<?
require 'inc/connect.php';
//htmlspecialchars() Перетворить спеціальні символи в HTML в сутності, будемо вважати для того, щоб найпростіші спроби зламати наш сайт обламалися.
$_GET['id'] = htmlspecialchars($_GET['id']);
// якщо в нас не запитували ніяку певну сторінку, то будемо виводити нашу найпершу. Якщо Ви її давно видалили, поставте замість одиниці ідентифікатор тієї сторінки, що Ви хотіли б завантажити за замовчуванням
if(empty($_GET['id'])) $_GET['id'] = 1;
$result = mysql_query("SELECT * FROM pages WHERE id = '".$_GET['id']."';", $link);
```

```
    $row = mysql_fetch_array($result);
?>
<html>
<head>
  <title></title>
</head>
<body>
<!-- меню -->
<a href="?id=1">перша сторінка</a>
<a href="?id=2">друга сторінка</a>
<a href="?id=3">контакти</a><br /><br />
<?//stripslashes() - Видаляє екранування символів - а їх ми поставили в ад-
мінке, коли завантажували дані в базу за допомогою функції
mysql_real_escape_string()?>
<?=stripslashes($row['body']);?> </body> </html>
```

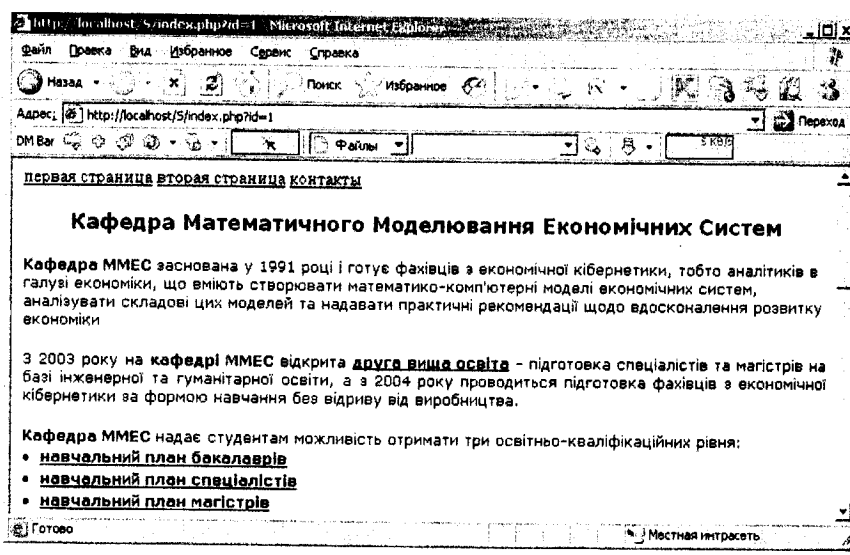


Рис.10.16. **Вікно прикладу 10.8.**

Запитання для самоконтролю

1. Яка архітектура сайту з базою даних?
2. Що Вам відомо про мову SQL?
3. Наведіть приклади використання баз даних.
4. Яка послідовність створення бази даних в phpMyAdmin?
5. Яким чином створюються таблиця бази даних за допомогою SQL?
6. Оператор SELECT призначення?
7. Які статистичні функції використовуються в SQL?

Задачі для розв'язування

1. Створити Систему перевірки знань по будь якому предмету.
2. Створити каталог бібліотеки.
3. Створити базу даних Склад.
4. Створити базу даних Аудит.
5. Створити базу даних анкетування.

РОЗДІЛ 11. ОГЛЯД МЕРЕЖНИХ ФУНКЦІЙ PHP

11.1. Функції для роботи з DNS

При написанні сценаріїв не залежно від мови програмування часто виникає потреба перетворення IP-адреси в доменне ім'я й навпаки. Перетворення IP-адреси в доменне ім'я виконує функція

string gethostbyaddr(string $ip_address).

Перетворення імені хосту в IP-адресу виконує функція

string gethostbyname(string $host).

Якщо вам потрібно одержати всі IP-адреси хоста з ім'ям $host, використайте функцію ***array gethostbynamel(string $host).***

Приклад 11.1. Одержання всіх IP-адрес хоста $host

```
<?
$host="www.yahoo.com";
$ips=gethostbynamel($host);
foreach($ips as $ip) echo $ip;
?>
```

Визначити поштовик для зазначеного хоста hostname можна за допомогою функції *int getmxrr(string hostname, array mxhosts, array [weight]).*

Приклад 11.2. Одержання IP адреси по доменному імені.

```
<?
$hostname="whois.relcom.ru";
$ip_address=gethostbyname($hostname);
echo ("ip $hostname:$ip_address");
?>
```

Приклад 11.3. Визначимо доменне ім'я.

```
<?
```

```
$ip_address="127.0.0.1";
$hostname=gethostbyaddr($ip_address);
echo ("имя хоста  $ip_address: $$hostname");
?>
```

Приклад 11.4. Визначення служби.

```
<?
$port=25;
$protocol="tcp";
$service=getservbyport($port,$protocol);
echo $service;
?>
```

Приклад 11.5. Визначення номера порту, з певною службою.

```
<?
$service='smtp';
//$port=25;
$protocol="tcp";
$port=getservbyname($service,$protocol);
echo $port;
?>
```

11.2. Одержання документа по протоколу HTTP

Одержати документ по протоколу HTTP досить просто:

Приклад 11.6. Одержання документа по протоколу HTTP .

```
<?
$file = join( '', file( 'http://localhost/index.html' ) );
echo $file;
?>
```

У першому рядку прикладу 11.6. ми одержуємо весь документ у рядок $file, а другому - відправляємо документ у браузер. Функція file() повертає масив рядків. N-ий елемент цього масиву відповідає N-му рядку файлу. Якщо нас цікавить HTML-код одержуваного документа, вивести код у браузер допоможе приклад 11.7.

Приклад 11.7. Виведення HTML-коду документа.

```
<?
1. $fcontents = file( 'http://localhost' );
2. while ( list( $line_num, $line ) = each( $fcontents ) ) {
3. echo "<b>Line $line_num:</b> " . htmlspecialchars( $line) . "<br>\n";
4. }
?>
```

Заборонити кешірування можна за допомогою встановлення заголовка Pragma: no-cache. Для повної заборони потрібно використати цілих чотири заголовки. Установити за допомогою Header їх можна так:

```
1.  Header("Pragma: no-cache");
2.  Header("Cache-control: no-cache, must-revalidate");
3.  Header("Expires: Mon, 01 Jan 1990 01:01:01 GMT");
4.  Header("Last-Modified: ".gmdate("D, d M Y H:i:s")."GMT");
```

Перший з них встановлює заголовок заборони кешірування згідно протоколу HTTP/1.0, а другий - HTTP/1.1. Третій визначає дату в минулому, а четвертий встановлює дату останнього відновлення документа. Функція gmdate() повертає дату в потрібному нам форматі. Установлювати всі чотири заголовки вкрай бажано, тому що заборона кешірування може не спрацювати або на проксі-сервері або в браузері, і користувач одержить застарілу версію документа.

11.5. Створення Whois – серверу

Створимо додаток, яке дозволить дізнатися на кого зареєстрована IP адреса. Для цього за допомогою функції fsockopen(); встановлюємо з'єднання з сервером whois.ripe.net, який надає такий сервіс.

Приклад 11.8. Створення Whois – серверу.

```
<html>
<body><center>
<form action="ip.php" metod=get>
<br>
<input type=text name="ip" size=35>
<input type=submit value='введіть IP адресу'>
</form>
</center>
<?
if(!empty($_POST['ip']))
{
$sock=fsockopen("www.ripe.net",43,$errno,$errstr);
if(!$sock) exit("$errno($errstr)");
else
{
fputs($sock,$_POST['ip']);
while(!feof($sock))
{
echo(str_replace
(":",":   nbsp;  ",fgets($sock,128)));
}}fclose($sock);}
?>
```

</body></html>

11.6. Основи завантаження файлів на сервер

Написання PHP-процедури прийому файлу не є складною. При завантаженні на сервер файл тимчасово міститься в деякий каталог, визначений на Web-сервері для цієї мети. Якщо файл не перемістити або не перейменувати, перш ніж сценарій завершить роботу, він буде знищений.

Оскільки HTML-форма містить поле з ім'ям **userfile,** PHP передаються чотири змінні:

- Змінна **$userfile** містить тимчасове місце розташування файлу на Web-сервері.
- Змінна **$userfile_name** містить ім'я файлу в системі користувача.
- Змінна **$userfile_size** містить розмір файлу в байтах.
- Змінна **$userfile_type** містить тип файлу — наприклад, text/plain (текст/простий) або **image/gif** (зображення/gif).

Звертання до них можливо також через масив

```
$userfile=$HTTP_POST_FILES['userfile']['name'];
$userfile_size=$HTTP_POST_FILES['userfile']['size'];
$userfile_type=$HTTP_POST_FILES['userfile']['type'];
```

Якщо відомо розташування й ім'я файла, його можна скопіювати в якийсь інший каталог. Це необхідно зробити до завершення сценарію завантаження, що приведе до видалення файла з тимчасового каталогу. Спочатку створити текстовий файл і зберегти.

Приклад 11.9. Завантаження файла.

upload.php

```
<html><head>
 <title>Administration - upload new files</title>
</head>
```

```
<body>
<h1>Завантаження файла</h1>
<form enctype="multipart/form-data" action="upload1.php"
method="post">
 <input type=Hidden name="MAX_FILE_SIZE" value="1000">
 Upload this file: <input name=userfile type=FILE   >
 <input type="submit" value="Send File">
</form>
</body></html>
```

Upload1.php

```
<html><head>
 <title> Завантаження файла</title>
</head>
<body>
<h1> Завантаження файла</h1>
<?
$MAX_FILE_SIZE=$_POST['MAX_FILE_SIZE'];
echo $MAX_FILE_SIZE;
$userfile=$HTTP_POST_FILES['userfile']['name'];
$userfile_size=$HTTP_POST_FILES['userfile']['size'];
$userfile_type=$HTTP_POST_FILES['userfile']['type'];
echo $userfile_name;
// $userfile is where file went on webserver
// $userfile_name is original file name
// $userfile_size is size in bytes
// $userfile_type is mime type e.g. image/gif
echo "Variables are:<br>";
```

```
        echo $userfile." ".$userfile_name." ".$userfile_size."
".$userfile_type."<br>";
        if ($userfile=="none")
        {
        echo "Problem: no file uploaded";
        exit;
        }
        if ($userfile_size==0)
        {
        echo "Problem: uploaded file is zero length";
        exit;
        }
        if ($userfile_type != "text/plain")
        {
        echo "Problem: file is not plain text";
        exit;
        }
        $upfile = "/www".$userfile_name;
        if ( !copy($userfile, $upfile))
        {
        echo "Problem: Could not move file into directory";
        exit;
        }
        echo "File uploaded successfully<br><br>";
        $fp = fopen($upfile, "r");
        $contents = fread ($fp, filesize ($upfile));
        fclose ($fp);
```

```
        $contents =
strip_tags($contents);
        $fp = fopen($upfile, "w");
        fwrite($fp, $contents);
        fclose($fp);
        echo "Preview of
uploaded    file
contents:<br><hr>";
        echo $contents;
        echo "<br><hr>";
        ?>
        </body></html>
```

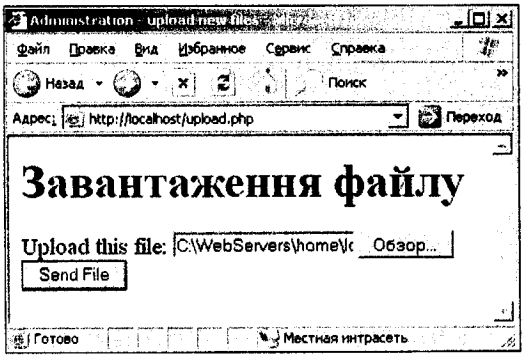


Рис.11.1. **Вікно прикладу 11.9**

11.7. Інші функції

Іноді потрібно записати деяку інформацію, наприклад, повідомлення про помилку, у системний журнал syslog. В PHP для цього передбачена ціла серія функцій:

1. int openlog(string ident, int option, int facility);
2. int syslog(int priority, string message);
3. int closelog(void).

Перша з них відкриває з'єднання з syslog. Друга - породжує системне повідомлення (іншими словами записує повідомлення із зазначеним пріоритетом до протоколу). Функція closelog() закриває з'єднання протоколу.

Питання для самоконтролю

1. Наведіть приклад найпростіших мережних функції PHP ?
2. Які функції для роботи з DNS Ви знаєте?
3. Які функції використовуються при завантаженні файлу?

РОЗДІЛ 12. ПУБЛІКАЦІЯ САЙТУ НА СЕРВЕРІ

12.1. Хостінг

У попередніх розділах розглядалися мови HTML та PHP, їх можливості по створенню Web-сторінок. Тепер вам потрібно з окремих сторінок сконструювати свій Web-сайт і розмістити його в Інтернеті.

Приступаючи до розробки свого сайту, потрібно чітко визначити його призначення. Звичайно сайти створюються для того, щоб заявити про себе чи про свою організацію, повідомити про результати роботи чи про свої досягнення, налагодити ділові зв'язки, дати рекламу про товари чи послуги тощо.

Крім призначення сайту, потрібно визначити коло його потенційних відвідувачів, тобто *аудиторію.*

Наступним етапом буде *підбір матеріалу.* Не весь матеріал по тематиці сайту, що ви маєте, варто публікувати в Інтернеті. Потрібно оцінити якість матеріалу та його цікавість для відвідувачів. При плануванні сайту, призначеного для якої-небудь організації, важливим є питання фінансування робіт.

Матеріали, що ви плануєте опублікувати на сайті, потрібно організувати у визначену структуру. Найчастіше для Web-сайтів обирається деревоподібна структура організації інформації. Коли сайт

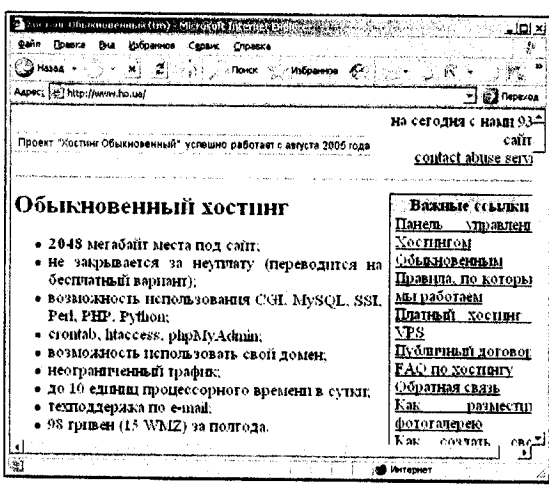


Рис.12.1 **Пошук хостінгу в Інтернет**

створено, його можна переглянути на власному комп'ютері. Для цього достатньо клацнути на піктограмі файлу index.htm - головна сторінка сайту відкриється програмою Internet Explorer. Щоб сайт був доступним для інших користувачів мережі, його потрібно розташувати на сервері. Цей процес називається публікацією сайту, або ***хостінгом.*** Під час публікування користувач пересилає html-файли з власного комп'ютера на сервер у режимі FTP-з'єднання. Для цього використовують деяку програму класу FTP-клієнт: CuteFTP, Windows Commander, Far Manager тощо. Публікацію можна здійснити автоматично, користуючись FTP-сервісом, вбудованим у програми MS FrontPage, Macromedia DreamWeaver, у середовищі Windows, якщо доступний Майстер Web-видань а також за допомогою засобів автоматизації процесу публікації, які надають сервери.

Сервер, на якому розміщують сайт, називається ***хостом,*** а сервіси, які надають його служби, - ***хостінгом.*** Хостінг буває безплатним або платним. Безплатний хостінг передбачає надання поштової web-скриньки і деякого обмеженого дискового простору для розташування сайту, причому на такому сайті автоматично буде розташована чужа реклама у вигляді банерів. Часто тут надаються засоби для сайтів на базі готових шаблонів, дене-де є шаблонні гостьові книги, форуми, чати тощо. Платний хостінг дає змогу не лише уникнути чужої реклами на власному сайті, але й організувати свою рекламу на інших сайтах, використати додаткові сервіси тощо. Якщо користувач є клієнтом місцевого web-провайдера, то доцільно сайт публікувати на його сервері. У мережі є велика кількість серверів, які надають безплатний хостінг. Для україно - чи російськомовних сайтів доцільно вибирати сервер чи портал, розташований в межах СНД.країн СНД, наприклад, Яндекс із сервером Narod.ru (www. narod.ru). Увійшовши на деякий сервер, варто ознайомитися зі списком багатьох інших серверів, що надають безплатний хостінг, та умовами співпраці з ними. Для цього

потрібно виконати пошук за словами „Бесплатный хостинг" чи „free hosting" - отримаєте не тільки адреси серверів, але й корисну інформацію про обмеження і наявність чи відсутність тих чи інших сервісів на них.

12.2. Реєстрація на сервері

Для користування платним чи безплатним хостінгом користувач спочатку повинен зареєструватися на сервері за допомогою команди Реєстрація. На деяких серверах під час реєстрації потрібно зазначати власну електронну адресу, куди сервер відішле повідомлення про результати реєстрації, підтвердить адресу сайту, логін і пароль. На інших це не вимагається - поштова адреса надається під час реєстрації і може мати, наприклад, такий вигляд: логін.narod.ru Є багато серверів, які надають безплатну електронну скриньку.

Логін (логічне ім'я) і пароль вказує користувач, дотримуючись вимог серверу, наприклад, логін може містити лише латинські літери і цифри та має починатися з літери; пароль має містити не менше чотирьох символів і не може збігатися з логіном. Під час заповнення реєстраційної карти пароль вводять двічі. Якщо користувач забуде пароль чи логін, він втратить доступ до сервісів, тому ці дані рекомендується занотовувати і зберігати в таємниці. Особливо забудькуваті можуть скористатися системою контрольного запитання і відповіді, поля яких варто заповнити під час реєстрації. На підставі цієї інформації сервер ідентифікує користувача і надає можливість ввести новий пароль. Деякі сервери надсилають на електронну адресу користувача втрачені дані. Приклади контрольних запитань: дівоче прізвище матері, ваша улюблена страва, поштовий індекс батьків. Можна сподіватись, що ця інформація відома не всім. Під час роботи з реєстраційною формою стежте, щоб були заповнені всі поля, позначені зірочками. Після заповнення реєстраційної форми потрібно натиснути на кнопку Ok і зачекати підтвердження реєстрації. Якщо введений користувачем логін

уже зайнятий чи пароль був уведений неправильно, чи залишились незаповненими обов'язкові поля, то реєстрацію доведеться повторити. Якщо реєстрація пройшла успішно, можна розпочати публікацію сайту. Насамперед корисно ознайомитися з розділом Угода з користувачем (Пользовательское соглашение), де є важлива інформація про умови надання послуг на сервері і правила поведінки користувачів - правила мережевого етикету.

Публікація засобами серверу. Для публікації сайту потрібно зайти на сервер, наприклад Narod.ru тощо, як зареєстрована особа, увівши у відповідні поля логін і пароль та натиснувши кнопку Увійти. На сервері Народ після входу потрібно вибрати посилання Створити сайт.

12.3. Розміщення сайту за допомогою програми FAR

Перед розміщенням сайту необхідно одержати у власника Web-серверу адресу URL, логін (ім'я) і пароль. Далі дійте за таким сценарієм.

Підключіться до Інтернету і завантажте програму FAR.

Натисніть клавіші Alt+Fl (F2) і виберіть у меню, що з'явилось, опцію FTP.

На панелі програми FAR ви побачите список FTP-з'єднань, встановлених з даного комп'ютера. Якщо ви користуєтесь FTP уперше, то даний список буде порожнім. Натисніть клавіші Shift+F4, у результаті чого з'явиться діалог для надання FTP-адреси

У верхній рядок діалогу введіть відповідно до шаблона логін, пароль і URL сайту. Логін і пароль розділяються двокрапкою, а пароль і URL - значком @. Замість імені сервера ви можете використовувати його IP-адресу, якщо вона вам відома, наприклад, 195.230.142.115.

Після того як ви розмістили свій сайт на Web-сервері, ви можете редагувати його змісту (додавати, видаляти, обновлювати файли). Якщо ви користуєтеся безкоштовним хостінгом, то редагувати сайт можна за допомогою передбаченого для цього інтерфейсу.

Для внесення змін у сайт підключіться до Інтернету і запустіть програму FAR. Потім відкрийте панель FTP, натиснувши клавіші Alt+Fl або Alt-вибравши опцію FTP. Якщо сайт, що редагується, є в наведеному списку укажіть на нього курсором і натисніть Enter. Якщо ж потрібного з'єднання в списку немає, натисніть Shift+F4 і в наступному діалозі наберіть дані необхідні для створення нового з'єднання. Після цього програма FTP виведе на свою панель вміст каталогу зазначеного вами сайта. Тепер ви можете працювати з каталогом сайту так, ніби він знаходиться на диску вашого комп'ютера. Користуйтеся звичайними прийомами роботи у Norton Commander і FAR щоб скопіювати або видалити будь-які файли і каталоги.

Після редагування сайту не відключайтеся від Інтернету, а завантажте програму браузера і зайдіть на цей сайт. Перегляньте уважно, як позначилися внесені вами зміни на зовнішньому вигляді і роботі Web-сайту.

12.4. Реєстрація сайту у пошукових системах

Коли сайт зареєстровано на сервері, бажано щоб про його існування дізналися зацікавлені у ньому користувачі мережі. З цією метою різними засобами популяризують адресу і зміст сайту. Дедалі частіше адреси на зразок www.name.server можна побачити на екранах телевізорів, сторінках книг і журналів. Для успішного просування бізнес-сайтів фірм і компаній, а також пізнавальних сайтів, присвячених актуальним питанням науки, освіти,

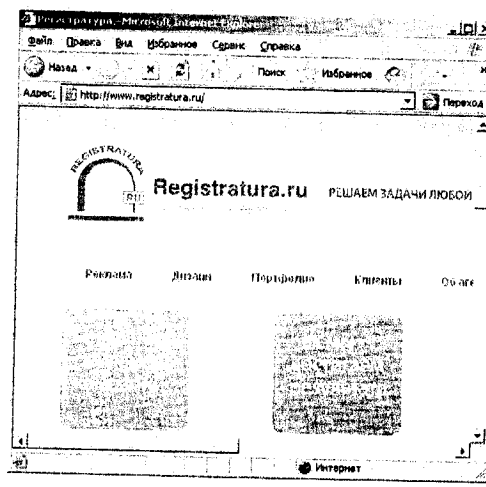


Рис.12.2.**Реєстрація сайту**

медицини тощо, їх рекомендують реєструвати у пошукових системах (каталогах, індексаторах). Пошукові системи призначені для відшукання у мережі сайтів з потрібної тематики чи з наявними у них ключовими словами. Таких систем є декілька десятків і безпосередня реєстрація сайту у кожній зайняла би багато часу.

Під час реєстрації потрібно заповнити анкету-форму, де зазначають адресу сайту, його логічну назву, короткий зміст, ключові слова, дані про автора, зокрема адресу електронної пошти тощо. Деякі каталоги не приймають безплатно на реєстрацію сайту, розташовані на серверах віддалених локальних провайдерів. Тому варто скористатися спеціалізованими системами реєстрації (наприклад, Реєстратура - www. registratura.ru, які, отримавши інформацію про сайт, беруть на себе функції реєстрування сайту в пошукових системах. Не всі послуги, пов'язані з реєстрацією, є безплатні. Після безплатної реєстрації користувач може отримати лист з пропозиціями щодо дальшого платного просування сайту чи проханням розташувати на своєму сайті посилання на реєстратора.

Важливий засіб популяризації сайту - банерна реклама й участь у системах обміну банерами. *Банер* — це графічний об'єкт стандартного розміру 468x60, 100x100,120x60, 88x31, 125x125 пікселів, часто з елементами анімації, який наочно передає зміст сайту і служить гіперпосиланням на нього. Власний банер можна створити засобами комп'ютерної графіки, зокрема, за допомогою програм CoralDraw, PhotoShop та інших або скориставшись конструктором банерів на базі колекції готових шаблонів-картинок.

12.5. Оплата послуг в Інтернет

Для отримання платних послуг (обслуговування в інтернет-магазинах, участь у інтернет-аукціонах тощо) потрібно завести солідну кредитну картку, наприклад Visa, MasterCard, American Express тощо, і

вводити її номер на відповідний запит системи. Аналогічні картки, які надають банки на території СНД, можуть бути не придатними для розрахунків із зарубіжними фірмами. Для отримання платних послуг від конкретного сервера в СНД варто придбати депозитну картку власне цього серверу. Не рекомендується тримати на такій картці значну суму грошей, оскільки теоретично існує ризик вдалої хакерської атаки на слабо захищений сервер, з яким раніше проводилися розрахунки і в пам'яті якого можуть зберігатися дані про картку клієнта.

WebMoney — зручний засіб для всіх видів розрахунків у Мережі.

Відкрити сторінку www.webmoney.ua. Компанія WM Transfer Ltd є власником і адміністратором платіжної системи Webmoney Transfer. Система Webmoney Transfer існує з 1998 року. Розробником програмного забезпечення платіжної системи WebMoney Transfer є ЗАТ «Вычислительные Силы», яке також здійснює технічну підтримку системи WebMoney.

Щоб стати учасником системи WebMoney Transfer, необхідно встановити на своєму персональному комп'ютері, КПК або мобільному телефоні клієнтський інтерфейс, зареєструватися в системі і прийняти її умови, отримавши при цьому WM-ідентифікатор - Ваш унікальний номер. Процес реєстрації також передбачає введення персональних даних і підтвердження їх достовірності через сервіс WM-атестації. Кожен користувач має WM-атестат - цифрове свідоцтво, складене на підставі наданих ним персональних даних.

Усі транзакції в системі є миттєвими і безвідзивними. Залежно від Ваших технічних можливостей, умов роботи або побажань WebMoney безкоштовно надає Вам інструменти для роботи з системою і здійснення платежів:

- KeeperClassic - є окремою програмою, що встановлюється на комп'ютері користувача ;

- Mobile - призначений для проведення розрахунків в режимі реального часу за допомогою мобільних пристроїв;
- Keeper Mini - ще один WebMoney Keeper, дуже простий і зручний у використанні.

Він виконаний у вигляді легкого сайту і підходить для роботи як зі звичайного комп'ютера, так і з будь-якого мобільного пристрою або КПК. Реєстрація і вхід відбувається за логіном і паролем.

Кожен учасник системи має певний бізнес-рівень (BUSINESS LEVEL). BL — це публічна сумарна характеристика рівня ділової активності власника WM -ідентіфікатора, яка обчислюється на основі даних про:

- тривалість активного використання WebMoney Transfer;
- кількість кореспондентів, з якими у користувача були трансакції;
- обсягу проведених трансакцій;
- наявності претензій або позитивних відгуків на адресу користувача.

Значення BL можна побачити в діалоговому вікні програми WM Keeper при роботі з конкретним контрагентом, а також на сторінках сервісів системи.

Система підтримує декілька типів титульних знаків, що забезпечені різними активами і зберігаються на відповідних електронних гаманцях:

- WMU — еквівалент української гривні (гаманець типа U).
- WMG — еквівалент золота (гаманець типа G).

При переказі коштів використовуються однотипні гаманці, а обмін різних титульних знаків здійснюється в обмінних сервісах.

Всі платежі в системі є миттєвими і безвідзивними.

При створенні комерційних сайтів повстає питання: як приймати платежі? Однієї з найбільш популярних у світі платіжних систем є PayPal . Вибір цієї системи часто визначається високою надійністю, простотою відкриття аккаунта й використання. Для відкриття аккаунта досить наявності

кредитної карти або рахунку в американському банку. Одним з основних недоліків часто називають дуже тверду політику безпеки. PayPal не відкриває аккаунти мешканцям СНД , але багато програмістів створюють сайти для закордонних замовників.

Інша форма оплати - за чеком - дійсна на території США і країн Західної Європи. Наявність мережевих платних послуг сприяє поширенню нового виду інтернет-заробітків. Інтернет-магазини, аукціони, сервери і комерційні сайти потребують постійної реклами. Тому їхні власники згідні платити звичайним користувачам просто за формальний перегляд реклами, клацання на рекламних посиланнях, заповнення анкет і форм, підписки, участь у різних акціях, за все, що сприяє залученню нових клієнтів.

12.6. Реклама в Інтернет

Реклама сайтів з'явилася практично одночасно із самою глобальною мережею. Сьогодні найвідомішими й ефективними видами реклами є банерна, контекстна й пошукова реклама.

Банерна реклама є однією з перших у мережі Інтернет. Сьогодні існує два її види:

1. Звичайна банерна реклама.
2. Участь у банерообмінній мережі.

У випадку звичайної банерної реклами, Ви просто оплачуєте показ свого банера на одному або декількох сайтах. Оплата береться, як правило, розраховуючи на 1000 показів, і визначається власником веб-ресурса, на якому розміщається банер.

Банерообмінна мережа – це об'єднання множини сайтів, з метою обміну показами своїх банерів. Принцип роботи банерообмінної мережі досить простий – Ви надаєте місце на своєму веб-ресурсі для показу банерів інших учасників мережі. Залежно від того, як часто на Вашому сайті показуються банери інших учасників, Ви заробляєте покази свого банера на їх-

ній веб-ресурсах. Власник мережі встановлює комісію на показ, що становить від 10 до 15 відсотків показів. Однієї з найбільших банерообмінних мереж у російському просторі Інтернету є мережа RLE, у якій за різним даними, походить від 35 до 55 мільйонів показів. Основною характеристикою ефективності банерної реклами є CRT(click through ratio). Цей показник відбиває співвідношення клацання по банеру до числа його показів. Сьогодні ж банерна реклама як і раніше є досить ефективною, але її популярність з року в рік падає.

Контекстна реклама – це спеціальний вид реклами в мережі Інтернет, що полягає в розміщенні Ваших рекламних оголошень на сторінках веб-ресурсів, близьких по тематиці Вашому сайту. Контекстна реклама є досить ефективної, тому що дозволяє охопити тільки цільову аудиторію, тобто тих користувачів, які так чи інакше зацікавлені у Ваших товарах або послугах.

Пошукова реклама є частковим випадком контекстної. Особливість її полягає в тому, що текст рекламного оголошення розміщається не просто на сторінках веб-ресурсів, а на сторінках пошукових систем, відповідно до введеного користувачем запита. Пошукова реклама є найефективнішої, і відповідно, самою дорогою. На сьогоднішній день самі великі пошукові системи, такі як Яндекс, Рамблер й Google надають рекламодавцям безліч способів і можливостей розміщення пошукової реклами на свої сторінках. Крім цих перерахованих вище видів реклами існує й деякі інші. Найбільш традиційної є реклама веб-ресурсу на спеціалізованих електронних дошках оголошень або рекламних порталів. Будь-яка, навіть найефективніша й дорога реклама Вашого сайту, не дасть тих результатів, які можна одержати, якщо використати її разом із просуванням сайту. Просування охоплює всі аспекти, що впливають на популярність сайту серед користувачів, і на його позиції в пошукових систем. Реклама сайту допомагає ще

більше підсилити ефективність просування, однак необхідно звернути увагу на оптимізацію сайту.

Оптимізувати сайт під вимоги пошукових систем – значить привести його до вигляду, що необхідний для правильної індексації пошуковими машинами. Цю частину роботи реклами в Інтернет ще називають внутрішньою оптимізацією сайту, тому що вона виявляє технічні помилки на сайті й виправляє їх, а також поліпшує внутрішні параметри сайту (теги), необхідні для успішної індексації. Щодня тисячі користувачів Інтернет шукають інформацію в пошукових системах, у тому числі й про Ваш товар. Якщо Вашого сайту немає в результатах пошуку, то більшість Ваших потенційних клієнтів ідуть на сайт Ваших конкурентів. Перед проведенням внутрішньої оптимізації сайту обов'язково проводиться ретельний *аналіз и аудит сайту*, який дає можливість виявити помилки сайту, спланувати стратегію оптимізації й просування.

Просування сайту в Інтернет виконує функції довгострокової кампанії реклами в Інтернет, що ще називають зовнішньою оптимізацією сайту. Для цього проводиться підвищення авторитетності в тематичному співтоваристві Інтернет:

1. відновлення інформації на сторінках сайту, іншими словами – перший крок до просування сайту, його підтримці,
2. обмін статтями й посиланнями з тематичними сайтами,
3. реєстрація в каталогах фірм, товарів і послуг,
4. відновлення повідомлень про сайт у соціальних мережах,
5. моніторинг положення Вашого сайту в пошуковій видачі,
6. моніторинг просування сайтів конкурентів.

Які можливості дає Інтернет реклама.

У нашому розумінні Інтернет реклама - маркетинговий комплекс, що забезпечує досягнення мети. І залежно від мети - комплекс може бути зовсім різним. Властиво ми:

1. Вміємо збільшувати продажі, за рахунок розвинених консалтингових сервісів.
2. Підвищуємо впізнання брендів за рахунок забезпечення масштабної повторюваності марки.
3. Забезпечуємо грандіозне охоплення аудиторії за рахунок партнерства із сотнями тисяч рекламних площ.
4. Реалізуємо складні моделі таргетинга, за рахунок власної технологічної платформи.
5. Ефективніше керуємо рекламними бюджетами наших клієнтів, забезпечуючи ефективну вартість Інтернет реклами.

Ефективна Інтернет реклама починається з формування мети й критеріїв її досягнення й закінчується аналізом результативності проробленої роботи

Дуже ефективною рекламою в Інтернет є пошукове просування сайту, тобто розміщення вашого оголошення (у даному виді послуг воно називається сніппетом) у так званій природній (не рекламної) видачі пошукової системи:

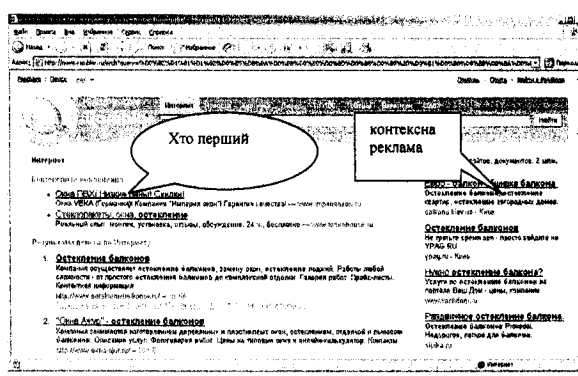


Рис.12.3 **Види реклами в Інтернет**

Плюси пошукового просування:

1. Ви звертаєтеся тільки до цільового аудиторії, яка шукає інформацію за допомогою уведення ключових слів;

2. Ви одержуєте довгостроковий ефект від реклами, тому що вже тільки одна лише підготовка сайту до просування дає свої результати, які залишаються з вами назавжди;
3. Користувачі Інтернет не розглядають пошукове просування, як рекламу, а ви одержуєте рекламу без негативного "рекламного" ефекту;
4. Ціни на такий вид Інтернет реклами порівняно невисокі.

Однак при всіх плюсах пошукової реклами сайту є істотне обмеження, про яке необхідно замислюватися - час необхідний для досягнення високих результатів може досягати декількох місяців, що часто не підходить для короткочасних акцій.

Контекстна реклама сайту представляє собою розміщення в рекламній зоні видачі пошукової системи, та існує за іншими законами, ніж пошук

Плюси контекстної реклами сайту:

1. Ви можете дуже швидко змінювати вміст ваших рекламних оголошень;
2. Можна також швидко налаштувати рекламу на різні регіони й показувати в різний час, що часто дозволяє ефективно використати бюджет;

Однак ключовим обмеженням у ряді галузей є вартість контекстної Інтернет реклами й часто низька комунікабельність порівняно з пошуковим просуванням. Щоб отримати бажаного результату й бути конкурентоздатним в Інтернет необхідно ефективно використовувати інструменти Інтернет маркетингу в комплексі.

Комплексна Інтернет реклама, завжди краще рішення. Ціни на Інтернет рекламу формуються з декількох складових.

1. Створення Інтернет реклами.

2. Вартість розміщення Інтернет реклами звичайно є основною статтею витрат.
3. Вартість послуг консультантів становить відсоток від вартості розміщення.

Запитання для самоконтролю

1. Що таке хостінг?
2. Яка послідовність дій користувача має бути при розміщенні сайту на безкоштовному Web-сервері?
3. Що розуміється під FTP-клієнтом і FTP-сервером?
4. Як розмістити сайт на сервері за допомогою FTP?
5. Як виконується редагування сайту в програмі FAR?
6. Які види реклами ви знаєте?

Предметний покажчик

СПИСОК ЛІТЕРАТУРИ

1. Томсон Л. Разработка Web – приложений на PHP и MySQL : підруч. : пер. с англ. / Л. Томсон, Л. Веллинг. – СПб. : ООО – «ДиаСофтЮп», 2003. – 672 с.
2. Глинський Я. М. Internet. Сервіси, HTML і web дизай : навч. посіб. / Я. М. Глинський, В. А. Ряжська. – Л., 2003. – 192 с.
3. Колисниченко Д. Н. Самоучитель PHP5 : навч. посіб. / Д. Н. Колисниченко. – СПб. : Наука и Техника, 2007. – 640 с.
4. Аргерих Л. Профессиональное PHP программирование : підруч. / Л. Аргерих, В. Чой, Д. Коггсхол. – СПб. : Символ-Плюс, 2003. – 1048 с.
5. Кузнецов М. В. PHP 5/6 : навч. посіб. / М. В. Кузнецов, И. В Симдянов. – СПб. : «БХВ-Петербург», 2009. – 1024 с.
6. Кузнецов М. В. Объектно-ориентированное программирование на PHP : навч. посіб. / М. В. Кузнецов, И. В. Симдянов. – СПб. : «БХВ-Петербург», 2007. – 608 с.
7. Котеров Д. В. PHP 5 в подлиннике: навч. посіб. / Д. В. Котеров, А. А. Костарев. – СПб. : «БХВ-Петербург», 2005. – 1120с.
8. Костарев А. Ф. PHP 5: навч. посіб. / А. Ф. Костарев. – СПб. : «БХВ-Петербург», 2008. – 1104 с.
9. Яргер Р. MySQL и mSQL. Базы данных для небольших предприятий и Интернета: підруч. / Дж. Риз, Т. Кинг. – СПб. : Символ-Плюс, 2000. – 560 с.
10. Хилайер С. Программирование Active Server Pages: підруч. / С. Хилайер, Д. Мизик. – М: «Русская редакция», 1999. – 296 с.

Навчальне видання

Цеслів Ольга Володимирівна

WEB-ПРОГРАМУВАННЯ

Навчальний посібник

В авторській редакції
Надруковано з оригінал-макета замовника

Темплан 2011 р., поз. 1-2-018

Підп. до друку 26.04.2011. Формат 60×84 1/16. Папір офс. Гарнітура Times.
Спосіб друку – ризографія. Ум. друк. арк. 17,2. Обл.-вид. арк. 28,6. Наклад 100 пр. Зам. № 11-115.

НТУУ «КПІ» ВПІ ВПК «Політехніка»
Свідоцтво ДК № 1665 від 28.01.2004 р.
03056, Київ, вул. Політехнічна, 14, корп. 15
тел./факс (044) 406-81-78