Міністерство освіти і науки України
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій

Л. Буяк, А. Мушак, Ю. Семененко, Н. Хома

Методичні вказівки для виконання практичних робіт з дисципліни

«WEB-ДИЗАЙН ТА WEB-ПРОЕКТУВАННЯ»

для студентів денної та заочно-дистанційної форм навчання спеціальностей 028 „Менеджмент соціокультурної діяльності" та 029 «Документознавство та інформаційна діяльність» ступеня вищої освіти „БАКАЛАВР"

Тернопіль
2025

Л. Буяк, А. Мушак, Ю. Семененко, Н. Хома **Методичні вказівки для виконання практичних робіт з дисципліни Web-дизайн та Web-проектування.** Для студентів денної та заочно-дистанційної форм навчання, спеціальностей 028 „Менеджмент соціокультурної діяльності" та 029 „Документознавство та інформаційна діяльність" освітньо-кваліфікаційного рівнів „Бакалавр". – Тернопіль: ЗУНУ, 2025. – 34 с.

Це методичне видання складається із шести практичних робіт. Кожна практична робота містить теоретичний матеріал, який поданий крізь призму навчальних завдань, що становлять другу частину роботи, і завершується переліком контрольних запитань для перевірки рівня засвоєння знань. Усі практичні роботи взаємопов'язані та мають творче спрямування – результатом виконання кожної з них є проходження деякого етапу розробки проекту домашньої сторінки/сайта. Щоправда, завданням такого роду в рамках кожної роботи передують тривіальні завдання для вправляння з метою здобуття навиків.

Автори: ***Леся БУЯК,***
докторка економічних наук, професорка, завідувачка
кафедри економічної кібернетики та інформатики

Андрій МУШАК,
кандидат технічних наук, доцент, доцент
кафедри економічної кібернетики та інформатики

Юрій СЕМЕНЕНКО,
доктор філософії, старший викладач кафедри
економічної кібернетики та інформатики

Надія Хома,
кандидатка фізико-математичних наук, доцентка,
доцентка кафедри економічної кібернетики та інформатики

Рецензенти: ***Сергій МАРТИНЮК,***
кандидат фізико-математичних наук, доцент, доцент
кафедри інформатики та методики її навчання
Тернопільського національного педагогічного
університету імені Володимира Гнатюка

Юрій БАТЬКО,
кандидат технічних наук,
доцент, доцент кафедри коп'ютерної інженерії
Західноукраїнського національного університету

Відповідальна за випуск: ***Леся БУЯК,***
докторка економічних наук, професорка, завідувачка
кафедри економічної кібернетики та інформатики

Затверджено
на засіданні кафедри економічної кібернетики та інформатики,
протокол №6 від 14 січня 2025 р.

Зміст

ПЕРЕДМОВА

Сьогодні документознавчу діяльність неможливо уявити без нових інформаційних технологій. Ними просякнуті найвіддаленіші елементи аналізованої сфери людських знань. Окрім цього сучасні інформаційні технології привносять такі можливості у документознавство, про які й не здогадувались при класичному підході. Мова, зокрема, йде про можливість донесення інформації найширшій аудиторії користувачів, послуговуючись системою World Wide Web. Відомо, що класичним засобом для реалізації цього завдання є мова розмітки гіпертексту – HTML. Через те постає важливим оволодіння знаннями, навиками та уміннями подання інформації в аналізованому форматі, що є ключовим моментом розглядуваної технології. Натомість другорядними (проте не менш важливими) є, наприклад, питання про розміщення створених матеріалів з метою наданя доступу до них віддалених користувачів і пришвидшення побудови веб-сторінок послуговуючись HTML-редакторами.

Нашою стратегічною дидактичною метою є навчання студентів прийомам написання веб-сторінок за допомогою мінімального набору засобів: найпростішого текстового редактора, для створення ASCII-текстових файлів у форматі HTML та браузера для пергляду та відлагодження отриманих файлів. Саме така „ручна" технологія дозволяє ґрунтовно вивчити теги мови та знати її сисинтаксис. Щоправда цей підхід вимагає значних затрат часу. В ролі найпростішого редактора пропонуємо послуговуватись програмою Notepad, яка входить у комплект поставки операційної системи Windows, а в ролі браузерів – Google Chrome чи Mozilla Firefox, оскільки нині вони є найпопулярнішими.

Виконуючи більшість практичних робіт послідовно рухаються у двох напрямах: спочатку вирішують запропоновані завдання вправляючись на відокремлених файлах, а далі – працюють над проектом власної домашньої сторінки (сайта), переносячи отримані у першому напрямі навики у роботу з чітко вираженим творчим спрямуванням.

Ці методичні вказівки є частиною методичного забезпечення навчального курсу „Web-дизайн та Web-проектування", а також може бути використаний в інших споріднених курсах.

Метою практикуму є оволодіння студентами основами подання інформації на Вебі в HTML-форматі. Практикум містить шість практичних робіт, які охоплюють такі теми:

- Наповнення веб-сторінок текстовою інформацією
- Монтування гіперпосилань та рисунків
- Конструювання списків
- Побудова таблиць
- Послуговування фреймами
- Мультимедіа-елементи на Вебі: аудіо- та відеокліпи

Кожна практична робота містить теоретичні відомості, завдання і контрольні запитання. Теоретичний матеріал окремих робіт підкріплений екранними копіями програм або їх фрагментами, з якими студенти зустрінуться під час виконання практичної роботи. Студенти виконують відповідні запропоновані завдання до кожної практичної, окрім першої, по варіантах. Останній додаток подає приклад розробленої домашньої сторінки.

ПРАКТИЧНА РОБОТА №1
НАПОВНЕННЯ ВЕБ-СТОРІНОК ТЕКСТОВОЮ ІНФОРМАЦІЄЮ

ТЕОРЕТИЧНІ ВІДОМОСТІ

У будь-якій успішній публікації, нехай це навіть глибокодумний трактат, текст повинен бути організований привабливим та ефектним чином. Перетворення записів у привабливий документ – це саме те, що дається HTML найкраще. Ця мова надає велику кількість способів для впорядкування тексту та виразного викладу його основного змісту. Окрм цього, вони допомагають так структурувати документ, що кожне слово легко знаходить шлях до читача.

Будь-який веб-документ, підготовлений за допомогою мови розмітки гіпертексту – HTML, має такий каркас[1]:

```
<html>
 <head><title>...</title>
 ...
 </head>

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

Розглядають два розділи між відкриваючим (<html>) та закриваючим (</html>) тегами для задання початку та кінця документа, відповідно. У першому розділі, який є головою аналізованого документа (теги <head> і </head>) зазначають службову інформацію, яка не є умістом веб-документа. Як відомо, вміст останнього відображається в області перегляду браузера. Єдиний елемент голови документа, який стосується умісту – це назва документа, яка міститься між відкриваючим і закриваючим тегами <title>...</title>. Назва документа відображатиметься у самому верхньому рядку вікна браузера. Основний розділ коду будь-якого HTML-файлу – це його тіло (<body>...</body>). Саме у цьому розділі подають, зокрема, текстову інформацію, яка міститиметься на веб-сторінці, а також тут її структурують і форматують.

Браузер розміщує текст на екрані по всій ширині вікна, переходячи на наступний рядок, після досягнення його краю. Достатньо зробити вікно ширшим, і текст автоматично розтечеться, заповнюючи рядки, які стали довшими. Якщо стиснути вікно, то текст витиснеться униз.

Проте на відміну від більшості текстових процесорів, в HTML явним чином застосовують теги абзаца <p> та кінця рядка
 для управління вирівнюванням і потоком тексту. Повернення каретки, хоча і дуже корисне для зручності читання, зазвичай ігнорується браузером, і автори веб-сторінок повинні послуговуватися тегом
, щоб явно вказати кінець рядка. Тег <p>, який також викликає перехід на наступний рядок, несе у собі додатковий зміст окрім повернення каретки.

Тег <p> сигналізує про початок нового абзаца. Зустрівши тег нового абзаца <p>, браузер зазвичай вставляє у документ перед його початком порожній рядок, дещо збільшений по висоті ніж звичайний. Далі браузер збирає в цей абзац всі слова і,

[1] Веб-документи для організації фреймової структури замість тега <body> містять тег <frameset>. Детальніше про це мова йтиме у п’ятій лабораторній роботі.

якщо вони присутні, вмонтовані зображення, опускає всі пропуски перед ними та в їх кінці (але не пропуски між словами, звичайно) і символи повернення каретки, які знаходяться у первинному тексті. Далі програма заповнює отриманою послідовністю слів та зображень своє вікно перегляду, автоматично генеруючи переходи на новий рядок після досягнення краю вікна.

Браузер уміє автоматично переносити довгі слова з рядка на рядок і „повністю вирівнювати" абзац, так що текст буде рівномірно розтягнутий між лівим та правим краєм вікна.

В результаті при складанні документів не потрібно турбуватися ні про довжину, ні про розбиття рядка.

Більшість браузерів автоматично вирівнюють текст по лівому краю. Щоб змінити такий стан справ, послуговуються атрибутом `align` тега `<p>`. В ролі значення цього атрибута вказують один із чотирьох видів вирівнювання – `left`, `right`, `center` або `justify`. Зокрема, при `align=justify` маємо вирівнювання одночасно як з правого, так і з лівого боку.

Вирівнювання зберігається до появи або наступного тега `<p>`, або поточного закриваючого тега `</p>`.

Тег `<br>` перериває нормальний процес заповнення рядків і розстановки переносів тексту абзаців в документі. У цього тега немає закриваючого. Він просто помічає у потоці тексту точку, з якої повинен починатися новий рядок.

Тег `<center>` – один із тих, чия дія є очевидною, – його уміст вирівнюється у вікні браузера по центру.

Користувачі витрачають чимало часу на читання з екрану. Довгий потік тексту, який не переривається заголовками, підзаголовками, ковзає перед очима, затуманюючи голову, не говорячи вже про неможливість при швидкому перегляді виявити тему, яка зацікавила.

Суцільний текст, потрібно розбивати на декілька невеликих розділів, забезпечуючи їх заголовками. Існує шість рівнів заголовків. Шість заголовкових тегів, які записують `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>` та `<h6>`, подають ієрархію заголовків від вищого (`<h1>`) до нижнього (`<h6>`) рівня.

Традиційно автори послуговуються заголовком `<h1>` для назви документа, `<h2>` – для назв розділів і т. д. Закриваючі теги для заголовків обов'язково присутні.

Вирівнювання заголовків за замовчуванням відповідає значенню `left` атрибута `align`. Окрім значення `left` цьому атрибуту можна присвоїти значення `center` і `right`.

До фізичних стилів тексту відносять, зокрема, напівжирний, курсив, підкреслений.

Тег `<b>` вимагає відображення напівжирним шрифтом символа чи тексту, який міститься між ним і його закриваючим тегом `</b>`.

Тег `<i>` пропонує браузеру відобразити текст, який міститься між ним та відповідним закриваючим тегом `</i>` курсивом.

Тег `<u>` говорить браузеру, щоб той підкреслив текст, який міститься між `<u>` та `</u>`. Підкреслення полягає у тому, що під текстом, включаючи пропуски між словами та розділові знаки, проводиться суцільна лінія.

Кожен із цих тегів потребує відповідного закриваючого тега. Допустимим є комбінування фізичних стилів.

Тег `<font>` дозволяє змінювати розмір, стиль і колір тексту. Значенням атрибуту `size` є або один із віртуальних розмірів (1-7), або величина зі знаком „плюс" або „мінус", що розглядається як відносний розмір, який браузер додає чи віднімає від основного розміру шрифту. Атрибут `color`, який використовується з тегом `<font>`, встановлює колір тексту, який міститься у тезі. Значення атрибуту може бути введене одним із двох способів: як встановлення червоної, зеленої та синьої (RGB) компоненти кольору або як стандартна назва кольору. За допомогою атрибута `face`

тега <font> змінюють стиль шрифту в текстових уривках. Значення атрибута face – це назва шрифту. Ось приклад використання тега <font> із розглянутими атрибутами:

```
<font size=+1 color=lightblue face=”Arial”>Файне місто
Тернопіль!</font>.
```

Послуговуючись атрибутом bgcolor тега <body> змінюють прийнятий за замовчуванням колір фону (тла) веб-документа. Як і для атрибута color тега <font>, обов’язкове значення bgcolor може бути виражене одним із двох способів – за допомогою вказання червоної, зеленої та синьої (RGB) компоненти вибраного кольору або його стандартної назви, наприклад,

```
<body bgcolor=”#FFFF00”>.
```

Якщо однотонний фон не влаштовує користувача, то за допомогою атрибута background тега <body> можна помістити на його місце зображення. Обов’язковим значенням атрибута background є URL зображення. Браузер автоматично повторює (у вигляді мозаїки) рисунок по вертикалі та по горизонталі, заповнюючи область перегляду вікна браузера. Ось праклад послуговування атрибутом background:

```
<body background=”Images/Smile.gif”>.
```

ЗАВДАННЯ

1. Створити робочий каталог. При потребі утворити підкаталоги для грамотної організації інформації.
2. Знайти у Мережі домашні сторінки інших людей[2]. Проаналізувати/оцінити їх. Ті, які найбільше сподобались, зберегти у робочому каталозі.
3. На основі чужих домашніх сторінок, розробити на папері проект власної домашньої сторінки.
4. Створити каркас веб-сторінки. Зберегти його у файлі, ім’я якого є Вашим прізвищем.
5. Наповнити домашню сторінку текстовим матеріалом та відформатувати його на свій розсуд.
6. Закрити вікна. Закінчити роботу.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Яким є каркас будь-якої веб-сторінки? Призначення розділів голови і тіла коду HTML-сторінки.
2. Яка відмінність в інтерпретації браузером тегів <p> і
?
3. Як вирівняти текст в абзаці одночасно з лівого та правого краю?
4. <h6> – це тег для задання заголовка найнижчого чи найвищого рівня?
5. Назвати найуживаніші теги для задання фізичних стилів тексту.
6. Яке призначення тега <font>?
7. Як задати колір фону сторінки та встановити фоновий рисунок?

[2] Задля цього виконуючи пошук послуговуються, зокрема, словосполученнями „домашня сторінка”, „домашняя страница”, „home page”.

ПРАКТИЧНА РОБОТА №2
МОНТУВАННЯ ГІПЕРПОСИЛАНЬ І РИСУНКІВ

ТЕОРЕТИЧНІ ВІДОМОСТІ

В результаті виконання попередньої лабораторної роботи був створений окремий веб-документ, котрий є ізольованим об'єктом[1]. При цьому акцент був зроблений на елементах мови, якими послуговуються для структурування та форматування матеріалу. Проте істинна сила HTML полягає у здатності об'єднувати зібрання документів у повні бібліотеки та пов'язувати ці бібліотеки з іншими збірками, які розкидані по усьому світу. Подібно до того, як користувач може у значному ступені керувати виглядом документа на своєму екрані, так і гіперпосилання дозволяють йому керувати черговістю появи документів.

Хоча основою більшості документів є текст, приправа із доречно вставлених зображень і загалом, мультимедійних елементів робить матеріал ошатнішим та привабливішим. Можливість включати у текстовий документ зображеня або у вигляді вбудованих компонентів (упроваджених зображень), або окремих документів, які можуть бути завантажені за допомогою відповідного гіперпосилання, або у вигляді фону документа, є однією із найпринадніших рис HTML.

Тег `<img>` дозволяє вставити у текстовий потік документа графічне зображення. Ні до, ні після аналізованого тега не потрібно обов'язкової прсутності кінця абзаца чи переходу на новий рядок, так що зображення може бути поміщене буквально „в лінію" з текстом та іншим умістом документа.

Атрибут `src` є обов'язковим для тега `<img>`. Значенням цього атрибута є URL файлу, який містить зображення, абсолютний чи вказаний відносно документа, який на нього посилається.

Можна почасти керувати вирівнюванням зображень відносно оточуючого тексту за допомогою атрибута `align` тега `<img>`. Стандарт HTML визначає п'ять значень для атрибута `align`: `left`, `right`, `top`, `middle`, `bottom`.

Браузери зазвичай беруть зображення, які у той же час є гіперпосиланнями (розташовані у тезі `<a>`, про що мова йтиме далі), в кольорову рамку товщиною два пікселі, даючи цим читачеві знати, що таке зображення можна вибрати та відвідати асоційований з ним документ. Атрибутом `border` та його значенням, яке визначає товщину рамки в пікселях, послуговуються щоб забрати її (`border=0`) чи зробити товстішою.

Для задання розмірів зображення послуговуються атрибутами `height` і `width` тега `<img>`. Обидва атрибути приймають цілі значення, які встановлюють розміри зображення у пікселях. Порядок, у якому вони з'являються в тезі `<img>` значення не має. Ось приклад послуговування тегом `<img>`:

```
<img src="Images/car.jpg" align=right weight=100 height=55>
```

Задля привнесення у веб-документи гіперпосилань на інші джерела та для створення у документах ідентифікаторів фрагментів використовують тег `<a>`.

Найчастіше тег `<a>` застосовується з атрибутом `href` для створення гіперпосилань до інших місць у тому ж документі або до інших документів. У таких

[1] Щоправда, це не зовсім так, оскільки реалізуючи у веб-документі фоновий рисунок, доводиться під'єднувати до HTML-файлу графічний файл. Цей єдиний виняток свідомо відносимо до першої лабораторної роботи, оскільки він з точки зору розробника-дизайнера швидше стосується форматування текстового матеріалу сторінки, аніж монтажу графіки загалом. Звісно, що розробник, який будує модель взаємодії аналізованого документа з іншими об'єктами думає по-іншому.

випадках документ, у якому розташоване гіперпосилання, – це джерело посилання, а значення атрибута `href`, URL, – це мета.

Інший спосіб використання тега `<a>` полягає у застосуванні атрибута `name`, щоб відмітити в документі мету гіперпосилання або ідентифікатор фрагмента.

Між тегом `<a>` та його обов'язковим закриваючим тегом можна розміщувати тільки звичайний текст, переходи на новий рядок, зображення та заголовки. Браузер відтворює всі це елементи звичайним чином, але додає при цьому який-небудь спеціальний ефект, даючи читачеві зрозуміти, що він має справу з гіперпосиланням на інший документ.

Атрибут `href` визначає URL мети гіперпосилання. Значенням цього атрибута є допустимий URL документа, абсолютний чи відносний, який включає ідентифікатор фрагмента. Якщо користувач вибирає уміст тега `<a>`, браузер намагається отримати та відобразити документ, вказаний в атрибуті `href`. Ось приклад використання тега `<a>` з атрибутом `href`:

```
<a href="http://www.example.com">Веб-сайт фірми Альпійське
Вікно</a>
```

Атрибут `name` використовують з тегом `<a>` для створення в документі ідентифікатора фрагмента. Одного разу сформований, він стає потенційною метою гіперпосилання. Значення атрибута `name` – це будь-який рядок символів, поміщених у лапки. Наприклад, нехай у коді веб-документа (файл kafedra.html) маємо такий запис: `<a name="Schedule">`. Тоді, наприклад, ось таке вибране користувачем посилання в деякому іншому файлі `<a href="kafedra.html#Schedule">Розклад занять працівників кафедри </a>`відразу перенесе його до фрагмента, ім'я якого `Schedule`.

Уміст тега `<a>` з атрибутом `name` відображається звичайним чином, без якихось спеціальних ефектів. Говорячи формально, немає необхідності поміщати що-небудь із умісту документа в тег `<a>` з атрибутом `name`, оскільки він помічає лише місце в документі.

ЗАВДАННЯ

1. Подати в HTML-форматі запропоновані, згідно з варіантом [2], речення, що містять гіперпосилання і рисунки[3].
2. Привнести, відповідно до змісту, гіперпосилання та рисунки у розроблюваний проект домашньої сторінки/сайта[4]. Декотрі із рисунків повинні бути гіперпосиланнями на інші веб-сторінки. Організувати за допомогою гіперпосилань найпростішу навігаційну систему на сторінці. А саме, на початку сторінки розмістити своєрідний зміст – назви структурних частин сторінки. Кожна назва є гіперпосиланням на відповідний розділ сторінки.
3. Закрити вікна. Закінчити роботу.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Які є можливості привнесення у текстовий документ зображень? 2. Характеристики атрибутів тега `<img>`. 3. Як забрати рамку довкола зображеня у тому випадку, коли воно є гіперпосиланням? 4. Як організувати гіперпосилання в рамках документа на його фрагменти?

---

[2] Варіанти подані у додатку А (див. кольорову вклейку).

[3] Вважатимемо, що текст, який поданий синім кольором та підкреслений, є гіперпосиланням. Рисунки, які обрамлені синьою рамкою теж є гіперпосиланнями. Зовнішні ресурси (власне їх URL'і) підібрати такі, щоб вони найкраще за змістом відповідали висловленим, за допомогою запропонованих речень, думкам.

[4] Інформацію на розроблюваній веб-сторінці доцільно структурувати. А відтак, варто перейти від проекту поодинокої сторінки (один .html-файл) до набору сторінок, кожна з яких є структурою частиною проекту. Такий підхід є особливо прийнятливим тоді, коли структурні частини сторінки мають значний об'єм. В результаті працюватимемо над проектом сайта. В цьому випадку організовуємо, зокрема, посилання з початкової сторінки сайта на інші сторінки, а також з кожної із них на початкову.

ПРАКТИЧНА РОБОТА №3
КОНСТРУЮВАННЯ СПИСКІВ

ТЕОРЕТИЧНІ ВІДОМОСТІ

Подання інформації у зручнішому вигляді – це єдина найважливіша позитивна якість HTML. Чудова колекція текстових стилів і засобів форматування, які входять у склад цієї мови, допомагають організувати інформацію у документи, які читачі швидко переглядають, легко розуміють та роблять витяги, можливо, застосовуючи автоматичні способи обробки.

Окрім можливості прикрасити текст за допомогою спеціалізованих текстових тегів, HTML надає також багатий набір знарядь, які дозволяють впорядкувати уміст документа у вигляді форматованих списків. Краса списків полягає у їх простоті. Вони базуються на загальновідомих зразках, з якими зустрічаємося щоденно. Списки є зручними способами організації інформації. Усі вони допомагають краще засвоїти її при швидкому перегляді, а також знайти та видобути із веб-сторінок потрібні відомості.

Подібно переліку покупок або зданих у пральню речей, невпорядкований список є сукупністю однорідних елементів, між якими немає відношень порядку чи слідування. Найрозповсюдженіші у мережі невпорядковані списки – це збірки гіперпосилань на інші документи. Невпорядкований список можна читати у якому завгодно порядку.

Тег `<ul>` сигналізує браузерові, що усе, котре слідує за ним, аж до закриваючого тега `</ul>` є невпорядкованим списком елементів. Всередині невпорядкованого списку кожен елемент помічається тегом `<li>`. Окрім цього, у список може входити практично будь-який HTML-уміст, включаючи інші списки, текст і мультимедійні елементи.

При відображенні браузер зазвичай передує кожен елемент списку спеціальним маркером і, розташовуючи кожен елемент з нового рядка, застосовує до нього деяке форматування, наприклад, відступ від лівого краю вікна.

Ось фрагмент коду, який браузер інтерпретуватиме у невпорядкований список:

```
Україна успадкувала від СРСР розгалужену систему освітніх та
концертних музичних організацій, що знаходяться у веденні
Міністерства культури і туризму. Зокрема це:
<ul>
 <li>оперні театри у Києві, Харкові, Львові, Одесі,
Дніпропетровську, Донецьку</li>
 <li>театри музичної комедії в Харкові та Одесі, а також театр
оперети в Києві</li>
 <li>Дитячий музичний театр у Києві</li>
</ul>
```

Упорядкованими списками послуговуються тоді, коли порядок елементів у них має істотне значення. Іструкція з установки пральної машини, зміст монографії чи список виносок і приміток – усе це хороші приклади упорядкованих списків.

Типовий браузер форматує уміст упорядкованого списку точно так само, як і неупорядкованого, лише з тим винятком, що елементи списку обладнуються номерами замість маркерів. Нумерація починається з одиниці і кожен наступний відмічений тегом `<li>` елемент списку отримує номер на одиницю більший, ніж попередній. Ось приклад деякого HTML-коду, який інтерпретуватиметься браузером у вигляді упорядкованого списку:

```
Загалом, <b>виготовлення хлібобулочних виробів поділяється на такі
операції</b>:
```

```
<ol>
 <li>підготовка сировини до виробництва - перемішування,
просіювання борошна, розчинення масла, дріжджів, розчинення солі у
воді, підготовка заквасок, замішування опари;</li>
 <li>замішування тіста, його бродіння;</li>
 <li>формування хлібобулочних виробів - поділювання, округлювання,
закатування тіста;</li>
 <li>вистоювання (ферментація), щоб тісто остаточно збільшилося в
об’ємі (стало пухкішим)</li>
 <li>випікання;</li>
 <li>охолоджування;</li>
 <li>нарізання, пакування, підготовка до транспортування.</li>
</ol>
```

Із попереднього викладу стає очевидним, що тег `<li>` визначає елемент списку. Цей універсальний тег застосовують як у впорядкованих (`<ol>`), так і у невпорядкованих (`<ul>`) списках.

У впорядкованих та невпорядкованих списках за тегом `<li>` може слідувати практично усе що завгодно, включаючи інші списки та послідовності абзаців. Браузер, якщо він взагалі використовує відступи, зазвичай послідовно застосовує їх до кожного вкладеного списка та уміст будь-якого елемента такої ієрархії вирівнюється по межі найглибшого вкладення для даного елемента.

У розглянуті списки можна вкладати інші списки. Відступи для кожного вкладення списків накопичуються, так що потрібно потурбуватися, щоб глибина вкладень не була занадто великою, – уміст списків швидко перетворюється у вузьку смужку вздовж правого краю області перегляду вікна браузера.

ЗАВДАННЯ

1. Подати в HTML-форматі списки, визначені згідно з варіантом[1].
2. Привнести, відповідно до змісту, списки у розроблюваний проект домашньої сторінки/сайта. Щонайменше один із них повинен бути мішаним.
3. Закрити вікна. Закінчити роботу.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Які види списків дозволяє реалізувати HTML?
2. Яка природа невпорядкованого та впорядкованого списку?
3. Що може бути елементом списку?
4. Яким є код-каркас для невпорядкованого та впорядкованого списку?
5. Яким є код-каркас для мішаного списку?

[1] Варіанти подані у додатку Б (див. кольорову вклейку).

ПРАКТИЧНА РОБОТА №4
ПОБУДОВА ТАБЛИЦЬ

ТЕОРЕТИЧНІ ВІДОМОСТІ

З усіх розширень, котрі знайшли своє місце в HTML, жодне так не віталося, як таблиці. Окрім послуговування ними для подання даних, таблиці також відіграють важливу роль у керуванні макетом документа. Творче використання таблиць може непогано оживити похмурий документ.

Стандартна модель таблиць є достатньо простою: таблиця – це набір слів і чисел, вибудуваних у рядки та стовбці комірок. Більшість комірок містять дані, інші – заголовки рядків та стовбців, які описують дані.

Спроектувавши таблицю поміщають усі її елементи між тегом `<table>` та його закриваючим тегом `</table>`. Елементи таблиці, такі як дані, заголовки рядків та стовбців, підписи, мають свої теги розмітки. Рухаючись зверху вниз та зліва направо, послідовно описують заголовок і дані в кожній комірці стовбців таблиці, і, таким чином, визначається уся таблиця, рядок за рядком.

У комірці таблиці можна розмістити майже усе, що може міститися у тілі HTML-документа, включаючи зображення, форми, лінійки, заголовки та навіть інші таблиці. Браузер поводиться з кожною коміркою як з цілим вікном, заповнюючи його простір потоком умісту, але з деякими обмеженнями та доповненнями у тому, що стосується форматування.

Усе різноманіття таблиць можна створити усього лише п’ятьма тегами: тегом `<table>`, який виділяє таблицю та її елементи із умісту тіла документа; тегом `<tr>`, який визначає рядок таблиці; тегами `<th>` і `<td>`, які визначають заголовки у таблиці та комірки даних, відповідно, і тегом `<caption>`, який визначає назву таблиці.

Браузер, зустрівши тег `<table>`, зупиняє потік тексту, переводить рядок, починає у новому рядку таблицю, а після її завершення відновлює потік тексту з наступного за таблицею рядка. Єдино допустимими в тезі `<table>` є один чи декілька тегів `<tr>`, які визначають кожен рядок таблиці, а також різні теги розділів таблиці, які не розглядатимемо в рамках викладу навчального матеріалу. Необов’язковий у тезі `<table>` атрибут `border` наказує браузеру провести лінії довкола таблиці, її рядків та комірок. За замовчуванням обрамлення відсутнє. Присвоївши аналізованому атрибуту ціле значення, досягають трьохвимірного подання опуклої рамки, шириною, яка дорівнює вказаній кількості пікселів.

Для задання рядка у таблиці послуговуються тегом `<tr>`. У цьому тезі розміщують одну чи декілька комірок, які містять заголовки (позначаються тегом `<th>`), чи такі, що містять дані (позначаються тегами `<td>`). В усіх рядках таблиці стільки ж комірок, скільки у найдовшому рядку. Браузер автоматично додає порожні комірки у рядки з меншою кількістю комірок. Атрибут `bgcolor` у тезі `<tr>` встановлює колір фону для усього рядка. Значенням аналізованого атрибуту зазвичай є стандартна назва кольору.

Теги `<th>` і `<td>` входять у тег таблиці `<tr>` та визначають комірки рядка і їх уміст. Обидва теги діють подібним чином, єдина між ними відмінність полягає у тому, що браузер відображає текст заголовку – вважається, що він подає назву чи опис даних у таблиці – жирним шрифтом і що прийняте за замовчуванням вирівнювання умісту заголовку може відрізнятися від такого для даних. Дані, зазвичай, за замовчуванням вирівнюються з лівого боку, а заголовки – по центру. Атрибути, які застосовують до поодинокої комірки заголовка чи даних мають пріоритет перед своїми аналогами, загальними для комірок рядка. Існують також спеціальні атрибути, які визначають кількість рядків та стовбців у таблиці, яку можуть охоплювати комірки.

Умістом тегів <th> і <td> може бути усе, що можна помістити в тіло документа, включаючи текст, зображення, форми і т. ін., навіть інші таблиці. Браузер автоматично створює таблицю достатньо великою – і по горизонталі і по вертикалі, – щоб відобразити уміст усіх комірок. Якщо у деякому рядку виявилось менше комірок, ніж в інших комірках, браузер розміщує порожні комірки у кінець рядка, заповнюючи рядок. Якщо потрібно створити порожню комірку на початку чи в середині рядка, щоб позначити, наприклад, відсутні дані, створюють комірку даних чи заголовку без умісту.

Атрибут align тегів <th> і <td> керує горизонтальним вирівнюванням умісту тільки у поточній комірці. Цьому атрибуту можна присвоїти значення left, right, center, наказуючи браузеру вирівняти уміст комірки по лівому, чи правому краю комірки, чи по її центру, відповідно.

Існує практика формування заголовків таблиці, які описують декілька розташованих під ними стовбців. Атрибутом colspan послуговуються у тегах заголовкових комірок та комірок даних, щоб об'єднати декілька комірок у рядку. Значення цього атрибуту дорівнює цілій кількості комірок рядка, котрі потрібно об'єднати. Наприклад:

```
<td colspan=3>
```

наказує браузерові створити комірку, яка займатиме той же простір по горизонталі, як три комірки в рядках, розташованих вище або нижче. Браузер розмістить уміст такої комірки на всьому виділеному просторі.

Подібно тому, як атрибут colspan об’єднує декілька комірок в одному рядку, атрибут rowspan об’єднує комірку з декількома, які знаходяться під нею. Останній атрибут включають в теги <th> і <td> і йому присвоюють значення, що дорівнює кількості рядків, які охоплюються цією коміркою. Створена комірка займатиме простір поточної комірки та відповідної кількості комірок під нею. Браузер розміщуватиме уміст об'єднаної комірки по усьому її просторі. Наприклад:

```
<td rowspan=3>
```

створює комірку, яка об’єднує поточну та ще два рядки, які лежать нижче.

Можна розповсюдити одну комірку одночасно по вертикалі і по горизонталі, включивши в теги заголовка таблиці чи даних таблиці атрибути colspan і rowspan разом. Наприклад:

```
<th colspan=3 rowspan=4>
```

створює заголовочну комірку, яка перетинає три стовбця і чотири рядки, містить поточну комірку, дві комірки справа і три комірки під поточною коміркою. Браузер вирівнює уміст такої комірки по усьому її просторі.

Для окремої комірки також можна змінити колір фону за допомогою атрибута bgcolor. Значенням цього атрибуту, зазвичай, є стандартна назва кольору.

Таблиця, як правило, містить заголовок, котрий пояснює її уміст, через те популярні браузери інтерпретують тег заголовку таблиці. Розробники веб-документів розміщують тег <caption> безпосередньо після тега <table>, але можна розміщувати його практично де завгодно всередині таблиці між тегами рядків. Тег заголовку може включати будь-який уміст тіла документа, так само як комірка таблиці.

Розглянемо приклад побудови таблиці.

```
<table border=2>
 <caption>Успішність студентів, котрі вивчають математику,
 фізику та інформатику</caption>
```

```
  <tr>
   <td colspan=2 rowspan=2></td>
   <th colspan=3 align=center>Успішність, %</th>
  </tr>

  <tr>
   <th>Математика</th>
   <th>Фізика</th>
   <th>Інформатика</th>
  </tr>

  <tr align=center>
   <th rowspan=2>Студенти</th>
   <th>Юнаки</th>
   <td>93</td>
   <td>91</td>
   <td>95</td>
  </tr>

  <tr align=center>
   <th>Дівчата</th>
   <td>90</td>
   <td>82</td>
   <td>97</td>
  </tr>
 </table>
```

Браузер інтерпретує цей код так, як показано на рис. 4.1.

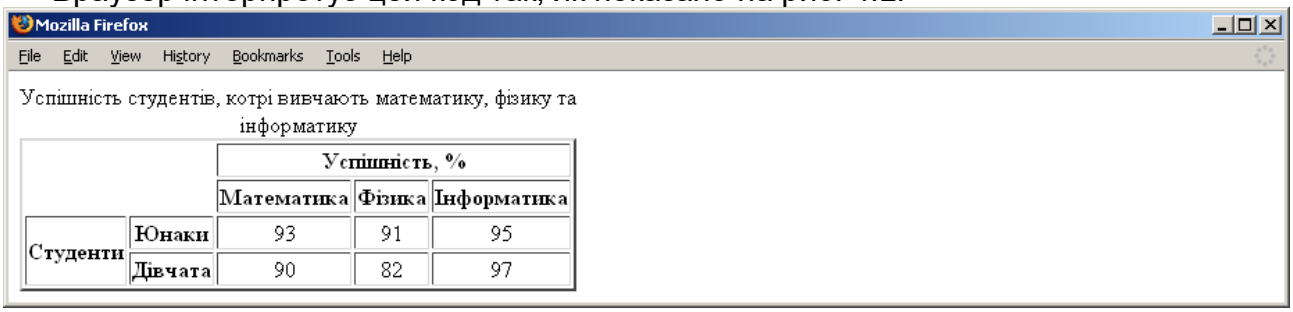

Рис. 4.1. Таблиця із заголовком у браузері Mozilla Firefox

ЗАВДАННЯ

1. Подати в HTML-форматі таблиці, визначені згідно з варіантом[1].
2. Привнести, відповідно до змісту, таблицю (таблиці) у розроблюваний проект домашньої сторінки/сайта.
3. Закрити вікна. Закінчити роботу.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. З якою метою послуговуються таблицями на Вебі? 2. Чи можна в деякій комірці таблиці розмістити іншу таблицю? 3. Якими тегами послуговуються для побудови таблиць? 4. Як змінити товщину ліній обрамлення таблиці? 5. Як розповсюдити одну комірку таблиці одночасно по вертикалі і по горизонталі? 6. Як вирівняти уміст комірки таблиці? 7. Як задати колір фону комірки, рядка? 8. Яким тегом формують заголовок таблиці?

[1] Варіанти подані у додатку В (див. кольорову вклейку).

ПРАКТИЧНА РОБОТА №5
ПОСЛУГОВУВАННЯ ФРЕЙМАМИ

ТЕОРЕТИЧНІ ВІДОМОСТІ

Починаючи з браузера Netscape Navigator 2.0, HTML-автори отримали можливість розділяти основне вікно браузера на декілька незалежних кадрів-вікон (фреймів), у кожному з яких відображається свій документ. Одержувалось щось наподобі стіни з моніторів в операторській кімнаті на телебаченні. Фрейми, які зараз же завоювали популярність, були взяті на озброєння (та розширені) компанією Microsoft для браузера Internet Explorer. Тепер вони входять у стандарт HTML.

Для формування HTML-документів, які містять фрейми, послуговуються двома тегами: `<frameset>` і `<frame>`.

Набір кадрів – це просто сукупність кадрів, які покривають область перегляду браузера. Атрибути встановлення стовбців та рядків в тезі <frameset> дозволяють задавати кількість та початкові розміри стовбців і рядків кадрів. Тегом <frame> встановлюється, який документ – HTML чи інший – спочатку виводиться у кадрі. У цьому тезі можна задати ім'я фрейма, який використовуватиметься для уміщення цільових документів, отримуваних за гіперпосиланнями.

Розглянемо код деякого HTML-документа з фреймами, який відображається браузером так, як подано на рис. 5.1.

```
<html>
<head>
<title>Frames Example</title>
</head>
<frameset rows="60%,*" cols="65%,20%,*">
 <frame src="frame1.html">
 <frame src="frame2.html">
 <frame src="frame3.html" name="fill_me">
 <frame src="frame4.html" scrolling=yes>
 <frame src="frame5.html">
 <frame src="frame6.html">
</frameset>
</html>
```

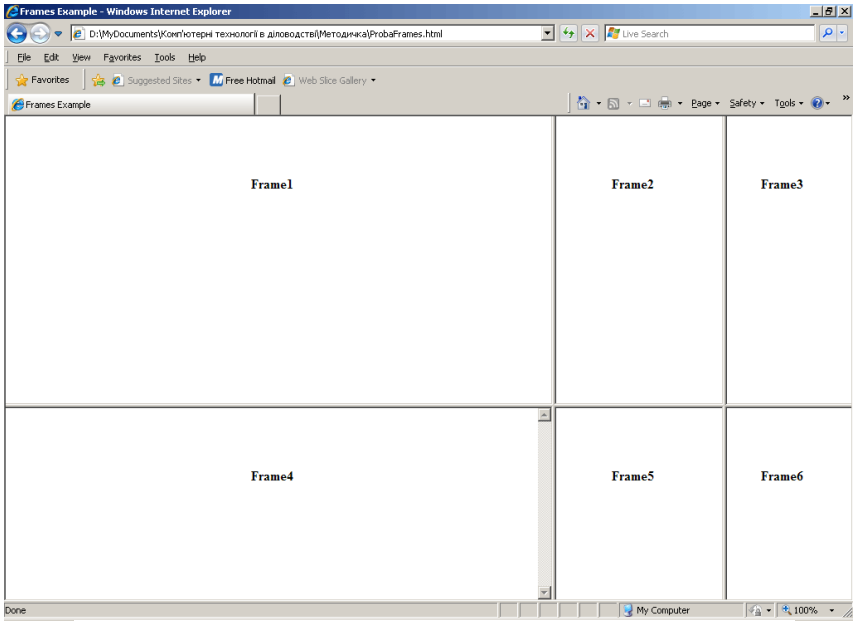

Рис. 5.1. Проста фреймова структура деякого HTML-документа

Браузер заповнює кадри, які входять у набір, зліва направо у порядку слідування рядків. Окрім цього, оскільки задано відповідний атрибут в коді документа, четвертий кадр буде обладнаний лінійкою прокрутки, незалежно від того, чи поміщається уміст кадра у ньому цілком чи ні. (Лінійка прокрутки у фреймі відображається автоматично, якщо розмір умісту кадра перевищує розміри самого кадра та якщо цей режим не вимкнений явно атрибутом `scrolling=no`.)

Присвоївши ім'я фрейму, можна звертатися до нього, призначеного для відображення документа, на який вказує гіпертекстове посилання. Для цього потрібно додати спеціальний атрибут `target` до тега якоря первинного гіпертекстового посилання (`<a>`). Наприклад, для введення посилання на документ `new.html` з метою його відображення в кадрі 3 (цей фрейм має ім'я `fill_me`) якір потрібно сформувати так:

```
<a href="new.html" target="fill_me">
```

Якщо користувач вибере це посилання, скажімо, в першому кадрі, то документ `new.html` замінить первинний уміст `frame3.html` у третьому фреймі.

Тег `<frameset>` встановлює сукупність кадрів або інші набори кадрів у документі. Допустимим є вкладення наборів кадрів, що забезпечує багатший діапазон можливостей компоновки. Тег `<frameset>` замінює в документі тег `<body>`. Документ не повинен включати нічого іншого, крім допустимого умісту `<head>` і `<frameset>`.

У тезі `<frameset>` використовують два атрибути, які дозволяють задати розмір і кількість стовбців (`cols`) і рядків (`rows`) кадрів або вкладених наборів кадрів, які необхідно відобразити у вікні документа. За допомогою цих атрибутів задається формат подання кадрів, подібно формату сітки чи таблиці. В кожному із вказаних атрибутів задається у лапках список значень, розділених комами. Цими значеннями задається або абсолютна чи відносна ширина (для стовбців), або абсолютна чи відносна висота (для рядків). Кількістю значень атрибута встановлюється, скільки рядків чи стовбців кадрів відображатиме браузер у вікні документа.

Кожне значення атрибутів `rows` і `cols` можна задати одним із трьох способів: як абсолютне число пікселів, у процентному відношенні від загальної ширини чи висоти набору кадрів або ж як частина простору, яка залишається після виділення місця для сусідніх елементів.

Браузер забезпечує максимально можливу відповідність специфікаціям розмірів. Проте слід відмітити, що при цьому, по-перше, не відбувається розширення меж головного вікна документа, і, по-друге, кадри розташовуються так, щоб не залишалось порожнього простору. При виділенні місця для конкретного кадру враховуються розміри всіх решта фреймів у рядку чи стовбці. Кадри розташовуються так, щоб заповнити все вікно документа. Окрім цього, головне вікно документа, яке містить кадри, не має лінійок прокрутки.

Ниже наведений приклад задання висоти рядків у пікселях:

```
<frameset rows="150,300,150">
```

Цим тегом задається створення трьох кадрів, кожен із яких займає у ширину все вікно документа. Висота верхнього та нижнього рядка встановлена у 150 пікселів. Висота середнього рядка – 300 пікселів. Якщо висота вікна браузера не дорівнює 600 пікселів, то браузер автоматично пропорційно розтягує або стискає верхній і нижній рядки так, щоб кожен з них зайняв четверту частину вікна. Для середнього рядка в цьому випадку відводиться половина вікна, яка залишається. У процентному відношенні розміщення аналізованого набору кадрів можна задати так:

```
<frameset rows=25%,50%,25%>
```

Природньо, що в сумі значення у процентах повинні складати 100. У протилежному випадку браузер пропорційно змінює розміри рядків.

Спростити задання розмірів рядків і стовбців можна за допомогою символу *, яким послуговуються для позначення однієї із рівних частин простору вікна, який залишається. Наприклад, тегом

```
<frameset cols="50,*">
```

створюється один фіксований 50-піксельний стовбець з лівого боку вікна, а решта місця відводиться під правий стовбець. Символ * можна використовувати й для позначення декількох стовбців або рядків. Наприклад, якщо заданий тег

```
<frameset rows="*,100,*">
```

то в результаті будуть створені: один рядок висотою 100 пікселів посередині набору кадрів і по одному рядку одинакової висоти над і під ним.

Якщо перед символом * поставити ціле значення, то для відповідного рядка чи стовбця буде виділено у пропорційному відношенні більше наявного простору. Наприклад, тегом

```
<frameset cols="10%,3*,*,*">
```

задається створення чотирьох стовбців. Перший займає 10% загальної ширини набору, другий – три п’ятих решта простору, а третій і четвертий стовбці – по одній п’ятій. Послуговування символом * забезпечує розподіл простору набору кадрів, який залишається.

Якщо явно не вказане протилежне, браузер забезпечує для користувача можливість вручну змінювати розміри окремих стовбців і рядків у документі, який містить кадри. Щоб цього не можна було зробити, потрібно додати в тег `<frame>` атрибут `noresize`.

Шляхом вкладення тегів `<frameset>` можна реалізувати складніші схеми роботи з фреймами. Будь-який кадр у наборі може вміщати інший набір кадрів.

Наприклад, на рис. 5.2 показане вікно, котре містить два стовбця кадрів. Перший стовбець складається з двох рядків, другий – із трьох. Таке подання створено за допомогою вкладення двох тегів `<frameset>` зі специфікаціями рядків в тег `<frameset>` верхнього рівня, яким задаються стовбці:

```
<frameset cols="50%,*">
 <frameset  rows="50%,*">
  <frame src="frame1.html">
  <frame src="frame2.html">
 </frameset>
 <frameset rows="33%,33%,*">
  <frame src="frame3.html">
  <frame src="frame4.html">
  <frame src="frame5.html">
 </frameset>
</frameset>
```

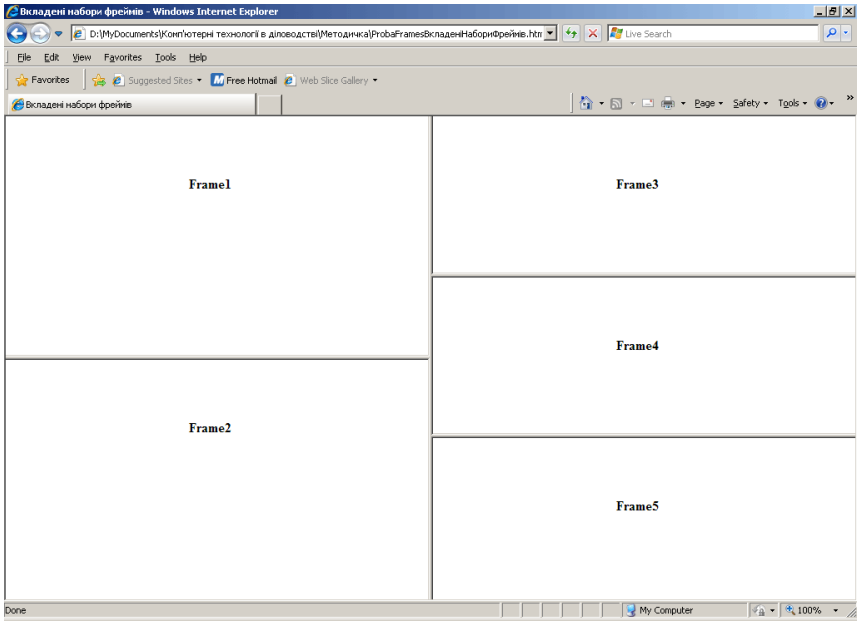

Рис. 5.2. Фреймова структура деякого HTML-документа із вкладеними наборами фреймів

Код HTML-документа, який містить кадри не включає в себе умісту, який відображається. Теги `<frame>` призначені для створення URL-посилань на окремі документи, які відображатимуться в кадрах. Теги `<frame>` – це автономні елементи, для яких не вимагається завершальних компонентів.

При розміщенні набору кадров у вікні вони впорядковуються так: зліва направо по стовбцям, зверху вниз по рядкам, тому порядок слідування і кількість тегів `<frame>` у тезі `<frameset>` відіграють дуже важливу роль.

Нижчее перераховані базові атрибути, якими можна послуговуватись в тезі `<frame>`.

`src=ім'я_документа`

Значення атрибута `src` – URL документа, який відобразиться в аналізованому кадрі. Таким документом може бути будь-який HTML-документ або объект, що відображається, включаючи зображення і мультимедійні дані. Документ, на який вказує посилання, також може містити кадри.

`name=ім'я_кадру`

За допомогою необов'язкового атрибута `name` кадр помічається для наступного звернення шляхом задання атрибута `target` у тезі якоря гиперпосилання `<a>`. Якщо вибирається посилання на `ім'я_кадру`, документ відображається у поіменованому кадрі. Значення атрибута `name` – текстовий рядок у лапках.

`noresize`

Навіть якщо явно задати розміри через атрибути в тезі `<frameset>`, користувачі можуть вручну змінити розміри рядка чи стовбця у фреймовій структурі. Щоб вилучити таку можливість, потрібно в тегах наборів кадрів, відносні розміри рядків або стовбців яких необхідно залишити без змін, задати атрибут `noresize`.

`scrolling=[yes,no,auto]`

Кадри, уміст яких не поміщається в межах відведеного для них простору, при відображенні браузером зазвичай обладнуються лінійками прокрутки. Якщо місця для умісту достатньо, лінійки прокрутки відсутні. Атрибут `scrolling` надає користувачеві можливість явно задавати лінійку прокрутки. Значенням `yes` вмикається режим відображення лінійки, а `no` – вимикається. Значення `auto` відповідає режиму за замовчуванням; це рівносильне тому, що даний атрибут взагалі не використовується.

`marginheight=висота marginwidth=ширина`

Як правило, браузер відображаючи кадр залишає між його межами та його умістом поля невеликого розміру. Ці поля можна встановити вручну за допомогою атрибутів `marginheight` і `marginwidth`, значення яких задаються у пікселях. При цьому потрібно враховувати такі обмеження: розмір поля, по-перше, не повинен бути меншим одного пікселя і, по-друге, не можна задавати його настільки великим, що для умісту кадру не вистачатиме місця.

Для тегів `<frameset>` та `<frame>` існують атрибути, які дозволяють налаштувати відображення обрамлення кадрів.

Атрибутом `frameborder` послуговуються для перемикання із режиму відображення об'ємного обрамлення у режим відображення звичайних ліній. За замовчуванням використовується об'ємне обрамлення (значення атрибута yes або 1)[1]; якщо задане значення `no` або 0, то будуть відображатися звичайні рамки. Цей атрибут можна поміщати як у тег `<frameset>`, так і в тег `<frame>`. Установка в окремому тезі `<frame>` відміняє значення в зовнішньому тезі `<frameset>`.

Атрибут `bordercolor` дозволяє встановити колір обрамлення у тегах `<frameset>` і `<frame>`.

У тезі `<frameset>` можна задавати товщину ліній обрамлення для усього набору кадрів. Це роблять за допомогою атрибута `border`. Стандартна товщина – 5 пікселів. Щоб отримати кадри без обрамлення, задають `border=0` і `frameborder=no`.

ЗАВДАННЯ

1. Організувати фреймову структуру, визначену згідно з варіантом[2].
2. Переробити проект домашньої сторінки/сайта так, щоб була наявною фреймова структура[3].
3. Закрити вікна. Закінчити роботу.

КОНТРОЛЬНІ ЗАПИТАННЯ

1. Яка суть фрейма (кадру)? У якому браузері вперше була реалізована концепція фреймової структури?
2. Яким є код-каркас мовою HTML для задання простої фреймової структури?
3. Яка природа тегів `<frameset>` і `<frame>`?
4. За яких умов у кадрі відображатиметься лінійка прокрутки?
5. Описати механізм, який дозволяє, перейшовши по гіперпосиланню в окремому документі, отримати результат – новий документ у завчасно визначеному фреймі.
6. Як задати рядки та стовбці фреймової структури?
7. Вкладений набір кадрів. Навести приклад коду-каркасу мовою HTML.
8. Чи можна вручну змінювати розміри окремих стовбців і рядків у документі – побудованій фреймовій структурі? Як заборонити можливість зміни користувачем зазначених розмірів?
9. Назвіть базові атрибути, якими послуговуються у тезі `<frame>`.
10. Назвіть атрибути тегів `<frameset>` та `<frame>`, які дозволяють налаштувати відображення обрамлення кадрів.

---

[1] З початку появи цього атрибута значення yes і no стосувалися браузера Netscape Navigator, а відповідні значення 1 та 0 браузера MS Internet Explorer. При роботі із популярними сьогодні браузерами: MS Internet Explorer, Mozilla Firefox, Safari, Opera, Chrome доцільно поекспериментувати та встановити, якими із значень атрибута послуговуватись.

[2] Варіанти подані у додатку Г (див. кольорову вклейку).

[3] Одним із варіантів можливого послуговування фреймами у розроблюваному проекті є реалізація навігаційної системи у вигляді множини гіперпосилань, котрі знаходитимуться в одному із фреймів. Цокання на таких гіперпосиланнях призводить до відображення інших файлів у сусідньому фреймі чи до відображення структурних частин певної сторінки.

ПРАКТИЧНА РОБОТА №6
МУЛЬТИМЕДІА-ЕЛЕМЕНТИ НА ВЕБІ: АУДІО- ТА ВІДЕОКЛІПИ

ТЕОРЕТИЧНІ ВІДОМОСТІ

Очевидно, що мультимедійні елементи привносять в HTML нове життя, додають в інформаційний простір ще один вимір, дозволяючи подавати відомості, які неможливо перенести, наприклад, на папір. Аналізовані елементи не можна сприймати як такі, що ними тільки прикрашають документ. У мережі немає жодних забобонів відносно типів умісту, яким можуть обмінюватися сервери та браузери. В цій лабораторній роботі розглянемо різні способи привнесення на веб-сторінки аудіо- та відеоінформації.

Відомо, що зображення можна включати в HTML-документи та відображати „у рядок” з іншим умістом засобами практично будь-якого браузера. Проте, інколи можна посилатися на зображення зовнішнім чином. Найчастіше мова йде про великі об’єкти, у яких деталі мають істотне значення, але не є на цю хвилину необхідною частиною умісту документа. На інші мультимедійні елементи, зокрема, аудіо та відео, також можна посилатися як на окремі документи, зовнішні відносно аналізованого.

Зазвичай, для посилань на зовнішні мультимедійні елементи послуговуються тегом `<a>`. Браузер завантажує та подає мультимедійні об'єкти (можливо, за допомогою зовнішніх прикладних програм чи плагінів) так само, як і інші елементи, посилання на які вибрав користувач. Процес перегляду зовнішніх елементів складається з двох етапів: спочатку завантажується документ, який посилається на мультимедійний об'єкт, а далі й сам об'єкт, якщо було активоване відповідне посилання.

Можна посилатись на будь-який зовнішній документ, незалежно від його типу чи формату, за допомогою традиційного гіперпосилання (тега `<a>`):

```
<a href=”sounds/Song.au”>Пісня „Червона Рута”</a> є символом
української пісенності.
```

Сервер надсилає браузеру запитаний мультимедійний об'єкт так само, як і будь-який інший документ, як тільки користувач вибирає відповідне гіперпосилання. Якщо браузер виявляє, що отриманий документ не написаний мовою HTML, а має якийсь інший формат, він автоматично викликає підходящий засіб відображення, щоб показати на екрані чи якимось іншим чином донести до користувача уміст отриманого документа.

Можна сконфігурувати браузер спеціальними допоміжними прикладними програмами, які будуть по-різному поводитися з документами різних форматів. Аудіофайли, наприклад, будуть передані засобу обробки аудіофайлів, а відеофайли – відправлені на відеоплеєр. Якщо браузер на відконфігурований для обробки деякого формату документів, то він повідомить про це і запропонує просто зберегти документ на диску. Пізніше користувач зможе вивчити документ за допомогою необхідного засобу перегляду.

Відомо, що зображення можна, зокрема, вмонтувати у веб-документ (так звані внутрішньорядкові зображення) або зробити на них посилання, а відтак, прикріпити до документа. Те ж саме має місце у випадку аудіо- та відеокліпів. Як зробити посилання на аудіо та відео було розглянуте раніше, а зараз зупинемось на тому, як вмонтувати аналізований уміст у саму сторінку посеред тексту та зображень.

Задля цього послуговуються, наприклад, тегом `<embed>`. При цьому посилаються на об'єкт даних за допомогою атрибута `src`, значенням якого є URL файлу, який повинен бути завантажений браузером. Наприклад, наступний фрагмент коду: