МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК І ТЕХНОЛОГІЙ
КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ МАТЕМАТИКИ І ПРОГРАМУВАННЯ

КУРС ЛЕКЦІЙ
З КОМП'ЮТЕРНОЇ ГРАФІКИ

для студентів всіх форм навчання

Галузь знань	***0305 «Економіка та підприємництво»***
Напрям(и) підготовки:	***6.030504 Економіка підприємства паливно-енергетичного комплексу – ЕПЕК***
	6.030507 Маркетинг – МПР

РОЗГЛЯНУТО
на засіданні кафедри ОМіП
протокол № 1 від 30.08.2011 р.

ЗАТВЕРДЖЕНО
на засіданні навчально-видавничої ради ДонНТУ
протокол № 6 від 06.10. 2011 р.

Донецьк, 2011

УДК 681.3.06 (071)

Курс лекцій з комп'ютерної графіки для студентів всіх форм навчання галузі знань 0305 «Економіка та підприємництво» напрямів підготовки: 6.030504 Економіка підприємства паливно-енергетичного комплексу – ЕПЕК та 6.030507 Маркетинг – МПР / О.М.Копитова. – Донецьк: ДонНТУ, 2011. – 81 с.

Курс лекцій містить теоретичний матеріал згідно вимогам робочого навчального плану підготовки бакалаврів з галузі знань 0305 – „Економіка та підприємництво".

Автор О.М. Копитова, доц.

Відп. за видання В.М. Павлиш, д.т.н., проф.

ЗМІСТ

ВСТУП

Комп'ютерна графіка в даний час вже цілком сформувалася як наука. Існує апаратне і програмне забезпечення для отримання різноманітних зображень – від простих креслень до реалістичних образів природних об'єктів. Машинна графіка використовується майже у всіх наукових і інженерних дисциплінах для наочності сприйняття і передачі інформації. Знання основ графіки у наш час необхідне фахівцеві в будь-якій галузі знань. Вживання під час ділових нарад демонстраційних слайдів, підготовлених методами комп'ютерної графіки і іншими засобами автоматизації праці менеджера, вважається за норму. У медицині стає звичайним отримання тривимірних зображень внутрішніх органів за даними комп'ютерних томографів. В наші дні телебачення і інші рекламні підприємства часто удаються до послуг комп'ютерної графіки і мультиплікації. Серед різноманіття можливостей, що надаються сучасними обчислювальними засобами, ті, що засновані на просторово-образному мисленні людини, займають особливе місце. Сучасні програмно-оперативні засоби комп'ютерної графіки є вельми ефективним інструментом підтримки такого мислення при виконанні робіт самих різних видів. З іншого боку саме просторово-образне мислення є неформальною творчою основою для розширення образотворчих можливостей комп'ютерів. Цю важливу обставину передбачає взаємно збагачуюча співпраця все більш досконалої техніки і людини зі всім багатством знання, накопиченого попередніми поколіннями. Око і раніше було ефективним засобом пізнання людиною миру і себе. Тому настільки привабливою виявляється комп'ютерна візуалізація, особливо візуалізація динамічна, яку слід розглядувати як найважливіший інструмент для навчання наукам.

Комп'ютерна графіка – це прикладна область інформатики, призначена для створення, зберігання і обробки зображень за допомогою програмно-апаратних засобів обчислювальної техніки. Термін цей виник в 1965 році. В порівнянні із звичайним текстом комп'ютерна графіка дозволяє піднести інформацію наочніше, конкретно і ясно, допомагає слухачам побачити взаємозв'язки між різними даними.

Кінцевим продуктом комп'ютерної графіки є зображення. Це зображення може бути технічним кресленням, ілюстрацією із зображенням деталі в керівництві по експлуатації, простій діаграмою, архітектурним видом передбачуваної конструкції або проектним завданням, рекламною ілюстрацією або кадром з мультфільму. Таким чином, комп'ютерна графіка охоплює всі види і форми представлення зображень, доступних для сприйняття людиною – або на екрані монітора, або у вигляді копії на зовнішньому носієві (папір, плівка, тканина і інше).

В наступних темах використовано матеріал із [7,8,9].

Тема 1. ТЕОРЕТИЧНІ ОСНОВИ КОМП'ЮТЕРНОЇ ГРАФІКИ. ВЕКТОРНА І РАСТРОВА ГРАФІКА

Види комп'ютерної графіки

Існує декілька підходів до класифікації комп'ютерної графіки. Розглянемо деякі з них.

Комп'ютерна графіка підрозділяється на:

- статичну (нерухома)
- динамічну (анімація, комп'ютерна мультиплікація).

Залежно від колірного обхвату комп'ютерну графіку ділять на **чорно-білу і кольорову.**

Залежно від спеціалізації по окремих областях комп'ютерна графіка буває: **ділова, наукова, інженерна, ілюстративна.**

Ділова графіка

Це оформлення документації, побудова графіків, креслень, схем, підготовка наочного та ілюстративного матеріалу у формі документа на папері, на екрані, на плівці і так далі

Результати розрахункових робіт і статистичні дані зручно представляти графічно у вигляді схем, діаграм, гістограм і графіків.

Ділова графіка застосовується для наочного представлення числової інформації з метою швидшого і якіснішого її сприйняття. Вона застосовується в презентаціях, звітах, рекламі. Це найпростіший і поширеніший вид комп'ютерної графіки.

Програмні засоби ділової графіки – Chart-Master, Vision, графічні блоки офісного програмного забезпечення: ***MS Word, MS Excel, MS PowerPoint*** і найпростіший графічний редактор ***Paint,*** вбудований в Windows;

.

Наукова графіка

Комп'ютерна графіка використовується і в наукових дослідженнях. Зокрема, вона виступає як засіб формування наукової документації з використанням спеціальної нотації (форми запису) – математичних знаків, індексів, шрифтів і тому подібне Останнім часом учені частіше почали звертатися до імітаційного моделювання на комп'ютері, що дозволяє відтворити у видимій формі те, що інколи в принципі не можна побачити очима: розподіл поля температур на поверхні іншої планети, напруги усередині злитка металу, будови складної органічної молекули і так далі

До наукової графіки, зокрема, ***відносяться:***

- засоби комп'ютерного моделювання: імітаційного, структурного і ін. (наприклад, тканин, одягу, і так далі) - це історично перше широке застосування комп'ютерної графіки;
- комп'ютерна томографія в медицині;
- засобу візуалізації будови речовини, векторних полів і інших даних в наукових дослідженнях;

- засоби для побудови і відображення формул на екрані;
- засоби побудови складних розподілів і графіків.

Через різноманітність видів наукового пізнання, ***клас програмних засобів, що використовують наукову графіку***, дуже широкий, але вони менш відомі. Відзначимо такі засоби як елемент MS Office – ***Microsoft Equation*** і сімейство ***Case-средств,*** таких як ***BPwin, Rational Rose, ARIS.***

Інженерна (конструкторська) графіка

Комп'ютеризацію креслярських і конструкторських робіт проводять давно і в даний час використовують різні системи автоматизації проектних робіт (САПР). Існує два класи таких систем: універсальні креслярські Сапри (Cad-системы) і спеціалізовані під певну наочну область (Cam-системы).

Метою инженерой графіки є найбільш точне представлення креслення об'єкту в пам'яті комп'ютера. При цьому більша увага приділяється відповідності цього об'єкту вимогам Гостів, чим красі і ефектності відображення об'єкту.

Найважливіші **сфери вживання інженерної графіки**:

- системи автоматизації виробництва за допомогою робототехнічних систем;
- системи автоматизації проектування, конструювання (літаків, автомобілів, і так далі);

Набор програмних засобів, що підтримують інженерну графіку: ***AUTOCAD, Компас, Solid Edge, Unigraphics, Inventor*** і ін.

Ілюстративна графіка

Ілюстративна графіка ***застосовується*** для створення красивих, ефектних віртуальних об'єктів. На відміну від всіх інших видів КГ, мета яких представити ***інформації про об'єкт*** у вигляді діаграм або графіків, формул або моделей, креслень або схем, ілюстративна графіка використовується сама по собі. З її допомогою створюють або редагують зображення на екрані комп'ютера, тому її часто ототожнюють з поняттям комп'ютерної графіки.

Сфера вживання ілюстративної графіки дуже широка. До неї належать:

- бізнес (наприклад, показ різної дисплейної техніки і її використання);
- мистецтво (використання засобів комп'ютерної графіки для підготовки високореалістичних сцен);
- засоби масової інформації (різні заставки і телеэффекты на екрані, створення рекламної продукції, розробка логотипів;
- учбові системи. Доведено, що матеріал, що супроводжується зображенням і звуком, легше сприймається і запам'ятовується на триваліший термін;
- комп'ютерні ігри і мультиплікація. Мультиплікація використовується не лише в іграх, але і в ділових розробках (наприклад, в архітектурі і дизайні), в рекламі, при демонстрації ділових пропозицій.
- поліграфія і друкарська справа;
- комп'ютерний дизайн програм;
- web-дизайн комп'ютерних сайтів;
- обробка фотографій і текстів;
- створення анімації;
- створення комп'ютерних ігор;
- комп'ютерна творчість.

Програми ілюстративної графіки: ***Adobe Photoshop, Adobe illustrator, Corel Draw, тривимірні Maya, 3DStudio Max*** і безліч інших

Залежно від способу формування зображень комп'ютерну графіку прийнято підрозділяти на:

растрову	**векторну**	**фрактальну**	**тривимірну**
базовым елементом зображення є ***крапка***	базовым елементом зображення є ***лінія***	заснована на математичних обчисленнях. Базовий елемент фрактальної графіки - сама математична формула, тобто ніяких базових об'єктів в пам'яті комп'ютера не зберігаються, і зображення будується виключно по рівняннях	(3D) поєднує векторний і растровий способи формування зображень

Растрова графіка застосовується ***для обробки готових зображень*** при підготовці електронних (мультимедійних) і поліграфічних видань. Растрові ілюстрації рідко створюють вручну за допомогою комп'ютерних програм. Частіше для цього використовують фотографії, підготовлені художником на папері, або ілюстрації, що сканують,. Для введення растрових зображень в комп'ютер застосовують також цифрові фото- і відеокамери. Тому більшість графічних растрових редакторів орієнтована більшою мірою ***на обробку*** зображень, а не на їх створення. Растрова графіка використовується і в Інтернеті.

Векторна графіка, навпаки, використовується ***для створення*** зображень, придатних для поліграфії. Програмне забезпечення для роботи з векторною графікою використовується в різних рекламних агентствах, дизайнерських бюро, редакціях і видавництвах. З його допомогою набагато простіше створювати зображення, засновані на вживанні шрифтів і простих геометричних елементів. Високохудожні добутки можуть створюватися за допомогою векторної графіки, але їх створення дуже складно.

Фрактальна графіка заснована на автоматичній генерації зображень шляхом математичних розрахунків. Створення фрактальних зображень полягає не в малюванні, а в програмуванні. Фрактальну графіку рідко застосовують для створення друкарських або електронних документів, але її часто використовують для створення незвичайних текстур, фонів, елементів зображень, для побудови як простих регулярних структур, так і складних ілюстрацій, що імітують, наприклад, природні ландшафти і тривимірні об'єкти.

За окремий предмет вважається ***тривимірна графіка***, що вивчає прийоми і методи побудови об'ємних моделей об'єктів у віртуальному просторі.

Ми зупинимося докладніше на растровій і векторній графіці.

Растрова графіка

Призначення растрової графіки:

1) для редагування різних типів цифрових зображень, у тому числі фотографій;
2) для виконання колірної корекції зображення;
3) для створення ілюстрацій, що використовуються в поліграфічному друці і мультимедійних електронних документах;
4) для розробки ілюстративного матеріалу при створенні web-сторінок

Особливості растрової графіки

Растрове зображення є прямокутною сіткою **(растр)**, комірки якої називаються **крапками**, або **пікселями** (***pixel*** - скорочення від слів picture element, тобто елемент зображення).

Піксель – мінімальна одиниця зображення, одна з найдрібніших кольорових крапок, з яких складається зображення. Кожна крапка має параметри:

- координати по горизонталі і вертикалі;
- колір;
- яскравість.

Таким чином, будь-яке растрове зображення − це набір забарвлених пікселів. Обробка растрових зображень зводиться до редагування окремих крапок.

На мал. 1.1 показано збільшене і розбите на крапки (пікселі) зображення вкладених квадратів білого і чорного кольору.

Лівий верхній кут малюнка має координати (0,0), а правий нижний - координати (12,12). Використовується наступне кодування кольорів - піксельне значення 0 − білий колір, піксельне значення 255 − чорний колір.

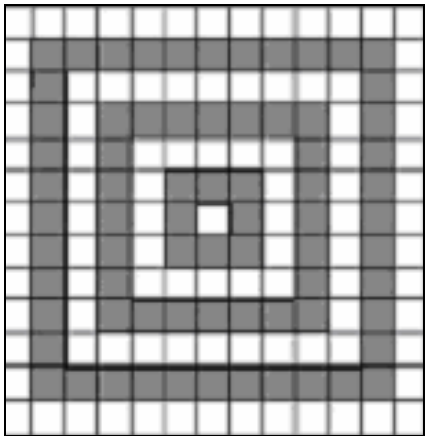
Терміном піксель окрім окремого елементу растрового зображення позначають також окрему крапку на екрані комп'ютера і окрему крапку на роздруку принтера. Для уточнення часто використовують **терміни:**

- ***піксель*** − окремий елемент растрового зображення;
- ***відеопіксель*** − найменший елемент зображення на екрані;
- ***крапка*** − найменший елемент, що створюється принтером.

При цьому для зображення одного пікселя на екрані комп'ютера може бути використаний один або декілька відеопікселів.

Кодування кольору в комп'ютерній графіці

Вважається, що піксель зображення не має ніякого розміру. Він лише область пам'яті комп'ютера, в якій зберігається інформація про колір, хоча на екрані звичайного пристрою відображення (електронно-променевої трубки) можна розгледіти окремі крапки, діаметр яких менше третьої частки міліметра.



Рiс.1.1 Растрове зображення

На рис.1.1 растрові дані виглядають таким чином:

```
0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
0, 255, 255, 255, 255, 255, 255,255, 255, 255, 255, 255,   0,
0, 255,   0,   0,   0,   0,   0,   0,   0,   0,   0, 255,   0,
0, 255,   0, 255, 255,255, 255, 255, 255,255,   0, 255,   0,
0, 255,   0, 255,   0,   0,   0,   0,   0, 255,   0, 255,   0,
0, 255,   0, 255,   0, 255, 255, 255,   0, 255,   0, 255,   0,
0, 255,   0, 255,   0, 255,   0, 255,   0, 255,   0, 255,   0,
0, 255,   0, 255,   0, 255, 255, 255,   0, 255,   0, 255,   0,
0, 255,   0, 255,   0,   0,   0,   0,   0, 255,   0, 255,   0,
0, 255,   0, 255, 255, 255, 255, 255, 255,255,   0, 255,   0,
0, 255,   0,   0,   0,   0,   0,   0,   0,   0,   0, 255,   0,
0, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,   0,
0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
```

[1] У векторній графіці цей же малюнок був би записаний у виді
Rect,White; 1,1,11,11, Rect,Black; 2,2,10,10, Rect,White; 3,3,9,9, Rect,Black; 4,4,8,8, Rect,White; 5,5,7,7, Rect,Black.
Чи в короткому вигляді:
0,0,13,13,R,W;1,1,11,11,R,B;2,2,10,10,R,W;3,3,9,9,R,B;4,4,8,8,R,W;5,5,7,7,R,B.

Піксель може бути забарвлений в будь-який колір. Колір запам'ятовується в комп'ютері за допомогою комбінації бітів. Чим більше бітів для цього використовуються, тим більше відтінків кольорів має зображення. Найбільш простий тип зображення складається з пікселів, які мають два можливі кольори, – чорний і білий. Зображення, які складаються з пікселів цього вигляду, називаються **однобітними**.

Кількість бітів, які використовуються для кодування кольору пікселя, називається **бітовою глибиною (**або **піксельною глибиною)**. Якщо для зберігання інформації про колір пікселя потрібно, наприклад, 2, 3, 8 або 24 біта, то зображення називається відповідно 2-,3-,8- або 24-х бітовим зображенням і при цьому говорять, що використовується **двух-, трьох-, восьми- або 24-х розрядна палітра**.

Хай k – бітова глибина пікселя. Тоді **число доступних кольорів дорівнює** 2^k. Якщо бітова глибина пікселя дорівнює 1, то використовуються два кольори,

якщо кількість бітів для кодування кольору пікселя дорівнює 2, то маємо 2^2, тобто 4 можливих кольори, і так далі, як показано в таблиці 1.

Таблиця 1. Бітова глибина і кількість кольорів

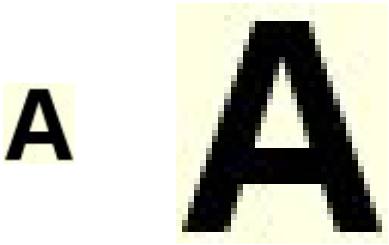
Бітова глибина (розрядність палітри)	Кількість можливих кольорів	Кодування цветов
1	2	0,1
2	4	00,01,10,11
4	16	0000, 0001 0010, .0011;..,1111
8	256	00000000, 00000001,..,11111111
24	16777216	Аналогічно до попередньому

Достоїнства растрової графіки

1. ***Точність передачі зображень, що сканують***. Хороше растрове зображення ***виглядає реально і природно,*** тобто растрова графіка ефективно представляє предмети фізичного миру. Це відбувається тому, що людське око сприймає мир як величезний набір дискретних елементів, який створюють ці предмети.
2. Растрове зображення найбільш ***адаптоване для*** поширених ***растрових пристроїв виведення:*** лазерних принтерів та ін.

Недоліки растрової графіки

1. У зв'язку з необхідністю кодування кожного пікселя зображення, растрові зображення пов'язані з ***великими об'ємами інформації***. Для обробки великих растрових зображень потрібні ***істотні комп'ютерні ресурси*** (по пам'яті і за часом). Наприклад, для роботи з ілюстраціями типа журнальної смуги потрібно більше 128 Мбайт оперативної пам'яті.
2. Оскільки зображення складається з крапок, те збільшення зображення приводить до того, що збільшується розмір крапок, стають помітні елементи растру, а це викликає спотворення ілюстрації. При повороті растрового зображення на невеликий кут також відбувається спотворення ліній. Це означає, що ***при будь-яких трансформаціях*** (поворотах, масштабуванні, нахилах) ***растрове зображення спотворюється.*** Цей ефект називається ***пикселизацией***.



Найбільш популярні растрові графічні редактори:

Adobe Photoshop, Corel PhotoPaint (доповнення до програми CorelDRAW)***, Paint*** (вхідний до складу всіх операційних систем Windows), ***Macromedia Fireworks, Fractal Design Painter***

Роздільність растрових зображень

Для растрових зображень, що складаються з крапок, особливу важливість має поняття **роздільності**, що виражає **кількість крапок,** що доводяться **на одиницю довжини**. При цьому слід розрізняти:

- роздільність оригіналу;
- роздільність екранного зображення;

• роздільність друкарського зображення.

Роздільність оригіналу. Дозвіл оригіналу вимірюється в **крапках на дюйм** ***(dpi —*** *dots per inch)* і залежить від вимог до якості зображення і розміру файлу, способу оцифрування або методу створення початкової ілюстрації, вибраного формату файлу і інших параметрів. У спільному випадку діє правило: чим вище за вимогу до якості, тим вище має бути дозвіл оригіналу.

Роздільність екранного зображення. Для зображення на екрані розмір пікселя змінюється залежно від вибраного екранного дозволу (з діапазону стандартних значень), р**оздільності** оригінала і масштабу відображення.

Монітори для обробки зображень з діагоналлю 20-21 дюйм (професійного класу), як правило, забезпечують ***стандартні екранні дозволи*** 640x480, 800x600, 1024x768, 1280x1024, 1600x1200, 1600x1280, 1920x1200, 1920x1600 крапок. Відстань між сусідніми крапками люмінофора біля якісного монітора складає 0,22-0,25 мм. Відмітимо, що роздільна здатність не визначається монітором взагалі. Вона визначається відеокартою і програмним забезпеченням, що працює з цим пристроєм.

Роздільність друкарського зображення і принтера. Розмір точки растрового зображення на папері (плівці і т. д.) залежить від линиатуры растру. **Лініатура растру** – це число ліній растру на дюйм зображення (***lpi*** *lines per inch*). Чим вище лініатура, тим вище роздільна здатність друку, і, отже, якісній відтворення дрібних деталей оригіналу.

Роздільність принтерів указується в ***dpi*** (dots per inch, кількість крапок на дюйм). Значення, які указують виробники принтерів, можуть досягати 9600 і більш dpi. Нагадаємо, що дозвіл цифрового зображення при обробці в програмі вимірюється в пікселях на дюйм (ppi). При друці ж один піксель відображується декількома крапками (dots), тому роздільність принтера в dpi зазвичай висока. В той же час, драйвер принтера приймає на обробку зображення в певній роздільності в ppi (т.з. фізичний дозвіл). Для принтерів Epson ця роздільність зазвичай дорівнює 720 ppi, для принтерів Canon 600 ppi.

Емпіричне правило

Для екранної копії досить роздільності 72 dpi, для роздруку на кольоровому або лазерному принтері 150 200 dpi, для виводу на фотоекспонуючому пристрої 200-300 dpi. Встановлено ***емпіричне правило:*** при роздруку величина роздільності оригіналу має бути в 1,5 разу більше, ніж лініатура растру пристрою виводу. У випадку, якщо друкарську копію треба збільшити в порівнянні з оригіналом, ці величини слід помножити на коефіцієнт масштабування.

ПРИКЛАД

Хай дано зображення розміром 1×1 дюйм^2 з роздільністю 72 ppi. Тоді дане зображення містить 1*72 × 1*72 = 5184 пікселів. Якщо зображення 1×1 дюйм матиме роздільність 300 ppi, то воно міститиме 90000 пікселів. За рахунок використання більшого числа пікселів висока роздільність дозволяє отримувати при друці зображень дрібніші деталі і тонші колірні переходи. При вищому дозволі кодується більша кількість пікселів, тому розмір файлу буде більший. Вибір оптимальній роздільності залежить від того, на якому дисплеї воно відображуватиметься або на якому принтері буде надруковано. Якщо роздільність

зображення нижче, ніж роздільність принтера, то якість друку буде низькою. Це пов'язано з великим розміром пікселів в цьому випадку.

изображение	принтер	лист
100 ppi	800 ppi	100 ppi
700 ppi	800 ppi	700 ppi
1500 ppi	800 ppi	800 ppi

Якщо роздільність зображення багато вище, ніж роздільність принтера, то друкарська ілюстрація буде роздрукована з максимальною якістю, яка дозволяє отримати принтер. При цьому об'єм файлу, де зберігається зображення, буде тим більше, чим більше роздільність. Великий об'єм файлу у свою чергу знижує швидкість друку і швидкість передачі файлу по мережі.

Можливості монітора, пов'язані з роздільністю зображення при його відображенні на екрані, набагато нижче, ніж можливості принтера при друці того ж зображення. Для цього досить порівняти дозволи зображень на звичайній кольоровій фотографії 10x15 см і екранного зображення монітора з діагоналлю 15 дюймів і роздільністю 800x600 пікселів. Роздільність першій – близько 260 dpi, другого – близько 72 ppi.

Векторна графіка

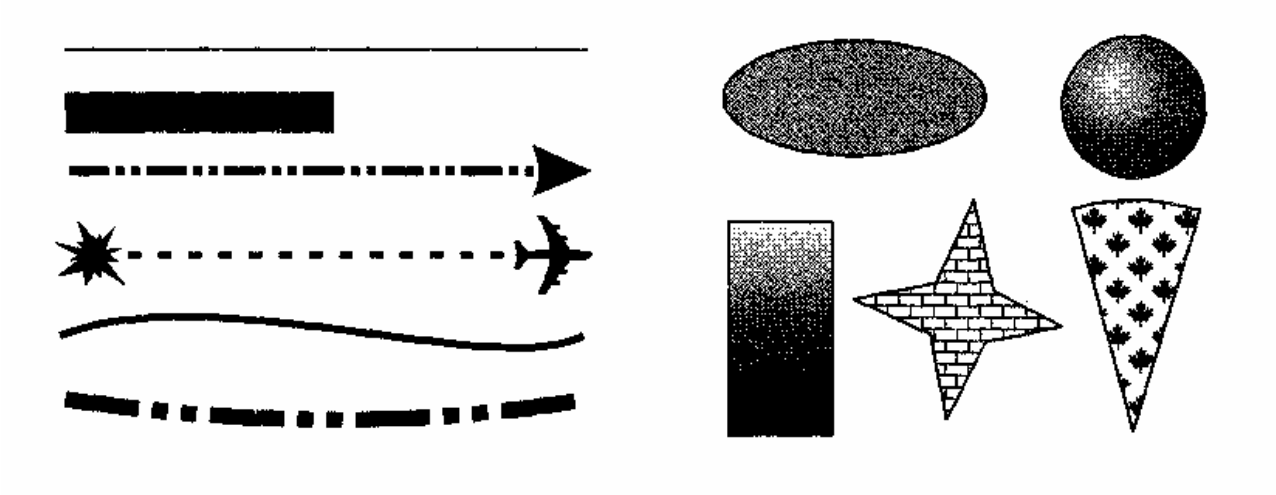
Призначення векторної графіки:

1) для обробки готових малюнків і документів з метою поліпшення їх якості;
2) для створення високоякісної рекламної продукції;
3) для створення таблиць і графіків в документах;
4) для оформлення ілюстрацій до технічних книг;
5) для розробки ілюстративного матеріалу при створенні web-сторінк .

Особливості векторної графіки

Якщо в растровій графіці базовим елементом зображення є крапка, то у векторній графіці ***лінія*** (звана ***вектором***). Відзнака ліній растрової графіки від ліній векторної графіки полягає в способі виділення пам'яті комп'ютера для них. ***Лінія в растровій графіці*** розглядується як ***сукупність крапок***. При цьому колір кожної крапки кодується окремо і для кожної крапки виділяється окремий елемент пам'яті. У векторній же графіці *вся* лінія задається своїми ***параметрами***, а не кожна її крапка окремо. При зміні лінії міняються тільки її параметри, кількість же параметрів залишається незмінною.

Всю векторну ілюстрацію можна розглядувати як сукупність ліній, для кожної з яких вказані свої параметри. Наприклад, куб можна скласти з шести зв'язаних прямокутників, кожен з яких, у свою чергу, утворений чотирма зв'язаними лініями. Можна представити куб і як дванадцять зв'язаних ліній, що створюють ребра. Наведемо приклади об'єктів векторної графіки:



Ріс.1.2. Об'єкти векторної графіки

Математичні основи векторної графіки

Як і всі об'єкти лінії мають властивості. Замкнуті лінії мають ***властивість заповнення***. Внутрішня область *може бути заповнена* ***кольором, текстурою або картою*** (карта заздалегідь підготовлене растрове зображення). Проста незамкнута лінія має дві вершини, звані вузлами, які теж мають свої властивості, від яких залежить, як виглядає вершина.

Розглянемо способи представлення різних об'єктів у векторній графіці

Крапка. Цей об'єкт на плоскості представляється двома числами *(х, біля),* вказуючими його положення відносно початки координат.

Пряма лінія. Їй відповідає рівняння $y = kx + b$. Вказавши параметри k і b, завжди можна відображувати нескінчену пряму лінію у відомій системі координат, тобто для завдання прямій вистачає двох параметрів.

Відрізок прямої. Він відрізняється тим, що вимагає для опису ще двох параметрів — наприклад, координат *xl* і х2 зачала і кінця відрізка.

Крива другого ладу. До цього класу кривих відносяться параболи, гіперболи, еліпси, кола, тобто всі лінії, рівняння яких містять ступені не вище другий. Крива другого ладу не має *точок перегину.* Прямі лінії є всього лише окремим випадком кривих другого ладу. Формула кривої другого ладу в спільному вигляді може виглядати, наприклад, так:

$$x^2 + a_1 y^2 + a_2 xy + a_3 x + a_4 y + a_5 = 0$$

Таким чином, для опису нескінченної кривій другого ладу вистачає п'яти параметрів. Якщо потрібно побудувати відрізок кривої, знадобляться ще два параметри.

Крива третього ладу. Відзнака цих кривих від кривих другого ладу полягає в можливій наявності точки перегину. Наприклад графік функції $y=x^3$ має точку перегину на початку координат. Саме ця особливість дозволяє зробити криві третього ладу основою відображення природних об'єктів у векторній графіці. Наприклад лінії вигину людського тіла вельми близькі до кривих третього ладу. Всі криві другого ладу, як і прямі, є окремими випадками кривих третього ладу.

У спільному випадку рівняння кривої третього ладу можна записати так:

$$x^3 + a_1 y^3 + a_2 x^2 y + a_3 xy^2 + a_4 x^2 + a_5 y^2 + a_6 xy + a_7 x + a_8 y + a_9$$

Таким чином, крива третього ладу описується дев'ятьма параметрами. Опис її відрізання зажадає на два параметри більше.

Приклад 1. Для зображення лінії в комп'ютерній графіці може використовуватися запис: Line, 12, 67, 140, 219, blue. Запис означає: від крапки з координатами 12 (по горизонталі) і 67 (по вертикалі) до крапки з координатами (140, 219) провести лінію (Line) блакитного кольору (blue).

Приклад 2. Зображення не закрашеного кола з контуром червоного кольору, з центром в крапці (118,136) і радіусом 50 пікселів у векторній графіці може бути записане в такий спосіб: Circle, 118, 136, 50, red. З метою скорочення розміру запису зображення використовується коротка форма запису виду L, 12, 67, 140, 219, B; C, 118, 136, 50, R.

Роздільність векторних зображень

Оскільки дисплей і принтер є растровими пристроями, то при передачі на них об'єктів векторної графіки над ними виконуються операції растеризування. При виконанні цієї операції лінії векторного об'єкту розбиваються на невеликі прямолінійні відрізки. Чим менше довжина таких відрізань, тим більше точно передається форма кривих ліній, але при цьому збільшується кількість елементарних відрізань і підвищується складність кривих ліній. При передачі векторного зображення на монітор або принтер кожен елементарний відрізок відтворюється як растрове зображення. При цьому програма проводить обчислення координат екранних крапок, тому векторну графіку називають обчислюваною графікою.

Достоїнства векторної графіки

1. Малий об'єм пам'яті. При кодуванні векторного зображення зберігається не само зображення об'єкту, а тільки декілька його параметрів, по яких програма всякий раз відтворює зображення заново. Тому об'єм пам'яті малий в порівнянні з точковою графікою.
2. Свобода трансформації. Векторне зображення можна обертати, масштабувати без втрати якості зображення.
3. Апаратна незалежність. Векторна графіка "працює" з ідеальними об'єктами, які самі пристосовуються до змін: можна не знати, для яких пристроїв робиться той або інший документ. Векторна графіка максимально використовує можливості роздільної здатності будь-якого вивідного пристрою: зображення завжди буде настільки якісним, на скільки здатний даний пристрій.

Недоліки векторної графіки

1. Вона обмежена в художніх засобах: неможливо зберегти півтонові зображення в близькому до оригіналу вигляді.
2. У програмах векторної графіки неможливо створювати і обробляти фотозображення (вставляти можна).
3. Векторний принцип опису зображень не дозволяє автоматизувати введення графічної інформації з сканера або цифрової камери.

Найбільш поширеними векторними графічні програми:

Visio, Adobe Illustrator, Corel Draw, Macromedia Freehand.

Вживання растрової і векторної графіки

Растрова графіка застосовується:

для зберігання і обробки півтонових зображень (картини, що сканують або спочатку створені на комп'ютері, фотографії);

у веб-сервері-дизайні. Вживані на веб-сторінках зображення, як правило не великі, а вивід їх на екран здійснюється самим веб-сервером-оглядачем без вживання додаткових програм.

Векторна графіка краще всього підійде у випадку:

збереження штрихових зображень (карт, креслень, малюнків олівцем, гравюр) в електронному вигляді;

створення невеликих зображень, які надалі оброблятимуться при виводі.

На практиці засоби векторної графіки застосовуються зазвичай для видавничих, оформлювальних, креслярських і проектно-конструкторських робіт. Векторна графіка незамінна в тих областях графіки, де принципове значення має збереження ясних і чітких контурів, наприклад, в шрифтових композиціях, в створенні логотипів і ін.

У решті випадків можна використовувати як векторну, так і растрову графіку, не забуваючи про їх недоліки і переваги.

Приклад визначення реального розміру растрового зображення

Хай розмір растрового малюнка заданий координатами його кутів на прямокутній сітці: (2, 32): (722, 32); (722, 512); (2,512) і він відображується на екрані дисплея з роздільною здатністю 100 пикселей/дюйм, або 40 пікселів/см. (т.к.1дюйм=2,5 см). Тоді розмір малюнка в пікселях рівний 720x480 пікселів. Реальні розміри зображення:

по горизонталі - 720/40 = 18,0 см

по вертикалі - 480/40 = 12.0 см

Отже, розмір растрового зображення дорівнює 18* 12см^2.

Визначення розміру файлу з растровим зображенням без стиснення

Існує пряма залежність між розміром файлу, в якому зберігається растрове зображення, і розміром самого зображення. Чим більше зображення (у пікселях), тим більше розмір файлу за умови, що порівнюються файли, записані в одному графічному форматі з однаковим методом стиснення.

Розмір файлу для зображення, яке не стисле, визначається в байтах по формулі:

$$F=(G*V*B)/8, \quad \text{де}$$

G - горизонтальний розмір зображення в пікселях

V - вертикальний розмір зображення в пікселях

B - бітова глибина, використовувана для даного зображення.

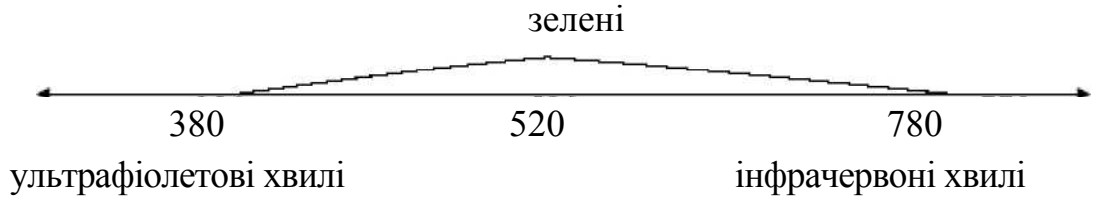
Ділення на 8 використовується для представлення розміру файлу в байтах.

Тоді для зображення без стиснення, яке має той же розмір, що і в попередньому прикладі, тобто 720x480 пікселів, за умови, що використовується палітра з шістнадцятьма кольорами (2^4=16 → B=4), отримуємо:

F = 720*480*4/8 = 172800 Байт = 168,75 Кб.

Тема 2. КОЛІР І КОЛІРНІ МОДЕЛІ. ФОРМАТИ ГРАФІЧНИХ ФАЙЛІВ

Ми бачимо предмети тому, що вони випромінюють або відображають світло. **Світло** – це електромагнітні хвилі, що викликають в оці людини зорові відчуття. Такою здатністю володіють тільки хвилі завдовжки ***від 380 нанометрів (сині кольори) до 780 нанометрів (червоні кольори)*** (див. рис.2.1). Найбільш чутливе око людини до зелених кольорів (520 нм), потім до червоного і синього. У видимому спектрі людське око розрізняє 120 кольорів.

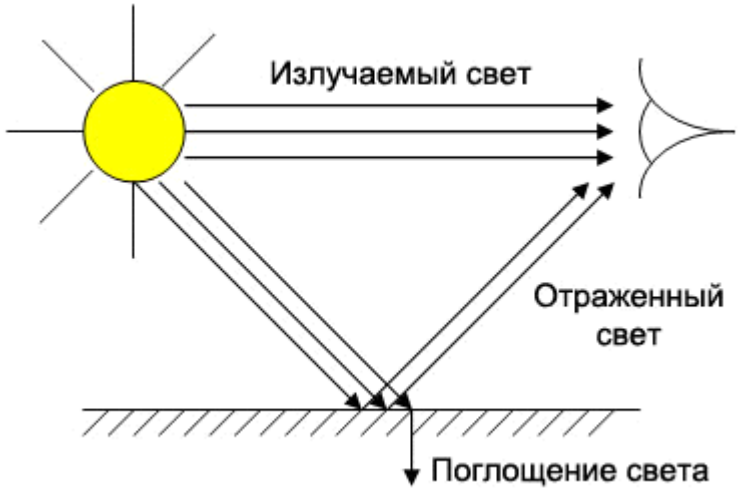


Ріс.2.1. Чутливість ока до довжини хвилі

Кольори, що існують в природі, ***діляться на ахроматичних і хроматичних***. До групи ***ахроматичних*** відносяться всі білі, чорні і проміжні між ними сірі кольори. Група ***хроматичних*** кольорів включає всі кольори спектру, а також кольори, яких немає в спектрі, але є результатом змішення спектральних кольорів - золотисті, тютюнові, теракотові, пурпурні і тому подібне

Промені світла, потрапляючи на сітківку ока, проводять відчуття кольору. **Колір** – це характеристика дії випромінювання на око людини.

Випромінюване світло – це світло, що виходить з джерела, наприклад сонця, лампочки або екрану монітора (мал. 2.2). Випромінюване світло, що йде безпосередньо від джерела до ока, зберігає в собі всі кольори, з яких він створений. При віддзеркаленні від об'єкту світло може змінитися. Будь-який предмет, *що не є* джерелом світла, частково відображає і частково поглинає падаюче на нього світло (див. мал. 2.2).



Мал. 2.2. Випромінювання, віддзеркалення і поглинання світла

Монітор випромінює світло. Папір, на якому друкується зображення, відображає світло. Оскільки колір може вийти і в процесі випромінювання, і в процесі віддзеркалення, то існують два протилежні методи його опису: ***аддитивна і субтрактивная*** колірні моделі. Спосіб розділення колірного відтінку на компоненти, з яких він складається, називається **колірною моделлю.** Вся безліч кольорів, які можуть бути створені в колірній моделі, називається **колірним діапазоном**.

Колірні моделі потрібні для ***математичного опису спектру кольорів***, видимих на екрані монітора, або на скануючих та друкуючих пристроях. Кольори представляються моделлю як результат ***змішення декількох базових (основних) кольорів, з яких вони складаються*** . Кожен базовий колір має свій ***діапазон інтенсивності***. При складанні всіх базових кольорів з різною інтенсивністю утворюються кольори, доступні для даної моделі. Колірні діапазони моделей можуть розрізнятися.

Розглянемо основні колірні моделі, найчастіше використовувані в комп'ютерній графіці. Вони мають абревіатури**: RGB, CMYK, HSB.**

Аддитивна колірна модель RGB

Це апаратно-орієнтована модель, в якій кольори описуються за допомогою складання ***трьох базових кольорів – червоного, зеленого, синього*** – в різних пропорціях**.** Тому модель **RGB називають аддитивною** (від англ. «add» складати, додавати). Кольори також називають ***колірними каналами моделі RGB***. Модель названа по перших буквах англійських слів:

R **RED** червоний
G **GREEN** зелений
B **BLUE** синій.

Кожен з базових кольорів може приймати ***інтенсивність в діапазоні від 0 до 255***. ***Повна кількість кольорів***, що представляються цією моделлю, дорівнює 256*256*256 = 16 777 216.

За допомогою моделі RGB ***описуються кольори, що отримуються змішенням світлових променів***. У цій моделі працюють монітори, телевізори, сканери, слайд-проекторы, кольорові лампи реклами і інші пристрої, в яких колір виходить шляхом змішення світлових пучків. Вона також використовується для опису кольорів на сторінках Інтернета в спеціальному шістнадцятиричному вигляді (#RRGGBB).

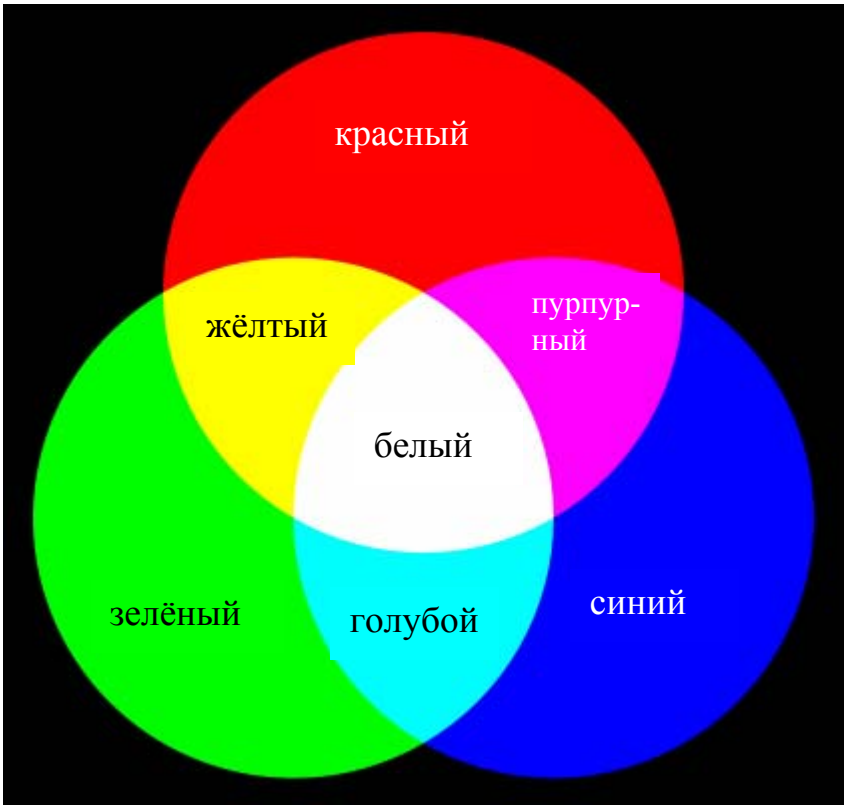
Кожному кольору можна поставити у відповідність ***код***, який ***містить значення яскравості трьох складових***. Використовуються ***десяткове і шістнадцятиричне представлення коду.*** Десяткове подання – це трійка десяткових чисел, розділених комами. Перше число відповідає яскравості червоної складової, друге – зеленої, а третє — синьої. Код кольору в шістнадцятиричному уявленні має вигляд 0xXXXXXX. Префікс 0x указує на те, що ми маємо справу з шістнадцятиричним числом, а не яким-небудь іншим. За префіксом слідують шість шістнадцятиричних цифр (0, 1, 2...,9, A, B, З, D, E, F). Перші дві цифри – шістнадцятиричне число, що представляє яскравість червоної складової, друга і третя пари відповідають яскравості зеленою і синьою складових. Якщо всі складові мають максимальну яскравість (255,255,255 – в десятковому поданні; 0xFFFFFF — в шістнадцятиричному поданні), то виходить білий колір. Мінімальна яскравість (0,0,0 або 0x000000) відповідає чорному кольору. Змішання червоного, зеленого і синього кольорів з однаковими яркостями дає шкалу з 256 відтінків (градацій) сірого кольору - від чорного до білого. Наприклад, (125,125,125) або (220,220,220). Зображення у відтінках сірого ще називають ***півтоновими зображеннями***.

Якщо за допомогою лупи розглянути екран працюючого монітора, то можна відмітити, що він складається з найдрібніших точок (зерен) червоного, зеленого і синього кольорів. Оскільки ці зерна дуже малі, наші очі змішують три кольори в один. Таким чином, сусідні різноколірні крапки зливаються, формуючи інші кольори:

червоний + зелений = жовтий;
червоний + синій = пурпурний;
зелений + синій = блакитний;
червоний + зелений + синій = білий.

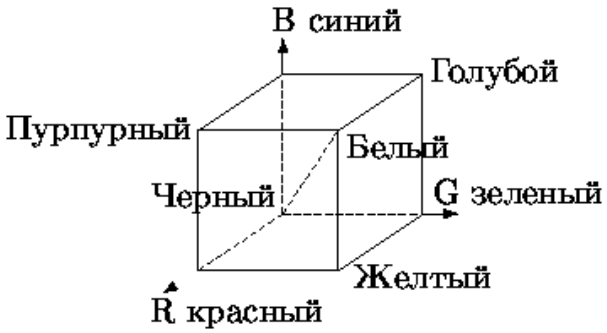
На мал. 2.3 показано, як різні комбінації червоного, зеленого і синього кольорів дають жовтий, пурпурний, блакитний і білий кольори.

Змінюючи ***інтенсивність свічення кольорових крапок***, можна створити ***велике різноманіття відтінків. Якщо інтенсивність кожного з них максимальна (255), то виходить білий колір. Відсутність всіх трьох кольорів дає чорний колір. Якщо змішуються всі кольори з однаковою інтенсивністю (але не максимальною і не мінімальною), отримуємо сірий колір.***



Мал. 2.3. Комбінації базових кольорів моделі RGB

Для зображення аддитивної моделі найчастіше ***застосовують одиничний куб*** з розподілом кольорів уздовж одиничних векторів (див. мал. 2.4). Початок відліку (0,0,0) відповідає чорному кольору. Максимальне значення RGB (1,1,1) відповідає білому кольору. (1;0;0)- червоному, (0;1;0) - зеленому, (0;0;1) синьому.



Мал. 2..4. Колірний куб моделі RGB

Недолік моделі RGB полягає в тому, що не всі кольори, створені в ній, можна вивести на друк. Проте більше 16 млн кольорів, що представляються в RGB, виявляються цілком достатніми для практичних потреб. Іншими словами, кольори на екрані вашого монітора можуть виглядати інакше при їх виводі на друк, причому ця відзнака може виявитися принциповою, а не тільки обумовленою низькою якістю принтера або монітора.

Субтрактівная колірна модель CMY (CMYK)

Це апаратно-орієнтована модель, ***використовувана в поліграфії*** для підготовки зображень на поверхнях, які ***відображають світло*** (друкарські і принтери фарби, плівки і так далі). Такі зображення видно у відбитому світлі.

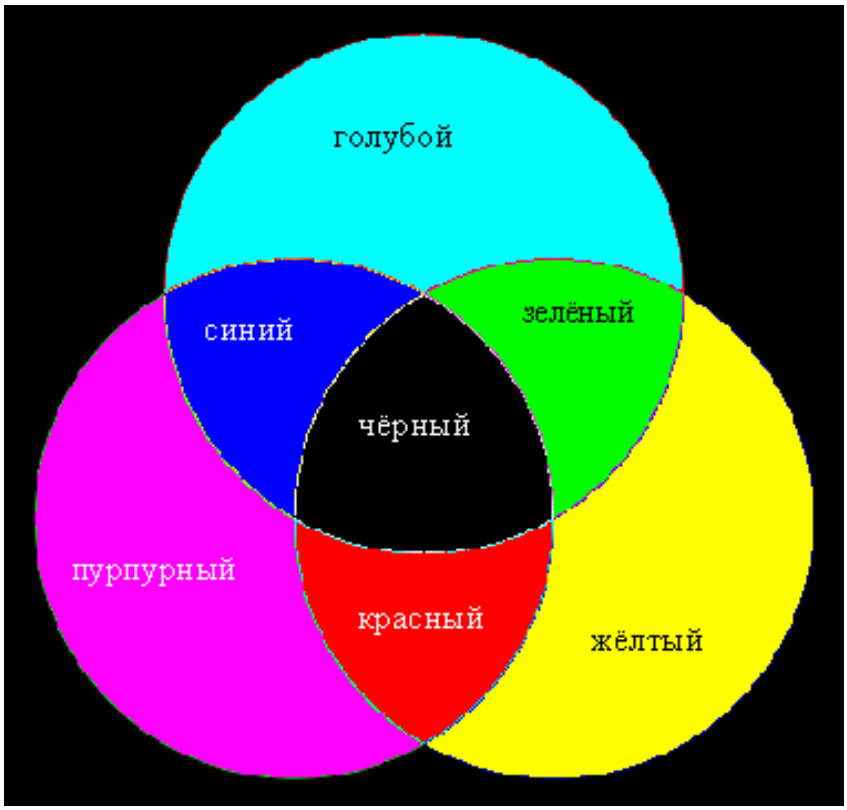
Базовими кольорами моделі CMY є кольори, ***додаткові до RGB***, які виходять в результаті ***віднімання основних кольорів RGB з білого.*** Звідси назва моделі ***субтрактивная*** (від англ. «to subtract» – віднімати). Базові кольори моделі CMY:

C	**CYAN**	блакитний = білий	червоний = зелений + синій
M	**MAGENTA**	пурпурний = білий	зелений = червоний + синій
Y	**YELLOW**	жовтий = білий	синій = червоний + зелений

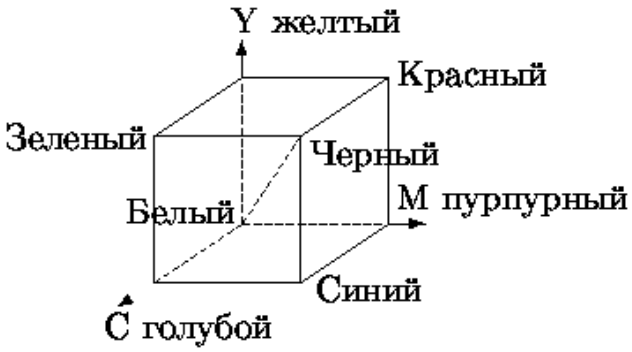
На мал. 2.6 показано, як різні комбінації блакитного, жовтого і пурпурного кольорів дають червоний, синій і зелений кольори.

Таким чином, система координат CMY – той же куб, що і для RGB, але з початком відліку в крапці, відповідній білому кольору. Колірний куб моделі CMY показаний на рис.2.7.

Базові кольори моделі CMYK можна представити за допомогою формул віднімання базових кольорів моделі RGB таким чином:



Мал. 2.6. Комбінації базових кольорів моделі CMYK



Мал. 2..7. Колірний куб моделі **CMY**

Cyan = RGB - R = GB = (0, 255, 255)
Yellow = RGB - B = RG = (255, 255, 0)
Magenta = RGB - G = RB = (255, 0, 255)

Зрозуміло, що віднімання з білого кольору білого будь-яка кількість разів дає в результаті чорний, а складання білих кольорів – білий:

RGB - RGB = RGB - RGB - ... - RGB = (0, 0, 0) - чорний колір
RGB + RGB = RGB + RGB + ... + RGB = (255, 255, 255) - білий колір

Віднімання з білого всіх базових кольорів CMYK дає білий:

RGB - (RGB - R) - (RGB - B) - (RGB - G) = RGB - RGB + R + У + G = RGB = (255,255,255).

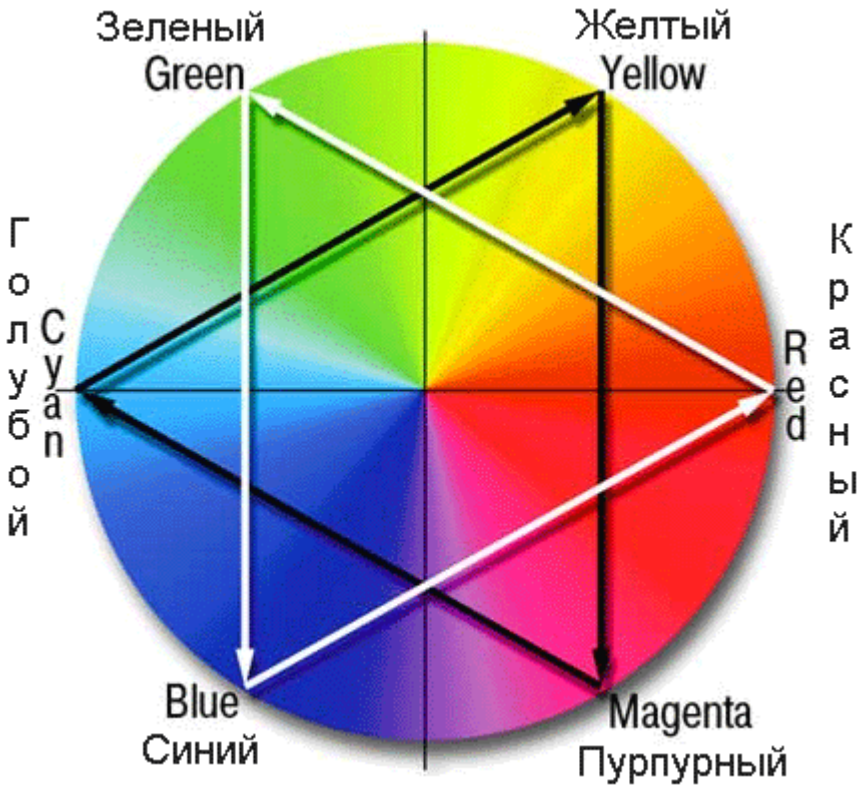
Складання базових кольорів CMYK дає в результаті чорний:

Cyan + Yellow + Magenta = RGB-R-B-G = (0,0,0).

Точніше, чорний колір повинен вийти теоретично, насправді ж із-за деяких особливостей друкарських фарб суміш всіх трьох основних кольорів дає брудно-коричневий тон, тому при друці зображення додається ще чорна фарба.

Субтрактівную колірну модель позначають абревіатурою **CMYK** (**Cyan** — блакитний, **Magenta** — пурпурний, **Yellow** — жовтий, **blacK** — чорний. Щоб не виникла плутанина з «Blue», для позначення «Black» використовується символ «К»).

Основні кольори моделей RGB і CMYK знаходяться в залежності, яку можна представити графічно за допомогою наступного малюнка.



Рiс.2.8. Колірний круг показує взаємозв'язок моделей RGB і CMY

Кожен колір розташований напроти доповнюючого його і між кольорами, за допомогою яких він отриманий. Щоб підсилити який-небудь колір, необхідно ослабити доповнюючий колір, розташований на протилежній стороні круга. Наприклад, щоб підсилити жовтий (Yellow), треба ослабити синій (Blue). На крузі кольорів жовтий розташований між зеленим (Green) і червоним (Red). Складання цих кольорів кольорів дає жовтий (Yellow).

Не всі кольори моделі CMYK можуть бути представлені в моделі RGB і навпаки. Колірний діапазон CMYK по кількості кольорів менше колірного діапазону RGB. Ця обставина має принципове значення, а не обумовлена тільки фізичними особливостями монітора, пристрою, що друкує, або фарб і полотна.

Перетворення малюнків з системи RGB в систему СМУК стикається з багаточисельними проблемами. Основна складність полягає в тому, що у них різна природа отримання кольорів, і те, що ми бачимо на екрані монітора, ніколи не можна точно повторити при друці.

Щоб отримати потрібний колір, краще використовувати яку-небудь систему підбору кольорів, наприклад Pantone. Pantone показує кожну з декількох тисяч кольорів у такому вигляді, як він буде надрукований на різних сортах паперу.

Модель HSB

Системи кольорів RGB і СМУК базуються на обмеженнях, які накладаються апаратним забезпеченням (моніторами комп'ютерів в разі RGB і друкарських фарб в разі СМУК). Більш інтуїтивним способом опису кольору є

представлення його у вигляді тону, насиченості і яскравості - система HSB (***Hue*** **–** тон або відтінок**, *Saturation* –** насиченість, ***Brightness* –** яскравість). Вона відома також як система HLS (Hue тон, Lightness освітленість, Saturation насиченість) або HSV (тон, насиченість, Volume величина) і ін.

HSB не строга математична модель, але вона дуже зручна для підбору відтінків і кольорів. Ця модель заснована на моделі RGB, але має іншу систему координат. Будь-який колір в моделі **HSB** визначається своїм колірним тоном (власне кольором), насиченістю (тобто відсотком доданої до кольору білої фарби) і яскравістю (відсотком доданої чорної фарби).

Перевага HSB перед іншими моделями полягає в тому, що ***вона більше відповідає природі кольору і добре узгоджується з моделлю сприйняття кольорів людиною.*** Тон є еквівалентом довжини хвилі світла, насиченість – інтенсивності хвилі, а яскравість – спільної кількості світла. Модель HSB краща, ніж RGB і CMYK, відповідає поняттю кольору, яке використовують професійні художники. У них зазвичай є декілька основних фарб, а всі інші виходять додаванням до них білої і чорної. Таким чином, потрібні кольори – це деяка модифікація основних фарб: освітлених або затемнених. Хоча художники і змішують фарби, але це вже виходить за рамки моделі HSB

Тон – це основний колір, який можна виділити в кольорі (довжина хвилі, яка переважає при випромінюванні).

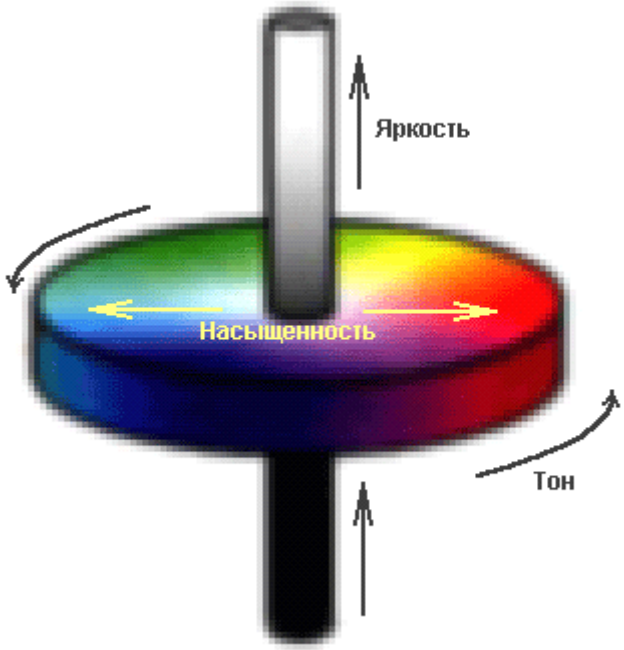
Насиченість кольору ***характеризує його «чистоту»:*** чим вона більша, тим колір "чистіший" (тобто ближче до тонової хвилі). Нульова насиченість відповідає сірому кольору, а максимальна насиченість – найбільш яскравому варіанту даного кольору. Можна вважати, що зменшення насиченості відповідає додаванню білої фарби. У білому кольорі насиченість дорівнює 0, оскільки неможливо виділити його колірний тон.

Яскравість розуміється як ступінь освітленості. При нульовій яскравості колір стає чорним. Максимальна яскравість при максимальній насиченості дають найбільш виразний варіант даного кольору. Можна вважати, що яскравість показує величину чорного відтінку, доданого до кольору. Яскравість чорного кольору – 0, а білого – 1. Чим більше чорної фарби додано, тим менше яскравість.

Недолік HSB полягає в тому, що для роботи на моніторах комп'ютерів її необхідно перетворювати на систему RGB, а для чотирьох кольорового друку – в систему СМУК.

Графічно модель HSB можна представити у вигляді кільця, уздовж якого розташовуються відтінки кольорів. Кожному відтінку відповідає свій градус, тобто всього налічується 360 варіантів (червоний — 0, жовтий — 60, зелений — 120 градусів і так далі). На зовнішньому краю круга знаходяться чисті спектральні кольори або колірні тони (параметр ***H вимірюється в кутових градусах***, від 0 до 360). Чим ближче до центру круга розташований колір, тим менше його насиченість, тим він більш бляклий, пастельний (параметр ***S вимірюється у відсотках***). Яскравість (освітленість) відображується на лінійці, перпендикулярній плоскості колірного круга (параметр ***B вимірюється у відсотках***). Всі кольори на зовнішньому крузі мають максимальну яскравість.

Така колірна модель набагато ***бідніше RGB***, оскільки дозволяє працювати всього лише з 3 млн. кольорів.



Мал. 2.9. Графічне представлення моделі HSB

Модель HSB не є орієнтованою ні на який технічний пристрій відтворення кольорів, тому її називають ***апаратно незалежною.***

Палітра

Палітра (palette) – набір кольорів, використовуваних в зображенні або при відображенні відеоданих. Палітру можна сприймати як таблицю кодів кольорів (зазвичай у вигляді RGB-троек байтів в моделі RGB). Палітра встановлює взаємозв'язок між кодом кольору і його компонентами у вибраній колірній моделі. Палітра може належати зображенню, частці зображення, операційній системі або відеокарті. Якщоі використовуваний колір не входить до палітри, він замінюється найближчим кольором, занесеним в неї.

Кольоророзподіл при друці

Отже, система RGB працює з випромінюваним світлом, а CMYK – з відбитим. ***Діапазон RGB ширший за діапазон CMYK***. Це означає, що кольори, створені на екрані, не завжди можна відтворити при друці. Якщо необхідно роздрукувати на принтері зображення, отримане на моніторі, спеціальна програма виконує перетворення однієї системи кольорів в іншу. Але в системах RGB і CMYK різна природа отримання кольорів. Тому колір, який ми бачимо на моніторі, досить важко точно повторити при друці. Зазвичай на екрані колір виглядає декілька яскравіше в порівнянні з тим же самим кольором, виведеним на друк.

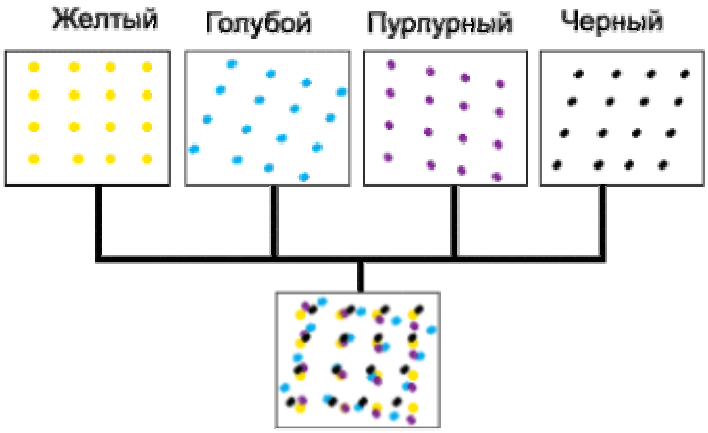
У друкарнях ***процес чотирибарвного друку*** можна розділити на два етапи:

1. Створення на базі початкового малюнка чотирьох складових зображень: блакитного, пурпурного, жовтого і чорного кольорів.
2. Друк кожного з цих зображень одного за іншим на одному і тому ж листі паперу.

Розподіл кольорового малюнка на чотири компоненти виконує спеціальна **програма кольоророзподілу**.

При цьому важливо, що замість суцільних кольорових областей програма кольоророзподілу створює растри з окремих крапок (мал. 2.10), причому ці точкові растри злегка повернені один щодо одного так, щоб точки різних кольорів не накладалися одна поверх іншої, а розташовувалися поруч.

Маленькі точки різних кольорів, близько розташовані один до одного, здаються такими, що зливаються разом. Саме так наші очі сприймають результуючий колір.



Мал. 2.10. Точкові растри для чотирибарвного друку

У таблиці 2.1 для прикладу представлений опис декількох кольорів з використанням моделей RGB, CMYK і HSB (діапазон зміни складових кольору — від 0 до 255).

Таблиця 2.1. Значення основних кольорів різних моделей

Колір	RGB	CMYK	HSB
Червоний	255,0,0,	0,255,255	0,240,120
Жовтий	255,255,0	0,0,255	40,240,120
Зелений	0,255,0	255,0,255	80,240,120
Бірюзовий	0,255,255	255,0,0,	120,240,120
Синій	0,0,255	255,255,0	160,240,120
Фіолетовий	255,0,255	0,255,0	200,240,120
Чорний	0,0,0	255,255,255	160,0,0
Сірий	70,70,70 125,125,125 193,193,193	70,70,70 193,193,193 125,125,125	160,0,59, 160,0,120, 160,0,180
Білий	255,255,255	0,0,0	160,0,240

Закони Грассмана

В середині XIX століття німецький вчений Герман Грассман сформулював три закони адитивного синтезу кольору.

Перший закон (тривимірності). Будь-який колір однозначно виражається трьома лінійно незалежними кольорами. Лінійна незалежність полягає в тому, що жоден з цих трьох кольорів не можна отримати складанням двох останніх.

Другий закон (безперервності). Колірний простір є безперервним. Якщо в суміші трьох кольорів один безупинно змінюється, а інші залишаються постійними, то колір суміші мінятиметься безперервно. Не існує такого кольору, до якого не можна було б підібрати нескінченно близький.

Третій закон (адитивності). Для суміші двох кольорів і завжди має місце рівняння:

$$Z1+Z2=(R1+R2)+ (G1+G2)+ (B1+B2).$$

Представлення графічних даних. Формати графічних файлів

Формат графічного файлу – це спосіб представлення графічних даних на зовнішньому носієві.

Єдиного формату графічних файлів, придатного для всіх застосувань, ***не існує***, проте деякі формати стали стандартними для цілої лави наочних областей. Слід розрізняти векторні (WMF, DXF, CGM і ін.) і растрові (TIFF, GIF, JPG і ін.) формати.

Нагадаємо, що файли векторного формату містять описи малюнків у вигляді набору команд для побудови простих графічних об'єктів (ліній, кіл, прямокутників, дуг і т. д.). У файлах растрового формату запам'ятовується розмір зображення, бітова глибина і колір кожного відеопікселя малюнка.

Для зменшення розміру файлу з растровим зображенням використовуються різні алгоритми стиснення графічних даних, тобто зменшення розміру файлу за рахунок зміни способу організації даних в нім. Кожен алгоритм стиснення добре стискує тільки зображення певної структури. Наприклад, у форматі PCX застосовується алгоритм стиснення, який добре працює з малюнками, що містять великі області однотонного зафарбовування. Зберігання ж відсканованих фотографій у форматі PCX не виправдане, оскільки розміри файлів, що виходять, дуже великі. В цьому випадку краще скористатися форматом JPEG, який заснований на теорії фрактальної упаковки і забезпечує високий коефіцієнт стиснення для зображень фотографічної якості.

Формат файлу визначається по його розширенню. Тому в більшості випадків позначення формату і розширення збігаються.

Зупинимося докладніше на деяких форматах (див. таб. 2.2, 2.3).

Таблиця 2.2. Векторні формати графічних файлів

Назва формату	**Програми, які можуть відкривати файли**
WMF(Windows MetaFile)	Більшість застосувань Windows.
EPS(Encasulated PostScript)	Більшість настільних видавничих систем і векторних програм, деякі растрові програми.
DXF(Drawing Interchange Format)	Всі програми САПР, багато векторних редакторів, деякі настільні видавничі системи.
CGM(Computer Graphics Metafile)	Більшість програм редагування векторних малюнків, САПР і видавничі системи.

Таблиця 2.3. Растрові формати графічних файлів

Назва формату	**Програми, які можуть відкривати файли**
BMP(Windows Device Independent Bitmap)	Всі програми Windows, які використовують растрову графіку.
PCX (Z – Soft PaintBrush)	Майже всі графічні застосування.
GIF(Graphic Interchange Format)	Майже всі растрові редактори; більшість видавничих пакетів; векторні редактори, що підтримують растрові об'єкти.
TIFF (Tagged Image File Format)	Більшість растрових редакторів і настільних видавничих систем; векторні редактори, що підтримують растрові об'єкти.
TGA(TrueVision Targa)	Растрові редактори
IMG(Digital Research GEM Bitmap)	Деякі настільні видавничі системи і редактори зображень Windows
JPEG(Joint Photographic Experts Group)	Останні версії растрових редакторів; векторні редактори, що підтримують растрові об'єкти.

TIF. При збереженні ілюстрації в цьому форматі не використовується жоден з видів стиснення. У цьому форматі отримують максимально можливу якість зображення. Це єдиний формат, використовуваний в професійному дизайні для зберігання зображень високої якості. TIF-изображения можуть

займати декілька сотень мегабайт. TIF-формат є кращим вибором при передачі зображень з растрової графіки у векторні програми і видавничі системи. А ось тримати «домашній фотоальбом» в цьому форматі безрозсудно: якісні TIF-изображения можуть займати декілька сотень мегабайт!

JPG. Використовується для стиснення зображення в десятки разів. Дозволяє використовувати різні ступені стиснення, вибираючи між якістю і розміром файлу. У професійній поліграфії цей формат не використовується із-за істотних втрат якості зображення. Для проглядання зображення на екрані монітора або для роздруку на принтері якості JPG-формата вистачає. У форматі JPG використовується метод стиснення JPEG. Цим методом краще стискуються растрові зображення фотографічної якості і погано стискуються логотипи або схеми. У цьому форматі добре і з меншими втратами стискуються великі зображення з високим доздільною здатністю 200-300 ppi і погано стискуються з низькою здатністю 72-150 ppi. Небажано зберігати зображення в JPG-формате, де важливі всі тонкощі перенесення кольорів, оскільки під час стиснення відбувається вікидання деякої колірної інформації. У цьому форматі слід зберігати тільки кінцевий варіант роботи, тому що будь-яке пересохранение приводить до нових втрат даних і перетворень зображення в кашу.

GIF. Формат растрової графіки, створений спеціально для Інтернету. Використовує метод стиснення, що позначається LZW. Має обмежену палітру кольорів не більше 256. Тому кольори в цьому форматі стають грубими, а само зображення зернистим. Не використовується в поліграфії і не рекомендується для зображень, призначених для монітора або принтера. Але якщо зображення однотонне або контрастне і з чітким кордоном між кольорами, то використання GIF дає більший ступінь стиснення, чим JPG, причому якість не змінюється.

PNG. Це формат, розроблений відносно недавно, призначений для того, щоб замінити GIF-формат. У нім використовується метод стиснення без втрат якості, який позначається DEFLATE. Стислі індексовані файли (з невеликою кількістю кольорів) мають менший розмір в порівнянні з аналогічними GIF-файлами. Глибина кольору у файлах може бути будь-якій до 48 біт. Дозволяє добитися однакового відображення інформації незалежно від апаратури користувача.

EPS. Найзручніший і універсальний спосіб зберігання графічних даних. Використовує спрощену версію мови PostScript, тому не може містити у файлі більш за одну сторінку і не зберігає лаву установок для принтера. Призначений для передачі векторних і растрових зображень у видавничі системи. Створюється всіма програмами, що працюють з графікою. Цей формат використовується тільки тоді, коли друк здійснюється на пристрої, що підтримує мову PostScript. Відкрити EPS-файл для редагування можуть тільки програми фірми Adobe – Photoshop, Illustrator. Решта графічних програм може відкривати тільки в режимі перегляду. Якщо EPS-файл роздрукувати на принтері, який не підтримує мову PostScript, то надрукований буде тільки ескіз – копія з низькою роздрібністю.

PDF. Це не залежний від графічних програм формат для створення електронної документації, презентацій, а також для передачі графіки через мережі. PDF-файлы створюються шляхом конвертації з PostScript-файла або функцією експорту. Програми Photoshop, Illustrator можуть створювати тільки односторінковий файл PDF. Всі дані у форматі PDF можуть стискуватися. Причому до різного типа інформації застосовуються різні типи стиснення. Файл PDF може бути оптимізований – з нього віддаляються елементи, що повторюються, встановлюється посторінковий лад завантаження сторінок з пріоритетом спочатку для тексту, потім для графіки. Формат PDF використовується для передачі по мережах в компактному виді графіки і тексту. Найбільш зручним засобом для роботи з PDF-файлами є програма Acrobat. Причому є 2 варіанти цієї програми: Acrobat Professional (для створення багатосторінкових файлів) і Acrobat Reader (для проглядання PDF-файлов).

PSD. Внутрішній формат програми Photoshop. Почав підтримуватися великою кількістю графічних програм.

AI. Внутрішній формат програми Illustrator. Може відкриватися програмою Photoshop. Його підтримують всі програми, пов'язані з векторною графікою. Є кращим засобом при передачі векторних зображень з однієї програми в іншу. Растрові графічні елементи при передачі через AI-формат в більшості випадків втрачаються.

CDR. Внутрішній формат програми Corel Draw. У форматі CDR можуть міститися і растрові графічні об'єкти. У цьому форматі застосовується стиснення, причому для векторних і растрових файлів різне .

WMF. Векторний формат, який використовується графічними програмами ОС Windows. Служить для передачі векторних зображень через буфер обміну в середі Windows. Цей формат приймається практично всіма програмами, що працюють з векторною графікою. Використовувати цей формат для растрових зображень не можна. Недоліки: спотворення кольору і незбереження лави параметрів, які встановлюються для зображень в графічних програмах.

BMP. Растровий формат, який є рідним графічним форматом Windows. Підтримується всіма редакторами. У цьому форматі зберігаються невеликі растрові зображення, призначені для використання в системі Windows. Це формат невисокої якості і з низьким ступенем стиснення. Його не рекомендується використовувати ні для web-дизайна, ні для передачі.

PCX. Найвідоміший формат. Його підтримує практично будь-яка програма, що працює з графікою. Формат PCX підтримує метод стиснення RLE. Використовується для штрихованих зображень і для зображень з невеликою глибиною кольору.

Тема 3. ДІЛОВА ГРАФІКА. ГРАФІЧНІ ЗОБРАЖЕННЯ В ДІЛОВИХ ДОКУМЕНТАХ

Ділова графіка – група зображень, широко вживана в бізнесі, економіці, управлінні, науці, техніці і інших галузях людської діяльності. Основне призначення ділової графіки – розкрити вміст висловлюваного матеріалу засобами легко сприйманих наочних образів, представити дані в ясній і легко аналізованій формі. Один з напрямів ділової графіки автоматизація створення типових документів.

1. Спільні відомості про форми MS WORD

Форми це типові бланки різних документів (договорів, анкет, контрактів, рахунків-фактур, платіжних доручень-вимог і тому подібне).

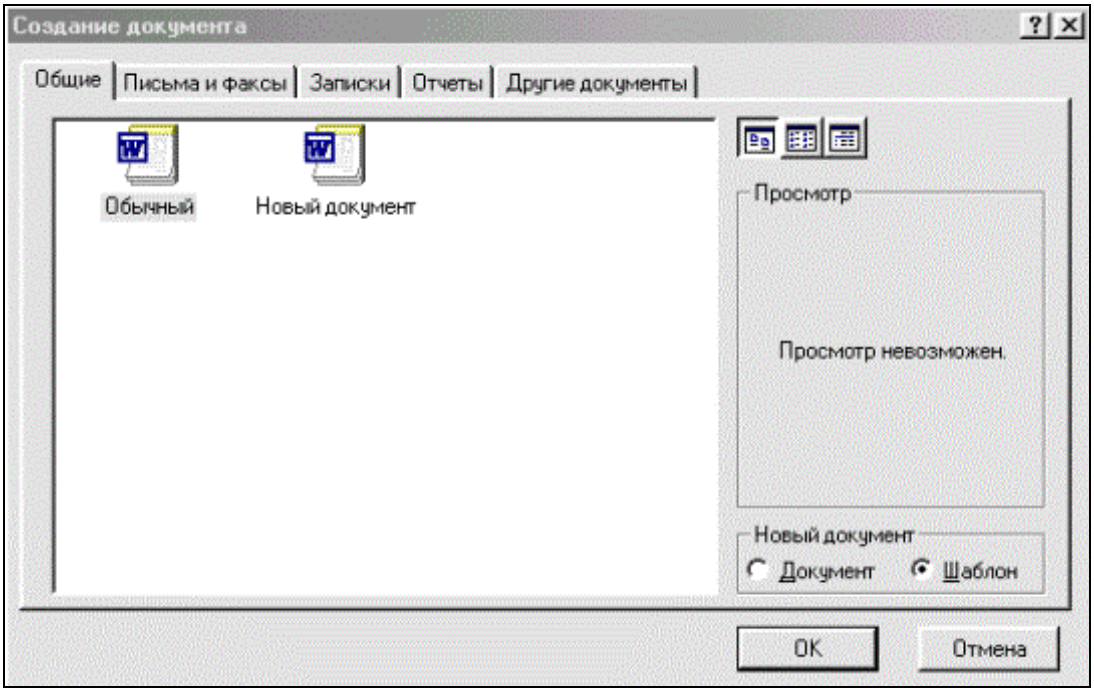
На будь-якому бланку, окрім стандартного тексту, є порожні поля, які заповнює користувач. У Word'е такі поля називають полями форми. Їх існує три типи: текстове поле форми, прапорець-перемикач, список, що розкривається. Включаючи ці поля в таблицю, створюємо сітчасті (тобто табличні) форми, а, вставляючи їх в текст документа, отримуємо текстові форми. Створивши і зберігши форму, можна нею користуватися в друкарському вигляді, заповнюючи вміст полів уручну або в інтерактивному режимі роботи з документом. Можна надрукувати тільки вміст полів або тільки структуру форми – бланк для заповнення.

2. Технологія створення форми

1 крок. Розробити вид форми (текстова або сітчаста). Створити шаблон документа, в якому розмістити структуру форми (команда меню *Файл/создать*, включити прапорець Шаблон, натиснути кнопку Ok (мал. 3.1).

2 крок. Вставити поля форми відповідних типів. Для кожного поля форми задати його розмір, ввести встановлювані за умовчанням значення, вказати вид інформації, що зберігається, підключити макроси, які виконуватимуться при вході в полі або при виході з поля, ввести довідку до поля.

3 крок. Захистити шаблон і зберегти його. Поля форми можна розставляти поступово, у міру підготовки тексту, або все відразу після набору тексту. У останньому випадку, там, де має бути поле форми, необхідно вставити 2 пропуски, тоді при заповненні поля його вміст автоматично відділятиметься від основного тексту з кожного боку одним пропуском.

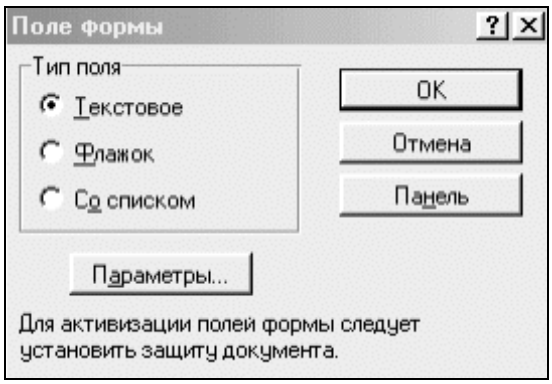


Мал. 3.1. Діалогове вікно ***Створити***

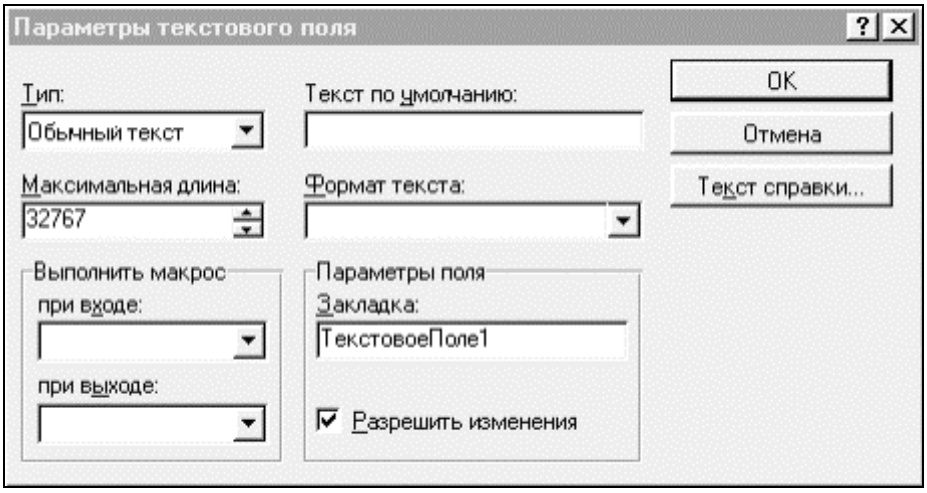
2.1. Вставка текстових полів форми

1 спосіб. Виконати команду меню *Вставка/Поле форми*. Відкриється діалогове вікно *Поле форми* (рис.3.2), в якому вказати типа поля, включивши відповідний прапорець – *текстове*, потім натискувати кнопку *Параметри.* Відкривається діалогове вікно *Параметри текстового поля* форми (рис.3.3).

2 спосіб (основний, зачинаючи з в Word 97). Вивести на екран панель інструментів *Форми* (команда меню *Від/панелі інструментів*, вказати *Форми*). Натиснути на панелі інструментів кнопку *Текстове поле* і мишею вказати місце його розташування в документі, потім натинути кнопку *Параметри*. Відкривається діалогове вікно *Параметри текстового поля форми* (мал. 3.3).



Рic.3.2. Вікно ***Поле форми***



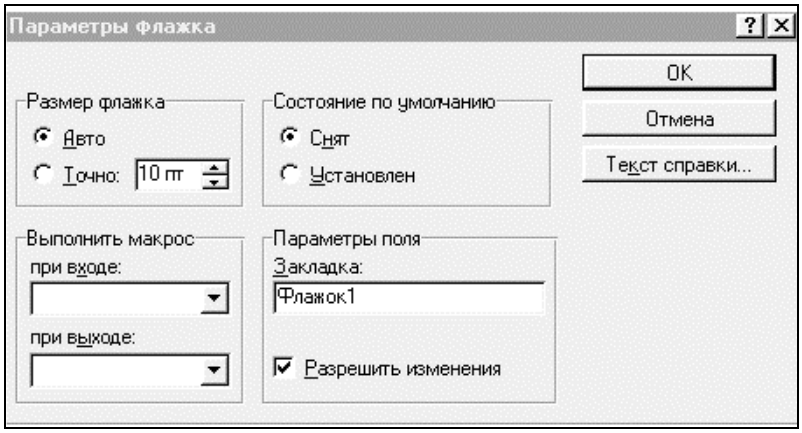
Рiс.3.3. Діалогове вікно ***Параметри текстового поля***

У списку *Тип* слід вказати потрібний тип даних текстового поля: *Звичайний текст* зберігає символи, числа, слова; *Число* зберігає тільки числа; *Дата* сприймає тільки дати; *Поточна дата* вставляє поточну дату; *Поточний час* вставляє поточний час; *Обчислення* підраховує значення відповідно до формули, що вставляється. Можна вказати обмеження довжини поля, підключити макрокоманди, які виконуватимуться при вході в полі або виході з нього; визначити текст за умовчанням; вказати формат відображення вмісту поля, визначити доступне поле для редагування або ні, включивши або вимкнувши прапорець *Дозволити зміни.*

Кожне поле форми має ім'я *закладку*, яке призначається полю автоматично при його вставці, а потім використовується, наприклад, "Текстове Поле1". Ім'я закладки можна змінити в діалоговому вікні *Параметри поля*. Пропуски в іменах не допускаються, перший символ – завжди буква. Після визначення параметрів текстового поля форми натиснути *Ok*.

2.2. Вставка перемикачів у форму

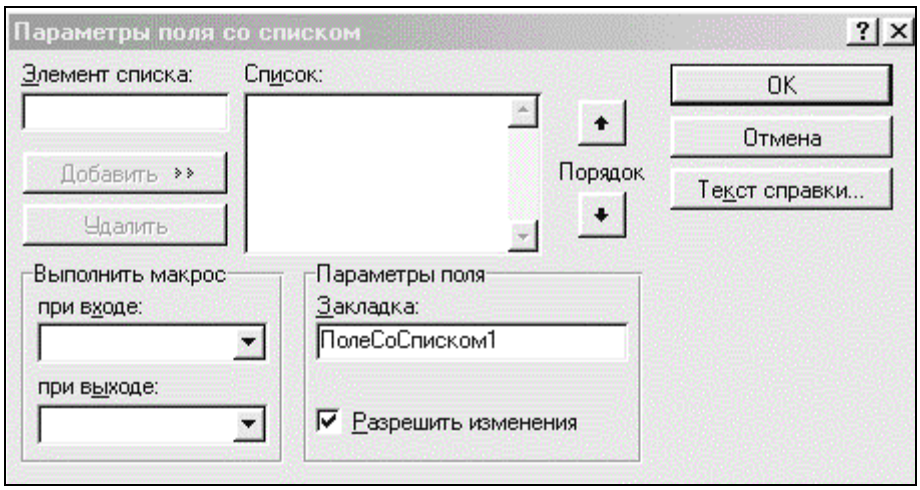
Якщо у формі є питання, відповіді на яких - "Та", "Ні", потрібно використовувати прапорці-перемикачі. Команда меню *Вставка/Поле форми*, в розділі *Тип* вказати *Прапорець* (рис.3.2), потім натиснути кнопку *Параметри*, з'являється діалогове вікно *Параметри прапорця* (рис.3.4), в якому можна встановити: розмір по висоті навколишнього тексту; включений або вимкнений за умовчанням; чи може користувач змінювати стан перемикача, змінювати ім'я закладки; призначити макрос при вході і виході з поля. При натисненні кнопки *Ok* вікно *Параметри* закривається, призначені параметри зберігаються.



Рiс.3.4. Діалогове вікно ***Параметри прапорця***

2.3. Вставка полів введення із списком

Це поле дозволяє вибрати потрібний варіант із списку, а не вводити вміст уручну. Можна визначити 25 можливих варіантів списку. Команда меню *Вставка/Поле форми*, в розділі *Типа* вказати: *Із списком* (рис.3.2), потім натиснути кнопку Параметри, з'являється діалогове вікно *Параметри поля із списком (*рис.3.5).



Рiс.3.5. Діалогове вікно ***Параметри поля із списком***

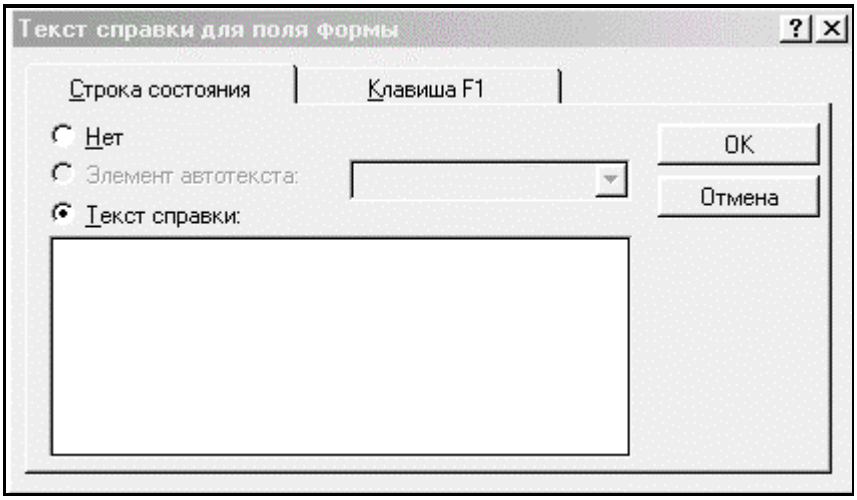
Для заповнення поля *Список* (елементів) треба в полі *Елемент списку* ввести варіант і натиснути кнопку *Додати* і так далі (максимально можливо додати 25 варіантів). Для видалення варіанту з поля *Список* елементів треба його виділити мишею і натиснути кнопку *Видалити*. Після внесення варіанту в полі *Список* елементів його редагувати не можна, можна тільки видалити і внести заново.

Для впорядкування елементів списку використовувати кнопки *Лад (Зрушення)* – на одну позицію вгору або вниз зрушує виділений елемент. Якщо включений прапорець ☑ *Вирішити зміни (Поле доступне)*, то користувач може вибирати варіанти, інакше доступний лише 1-й варіант, який заданий завжди за умовчанням. Решта властивостей визначається аналогічно. Кнопка *Ok* зберігає призначені властивості поля.

2.4. Створення довідки для поля форми

Щоб правильно заповнити форму в процесі використання, кожному полю можна визначити довідковий текст, який виводитиметься при вході у форму в рядку стану або при натисненні клавіші *F1*. У останньому випадку довідка виводиться в інформаційному вікні.

Щоб створити довідку поля, треба заздалегідь його виділити і відкрити діалогове вікно *Параметри поля форми*, в якому натиснути кнопку *Текст довідки*. З'являється діалогове вікно *Текст довідки для поля форми* (рис.3.6). Вікно має дві вкладки (сторінки): *Рядок стану* і *Клавіша F1*.



Рiс.3.6. Вікно ***Текст довідки*** для поля форми

Якщо вибрати вкладку *Рядок стану*, за умовчанням включається прапорець *Немає*, що означає: немає довідки до поля. Можна вибрати *Елемент автотексту*, якщо використовується один і той же текст довідки для декількох полів. В цьому випадку автотекст спочатку має бути створений (меню *Правка/автотекст/создать*). Можна вибрати прапорець *Текст довідки*, нижче активізується поле для введення довільного тексту довідки, який з'являтиметься в рядку стану при вході в полі.

Вкладка *Клавіша F1* має такий же вигляд і можливості. Текст довідки виводиться в інформаційне вікно при натисненні клавіші *F1*.

Текст довідки в рядку стану може містити максимально 138 символів, в інформаційному вікні - 255 символів.

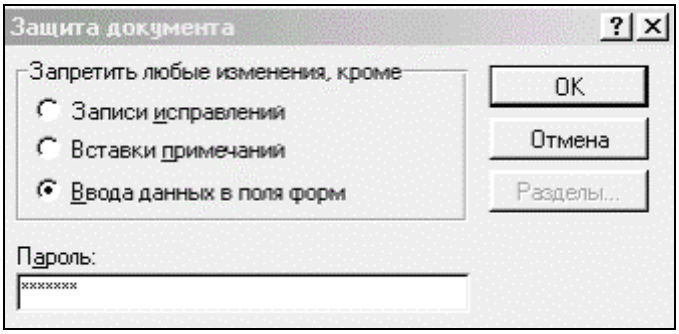
2.5. Захист форми

Створивши форму, її потрібно перевірити, а потім зберегти як шаблон. Обом операціям повинен передувати захист форми, після якої поля активізуються і форма готова до використання.

Існує два рівні захисту: з паролем і без нього, а також можна захистити окремі частки (розділи) форми.

Щоб встановити захист, треба виконати команду меню *Сервіс/установіть захист* (або натиснути відповідну кнопку панелі інструментів *Форма*), з'являється

діалогове вікно *Захист документа* (рис.3.9). Треба вказати: *Введення даних в поля форм*, при необхідності ввести пароль, натиснути *Ok*. Захист без пароля не дуже надійний, він дозволить будь-якому користувачеві відкрити форму і зняти захист (меню *Сервіс/снять захист* або "віджати" кнопку *Захист форми* на панелі інструментів *Форми*).



Рic.3.9. Вікно ***Захист документа***

Для зміни опцій (властивостей) полів форми потрібно зняти захист, потім виділити потрібне поле, відкрити вікно *Параметри поля форми*, натиснути кнопку *Параметри (Властивості)* на панелі інструментів *Форми*.

2.6. Збереження шаблону форми

Кожен документ Word базується на певному шаблоні. Шаблон – це зразок для підготовки типових документів. У шаблоні Word зберігає: стилі оформлення, макрокоманди, елементи тексту і автотексту, індивідуальні меню і панелі інструментів користувача. Розрізняють глобальні шаблони, шаблони користувача і шаблони робочої групи (спільні шаблони).

За умовчанням новий документ створюється на основі глобального шаблону Normal.dot: об'єкти, що все зберігаються в нім, доступні у будь-який час в будь-якому документі і шаблоні. Normal.dot завантажується кожного разу при запуску Word.

Шаблони користувача і шаблони робочої групи розрізняються тільки містом зберігання, залежно від цього їх можна використовувати по-різному. Шаблони користувача – це шаблони типових документів (бланки, форми Word), які користувач створив і зберігає на своїй робочій станції. Шаблони робочої групи зберігаються на мережевому диску і доступні всім, хто має можливість відкрити теку, в якій вони знаходяться.

Після захисту форми потрібно зберегти її як шаблон: меню *Файл/сохраніть* (або *Зберегти як*). У діалоговому вікні *Збереження документа* (рис.3.10) ввести ім'я шаблону, в полі *Тип файлу* вже вибраний тип: *Шаблон документа*, натиснути *Зберегти*.

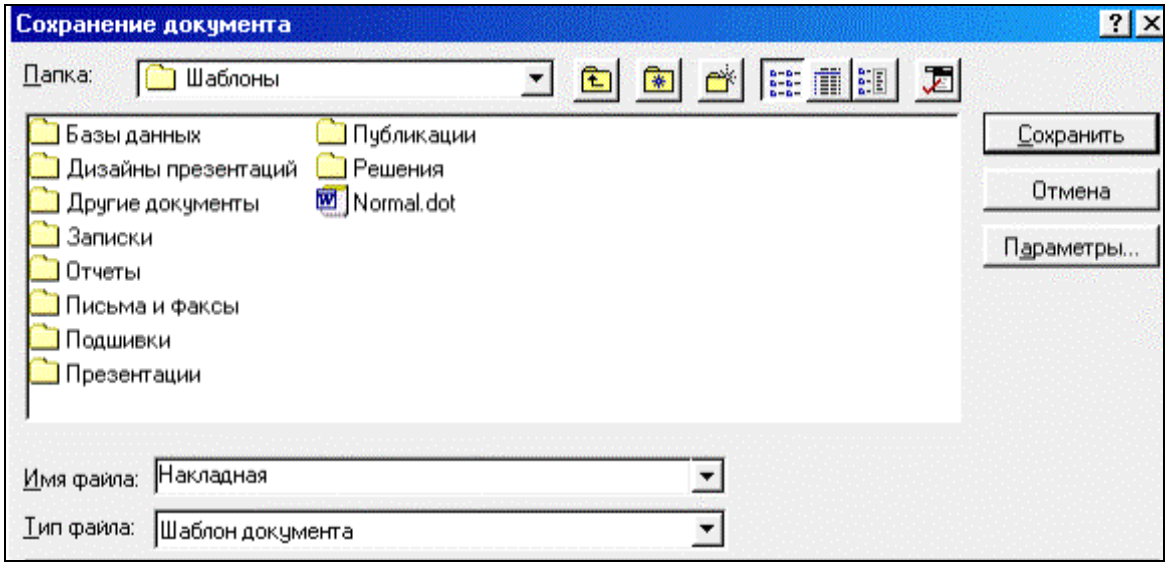


Рис.3.10. Діалогове вікно ***Збереження документа***

Вибір теки для зберігання шаблонів користувача, на основі яких створюються документи (у діалоговому вікні *Створити*), здійснюється в меню *Сервіс/параметри*, на вкладці *Розташування* діалогового вікна *Параметри* (рис.3.11). У списку *Типів файлів* слід вибрати елемент *шаблони користувача* або *спільні шаблони* (для шаблонів робочої групи), натиснути кнопку *Змінити*. Відкриється вікно *Зміна розташування*, в якому вказати шлях до своєї теки, наприклад C:\Documents and Settings\User\Application Data\Microsoft\Шаблоны\ лаба по формах.

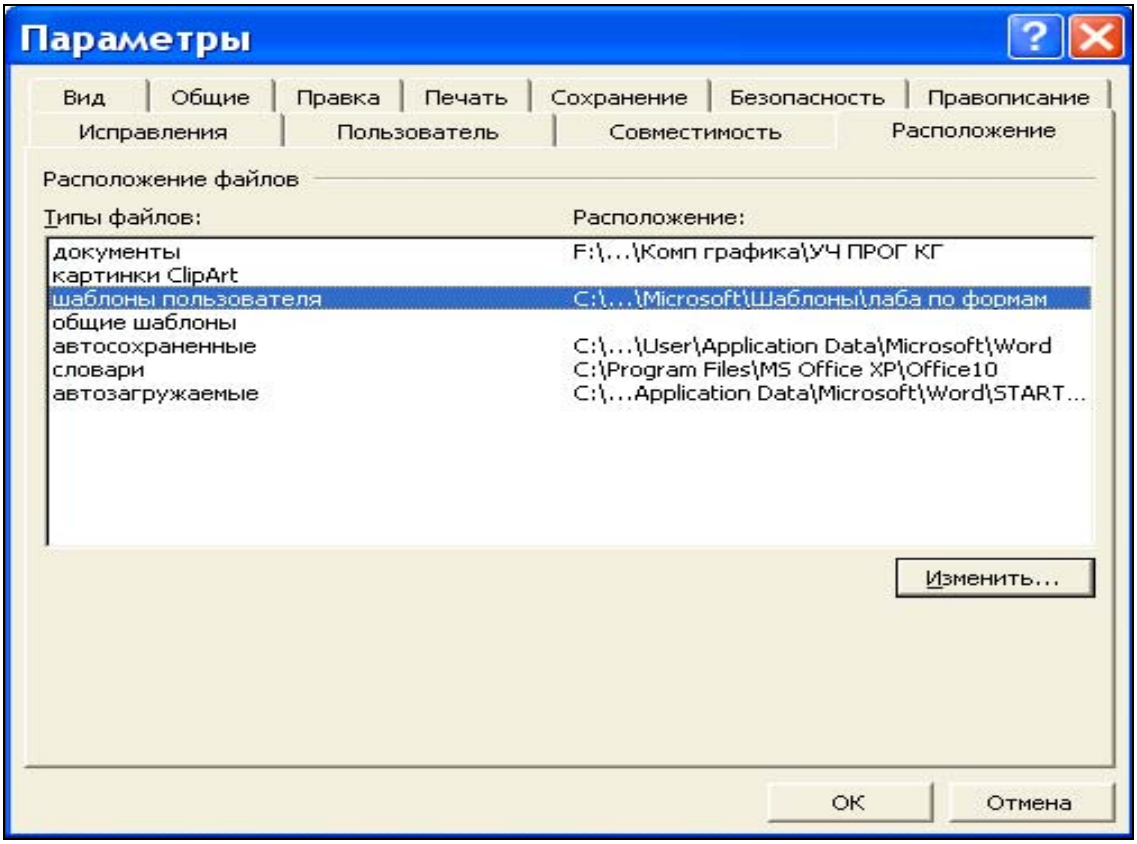


Рис.3.11. Діалогове вікно ***Параметри***

2.7. Заповнення форми в інтерактивному режимі

Щоб створити новий документ, заснований на шаблоні форми, слід виконати команду меню *Файл/создать*, із списку *З шаблоном* вибрати шаблон форми, при цьому має бути включений прапорець *Документ*, натиснути *Ok*.

Якщо при розробці форми додано затінювання полів, то поля виділяються сірим фоном в документі (меню *Сервіс/параметри*, вкладка *Вигляд*, в полі із списком Затінювання *полів,* вибрати Завжди або При *виділенні)*

Для переміщення курсора введення по полях форми можна використовувати клавішу *Tab* і *Enter*, клавіші управління курсором і мишу.

У текстовому полі з типом даних *Обчислення* вводяться числа, а не букви.

Клавіша *F1* або рядок стану виводять підказку для активного поля.

У полі типа *Прапорець* для установки значення використовується миша або клавіша *Пропуск*.

У полі типа *Список* вибір елементу здійснюється мишею або клавішею *Tab* і *Enter*. Закрити список без вибору - *Esc*.

Зберегти форму можна повністю: структуру плюс дані або тільки дані. Використовується меню *Сервіс/параметри*, вкладка *Збереження*, включити або відключити (за умовчанням відключений) прапорець *Зберігати тільки дані для форм*.

Після заповнення форми її потрібно зберегти як звичайний документ (кнопка *Зберегти* на панелі інструментів *Стандартна*).

Роздрукувати форму можна трьома способами:

1) як порожній бланк і заповнити його на папері: текстові поля заздалегідь заповнити пропусками, оскільки розмір поля залежить від довжини введеного тексту (діалогове вікно *Параметри текстового поля Текст за умовчанням*), потім виконати команду меню *Файл/Друк*);

2) друк форми, заповненої в діалоговому режимі: меню *Сервіс/параметри*, на вкладці *Друк* відключити прапорець *Друкувати тільки дані для форм*, натиснути кнопку *Друк* на панелі інструментів *Стандартна*;

3) друк на бланках: включити прапорець *Друкувати тільки дані для форм*. Бланки спочатку потрібно приготувати (див. 1) п.).

Приклад. Розробка сіткової форми "Накладна". Розглянемо процес створення бланка накладної (рис.3. 12):

1. Створення накладної починаємо із створення шаблону: меню *Файл/создать*, включити прапорець *Шаблон, Ok*. Потім вставимо таблицю виду рис.3. 13: меню *Табліца/вставіть таблицю* вказати: число стовпців - 5, число рядків - 14, натиснути *Ok*. З'явиться порожня таблиця.

2. Встановити курсор всередину таблиці і виконати команду меню *Табліца/виделіть таблицю*. Відкрити список *Розмір шрифту* на панелі інструментів *Форматування* і вибрати 8 пт, в списку *Шрифт* вибрати гарнітуру шрифту Times New Roman Cyr.

3. Позиціюючи курсор мишею в потрібне вічко, внесемо: у С2 - "Накладна №", у А3 - "Кому", в А4 - "От кого", в А6 - "№ п/п", у В6 - "Найменування", в С6 -

"Кількість", в D6 - "Ціна", в E6 - "Сума", в D12 - "Разом:", у B13 - "Здав", в C13 - "Прийняв", у B14 - "М.П.".

20 июнь, 2001

Накладная № _____

Кому_____

От кого_____

№ п/п	Наименование	Количество	Цена	Сумма
			Итого:	

Сдал Принял

М.П.

Рiс.3.12. Бланк накладної

		Накладная №		
Кому				
От кого				
№ п/п	Наименование	Количество	Цена	Сумма
			Итого:	
	Сдал	Принял		
	М.П.			

Рiс.3.13. Перший етап створення накладної

4. Позиціювати курсор у комірку D1 і виконати команду меню *Вставка/Поле форми*, вказати *текстове поле*, в діалоговому вікні *Параметри текстового поля* в полі *Типа* вказати *Поточна дата*, в списку *Формат тексту* вибрати потрібний.

5. Виділити вічко C2, збільшити розмір шрифту 14 пт. Виділити діапазон вічок B3:e3, виконати команду меню *Табліца/об'єдініть вічка*. Так само об'єднати вічка B4:e4.

6. Вставити текстові поля форми у комірки D2, B3, B4, A7:E11, E12. У комірки E7:E11 вставити текстове поле типа *Обчислення*, внести формулу = C7* D7 (у комірку E7) і так далі, що обчислює суму як добуток ціни на кількість. У комірку E12 ввести формулу = Sum(above) – суму всіх елементів таблиці, розташованих зверху. У комірки D7:E11 вставити текстове поле типа *Число*, вибрати формат - ###0,00р.; (###0,00р.). Вказаний формат означає, що в представленні числа використовується два знаки після коми, негативні значення полягають в дужки.

7. Виділити вічка A6:E6, натиснути на панелі інструментів *Форматування* кнопку *Вирівнювання по центру*.

8. Виділити вічка D2:E2 і на панелі інструментів *Форматування* відкрити список *Кордону*, вибрати вид обрамлення *Знизу*. Так само вибрати вид обрамлення знизу для вічок B3, B4.

9. Виділити діапазон вічок A6:E12 і на панелі інструментів *Форматування* відкрити список *Кордону*, вибрати вид обрамлення *Сітка*.

10. Вставимо у форму друк. Додамо об'єкт WordArt кнопкою *Додати об'єкт WordArt* в панелі *Малювання.* Діалогове вікно *Текст* призначено для введення і редагування тексту, що вставляється. Внесемо: Іванов Іван Іванович. У панелі інструментів WordArt кнопкою відкриваємо список *Форма* і вибираємо вид розміщення тексту. Слова розташуються по колу і в центрі. Вибираємо тип шрифту, його розмір, натискуємо кнопку *Тінь, Ok*. Потім, використовуючи панель інструментів *Малювання*, можна укласти текст у вкладені один в одного кола. Перемістимо об'єкт мишею в потрібне місце.

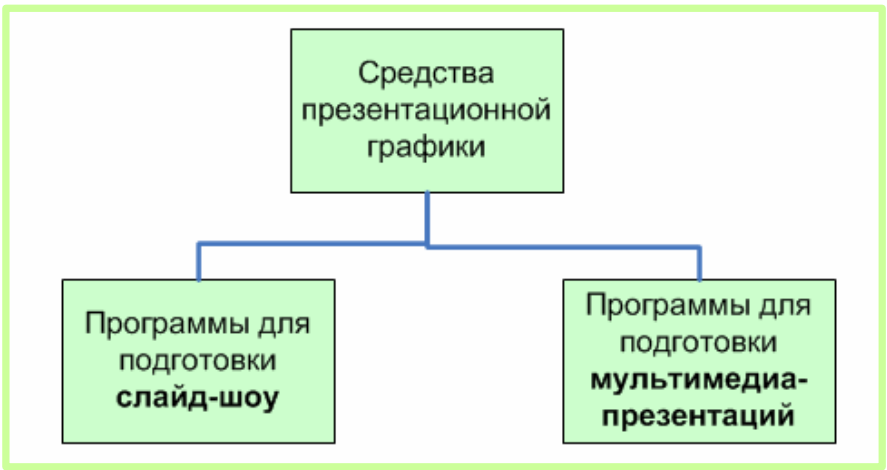
Форма готова. Зберігаємо форму у вигляді шаблону.

Тема 4. ПРЕЗЕНТАЦІЙНА ГРАФІКА. ПРИЗНАЧЕННЯ, ВИДИ І ЕТАПИ СТВОРЕННЯ ПРЕЗЕНТАЦІЇ

Призначення презентації

- Основною метою презентації є *надання інформації*. Презентація дозволяє зробити інформаційне повідомлення зрозуміліше простіше для сприйняття.
- Для досягнення цієї мети використовуються такі засоби, як мова, графіка, форма, колір, звук і ін. – тобто *мультимедіа*.

Засоби презентаційної графіки



Види презентацій

1. Презентація - супровід доповіді

Результати наукових досліджень підтвердили той факт, що наочність подачі матеріалу і використання кольору презентації значно збільшує об'єм інформації, що зрозуміла і запам'ятала, а мультимедійні ефекти підсилюють дію на слухачів, переконуючи погодитися з доповідачем.

Програми для презентації-супроводу доповіді:

- MSPowerPoint
- MSVisio

Основною метою презентації, що демонструється паралельно з доповіддю, є **представлення інформації**. Перевага програми MS Power Point полягає в тому, що вона дозволяє зробити повідомлення зрозуміліше і простіше для сприйняття.

Для досягнення цієї мети використовуються такі засоби, як мова, графіка, форма, колір, звук і ін. – одним словом, **мультимедіа.**

2. Презентація – відеокліп

- Корпоративна мультимедіа-презентація.
- Презентация-подарочное видання.
- Презентація проекту.
- Презентація для виставки.

Етапи роботи над презентацією

1. Визначення мети презентації.
2. Збір і аналіз матеріалу.
3. Розробка концепції і структури презентації, її візуалізація.
4. Наповнення слайдів.
5. Текстова і графічна оптимізація.
6. Дизайн і отрисовка.
7. Верстка.

Розглянемо ці етапи докладніше.

1. Визначення мети презентації

Мета презентації — спонукати слухачів до запланованої дії:

- *мета має бути виражена однією пропозицією*
- *мета має бути реалістичною*

2. Збір і аналіз матеріалу

Збір інформації – ***опис компанії, продукту, послуги або проекту, які мають відношення до теми презентації.***

3. Розробка концепції і структури

Розробка концепції і структури ***має на увазі забезпечення такої послідовності слайдів, щоб презентація ефективно реалізовувала поставлені завдання, щоб плавно підвести людину до ухвалення ключової ідеї презентації і до виконання поставленої мети.***

Спільна структура презентації може бути наступною:

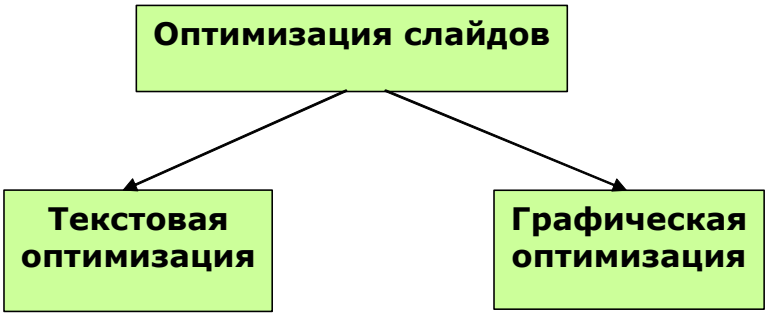
A. Оголошення основної теми
B. Роз'яснення поточної ситуації
C. Список можливих шляхів вирішення проблеми
D. Визначення достоїнств і недоліків кожної стратегії
E. Вибір однієї або декількох стратегій
F. Пропозиція конкретних кроків для здійснення вибраної стратегії.

4. Наповнення презентації (переклад інформації у формат PowerPoint)

Зібраний і проаналізований матеріал необхідно перевести у формат PowerPoint згідно розробленій структурі.

5. Текстова і графічна оптимізація слайдів

- Головне завдання оптимізації слайдів — зробити текстове і графічне наповнення слайдів максимально доступним, наочним, цікавим.
- Матеріал стає легким для сприйняття і запам'ятовування.



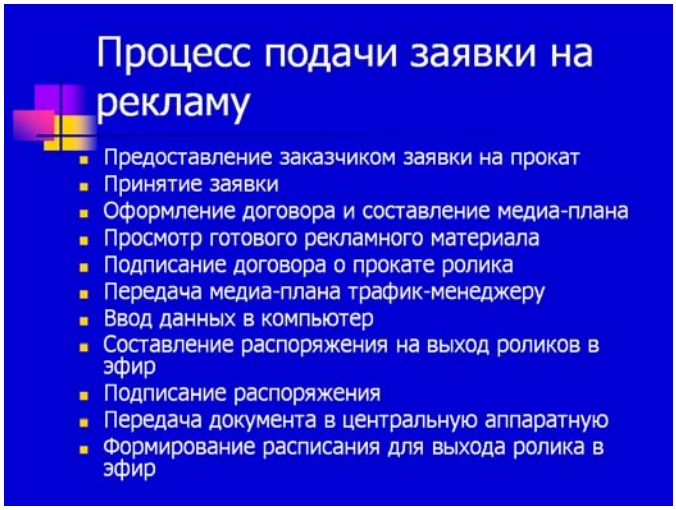
А. Текстова оптимізація слайдів

1. Переробка тексту в тези
2. Створення паралельних списків
3. Опрацьовування заголовків
4. Опрацьовування тез висновку

1. Переробка тексту в тези

Теза – положення, що коротко висловлює одну з основних думок вигадування, доповіді (С.І.Ожегов).

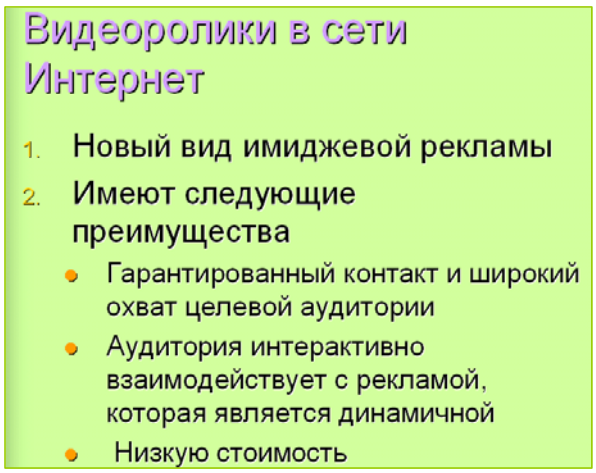
ПРИКЛАД: **ДО** **ПІСЛЯ**

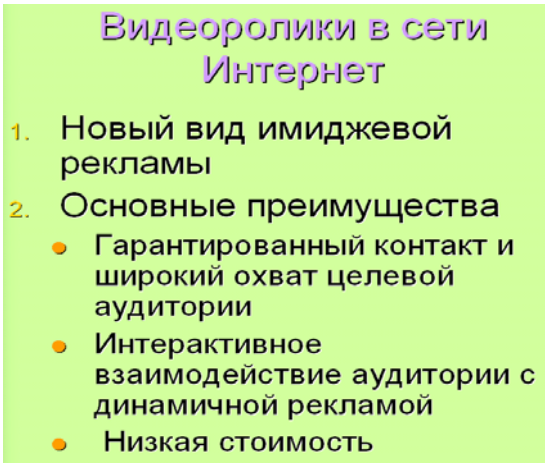


2. Створення паралельних списків

- ***Всі фрази в паралельному списку повинні починатися з однієї частини мови, стояти в одній формі*** (у одному роді, числі, відмінку, часі і заставі).
- ***Не можна***: строчки починати з дієслів в різному часі, чергувати застави**.**

ПРИКЛАД: **ДО** **ПІСЛЯ**





3. Опрацьовування заголовків

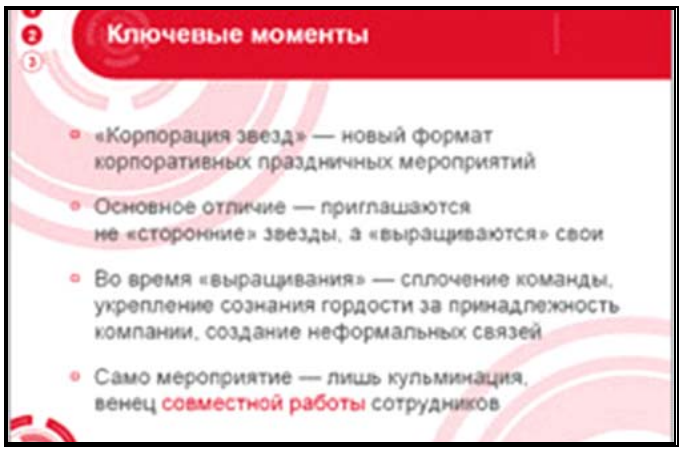
Заголовки повинні відображати основну ідею слайдів, бути такими, що запам'ятовуються, і ємкими. Тому тексти заголовків необхідно відкоректувати**.**

ПРИКЛАД: ДО ПІСЛЯ

4. Опрацьовування тез висновку

- ***Слайд, на якому представлені виводи, завершує презентацію і краще за інших запам'ятовується слухачами. Тому поважно грамотно і максимально зрозуміло для аудиторії сформулювати*** *тези, що представляють ключову ідею презентації.*
- ***Роботі над вмістом висновку необхідно приділити багато уваги, оскільки саме висновок ставить крапку у виступі і від нього залежить ефективність презентації.***

ПРИКЛАД:



В. Графічна оптимізація

- ***Графіка може піднести інформацію конкретніше і ясно в порівнянні з текстом і допомогти слухачам побачити взаємозв'язки між різними даними***
- ***Ідеї некількісного характеру можна виразити у візуальній формі. Так малюнки або «візуальні метафори» здатні зробити представлення інформації наочним і таким, що запам'ятовується.***

Оптимізація графічного наповнення слайдів включає:

1. ***Перетворення тексту в малюнки***
2. ***Перетворення тексту в схеми***
3. ***Перетворення тексту в таблиці***
4. ***Перетворення цифр в діаграми***

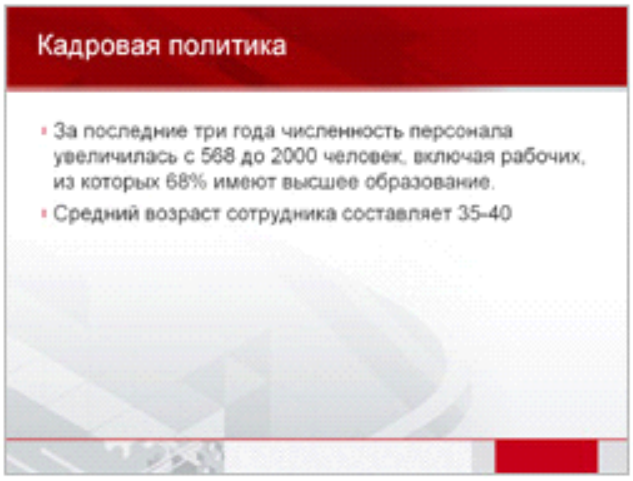
1. Перетворення тексту в малюнки

ПРИКЛАД: ДО ПІСЛЯ

2. Перетворення тексту в схеми

- ***Дані некількісного характеру можна представити у вигляді схем. Схеми личать для передачі таких ідей, як взаємодія, дія, перешкоди і взаємозв'язки, або таких понять, як структура, послідовність і процес.***
- ***Наприклад, складний технологічний процес можна описати словами або скористатися можливостями «візуальних концепцій», що складаються з абстрактних геометричних форм (стрілок, кругів, трикутників і т. д.).***

ПРИКЛАД: ДО ПІСЛЯ



3. Перетворення тексту в таблиці

Демонстрацію числових даних можна представити у вигляді таблиці.

ПРИКЛАД: **ДО** **ПІСЛЯ**

4. Перетворення цифр в діаграми

ПРИКЛАД: **ДО** **ПІСЛЯ**

6. Дизайн і отрисовка

Дизайн презентації — ***формування спільної візуальної лави презентації, що складається з дизайну загальнофонових часток і окремих елементів, що входять в презентацію.***

Дизайн і отрисовка **містять**:

1. ***Спільний дизайн***
2. ***Отрісовка від руки***
3. ***Схеми***

4. ***Малюнки***
5. ***Діаграми***
6. ***Карти***

1. Спільний дизайн

Спільний дизайн складається з трьох типів слайдів:

- ***Титульний***
- ***Основний***
- ***Роздільник***

Титульний слайд або обкладинка презентації — перший слайд, який побачить аудиторія на презентації.

Основний слайд. Фон основних слайдів несе вміст презентації, розроблений з врахуванням специфіки інформації, включеної в презентацію.

Слайд-роздільник істотно полегшить розуміння слухачами структури презентації.

2. Отрісовка від руки

Різні схеми і зображення можуть відмалювати «від руки», що додає презентації індивідуальність, оскільки такі зображення унікальні.

3. Схеми

Поняття графіка включає схеми, малюнки і діаграми:

4. Малюнки

Приклад отрисовки малюнків при створенні презентації

5. Діаграми і карти

Дані кількісного характеру в презентації можуть бути представлені у вигляді графіків і діаграм на тлі карт.

7. Верстка

Верстка *– компоновка в єдину презентацію текстового, графічного наповнення, дизайн, аудио- і відеороликів.*

В процесі фінальної верстки *з'являється* **єдина логіка** *оформлення всіх слайдів. Все раніше задумане «оживає», виходять слайди, що акуратно верстають, з цікавою і продуманою анімацією.*

Верстка включає:

- Створення зразків слайдів
- Робота з шрифтами (обмеження кількості)
- Підгонка заголовків і тексту під слайди
- Розміщення схем, таблиць, малюнків
- Розміщення і форматування тексту (відступи, інтервали, перенесення)
- Форматування списків (маркери, відступи)
- Додавання гіперпосилань
- Вибір способів появи слайдів і елементів (анімація)

Розмір шрифту

Рекомендується для демонстрації презентації за допомогою проектора використовувати:

- ***для заголовків шрифт розміром*** *не менше 26 пунктів*
- ***для звичайного тексту —*** *не менше 20* ***залежно від специфіки презентації***

При виборі розміру шрифтів необхідно керуватися не лише зручністю сприйняття, але і мати на увазі майдан екрану, на який проектується презентація, і відстань до нього від останніх лав.

Не слід користуватися більш ніж трьома шрифтами в презентації:

- ***одним для заголовків***
- ***іншим для тексту в маркіруваному списку***
- ***третім для спеціального виділення тексту на окремих слайдах.***

Переважно, проте, обмежитися двома шрифтами.

Поширені помилки:

- *оформлення списків на слайдах в лапках («.»);*
- *тире «—» на початку фрази або крапка з комою в кінці (;).*

Поширені помилки:

- ***знаки, які використовуються для оформлення документів в Word, непридатні для презентацій. Вони відволікають увагу слухачів і створюють*** відчуття закінченості фраз***, в той час, як текст слайдів повинен викликати інтерес до розповіді оратора.;***

- ***під час верстки поважно враховувати психологію сприйняття, — те, хто ваші слухачі і як вони сприймають те, що бачать.***

Тема 5. ОСНОВИ ПОБУДОВИ WEB-САЙТІВ

Створення сторінок в Інтернет – один з напрямів, де необхідні знання по комп'ютерній графіці. В цій лекції коротко описана мова HTML розмітки гіпертексту, за допомогою якої створюється велика частка сторінок в усесвітній павутині. За допомогою HTML можна створювати не лише сторінки для Інтернету. Наприклад, в HTML можна створити інтерактивну презентацію будь-якій складності або оформити і підготувати документ (текст, графічний колаж) для друку. Плюс до всього цього, можна освоїти спеціальні мови програмування (найбільш популярні з них - Javascript і PHP). З їх допомогою створюється так званий динамічний HTML (Dynamic HTML). Так створюються гостьові книги, чати, Інтернет-магазини і інші інтерактивні сторінки (що дозволяють користувачеві взаємодіяти з собою). За допомогою динамічного HTML можна також створювати ефективніші сторінки (наприклад, замість 50 схожих сторіночок написати одну динамічну).

Основні поняття Web

Неформатований текст (звичайний текст, ASCII текст) – створюється в блокноті (Notepad) або другом простому текстовому редакторові. Неформатований текст – тобто без форматування, в нім не можуть використовуватися різні шрифти, різний розмір тексту, вирівнювання і так далі

Код HTML – теж може бути створений в простому текстовому редакторові, але відрізняється від звичайного тексту тим, що містить спеціальні команди мови розмітки гіпертексту (HTML) – теги.

Web-сторінк а (або HTML-сторінк а, HTML-документ, сторінка, сторіночка) – один файл, що містить код HTML, і що має розширення htm, html, php, phtml (є та інші, рідше використовувані розширення).

Web-дизайнер – той, хто створює web-сторінки . Web-дизайнер може:

- Створювати HTML-код в текстовому редакторові (або в редакторові коди). Далі розглядується саме такий спосіб.
- Створювати web-сторінку візуально. Для цього використовується спеціальний візуальний (WYSIWYG, What You See Is What You Get – що бачиш, то і отримуєш) редактор. Другий спосіб простіше для початківця, проте, частенько візуально створені сторінки відрізняються кривим кодом і великою кількістю помилок.

Браузер (browser, броузер) – програма користувача для проглядання Web-сторінк . Таким чином, Ви створюєте сторінку в одній програмі – редакторові, проглядаєте її в іншій – браузері. Можна сказати, що браузер – програма для візуального відображення коди HTML. ***Приклади браузеров:*** Microsoft Internet Explorer, Mozilla, Netscape, Firefox, Opera.

Інтернет (Internet, World Wide Web, усесвітня павутина, мережа) – велика кількість комп'ютерів по всьому світу, сполучених між собою.

URL (Universal Resource Locator) – адреса web-сторінки або файлу в усесвітній мережі Інтернет. Кожна сторінка або файл в мережі має свою унікальну адресу URL. Ця адреса записується в спеціальний адресний рядок браузера. Приклади URL:

http://www.yandex.ru
http://seegix.net
http://1188.ru/index.php

У більшості браузеров не обов'язково навіть писати http:// і www:

yandex.ru
seegix.net
1188.ru/index.php

Користувач (або user) – той, хто проглядатиме Web-сторінки (у контексті Web-дизайна).

Web-сайт (website, site, сайт) – сукупність web-сторінк , об'єднаних деякою спільною ідеєю і зв'язаних між собою гіперпосиланнями.

Гіперпосилання – візуальний елемент сторінки, що пов'язує її з іншими сторінками або файлами. При натисненні на гіперпосилання (зазвичай є текстом або картинкою), Ви переходите до іншої сторінки, виконуєте певну команду або викачуєте файл з мережі.

Введення в HTML

Далі вводиться поняття "тег" і розглядується приклад простої web-сторінки, детально розглядуються теги і наводяться приклади їх параметрів. Потім розглядується структура web-сторінки, теги параграфа, заголовка, відступу і способи організації коментаря.

HTML (Hyper Text Markup Language) означає мову розмітки гіпертексту. Ця мова була розроблена Тімом Бернерсом-Лі в рамках створення проекту розподіленої гіпертекстової системи, яку він назвав **World Wide Web** (WWW) або Усесвітня павутина. HTML призначений для написання гіпертекстових документів, що публікуються в World Wide Web. Документ на мові HTML може включати наступні компоненти:

стилізований і форматований текст
команди включення графічних і звукових файлів
гіперзв'язки з різними ресурсами Internet.
скрипти на мові JavaScript і VBScript.
різні об'єкти, наприклад Flash-анимацию

Документи HTML є звичайними текстовими файлами, що містять спеціальні **теги** (або ***елементи, що управляють***) розмітки. Теги розмітки указують браузеру, як треба вивести сторінку.

Файли HTML зазвичай мають розширення ***htm*** або ***html***. Їх можна створювати за допомогою будь-якого текстового редактора.

Мова HTML є підмножиною потужної мови SGML (Standard Generalized Markup Language), яка широко використовується у видавничій діяльності. Основний виграш від використання цих мов полягає в переносимості тексту між різними видавничими системами. Ця ж особливість зберігається і в HTML. Так, читаючи документ, користувачі можуть встановлювати способи виділення тексту, гарнітуру і розмір шрифтів за своїм смаком; вони можуть відмінити проглядання малюнків.

У документі HTML можна виділити ***два основні блоки***: **головна частина і тіло документа.** Вміст головної частини не виводиться на екран користувача, за винятком заголовка. У ній, як правило, указують ключові слова, авторів і іншу службову інформацію, а також підключають зовнішні таблиці стилів і скрипти. У тілі документа розміщують ту інформацію, яка буде виведена користувачеві.

У різних операційних системах є різні редактори, яких можна використовувати для створення документів HTML. Якщо ви використовуєте Microsoft Windows, то запустіть редактор Notepad.

Файли HTML можна створювати і в редакторові Microsoft Word, в якому є можливість зберегти документ як Web-сторінк у (у меню "Файл"), проте використовувати цю можливість не рекомендується. По-перше, тому що HTML-код, MS Word, що генерується, не оптимальний і містить безліч непотрібних елементів розмітки, і, по-друге, автоматична генерація коди не сприятиме вивченню і правильному розумінню HTML.

Є також велика кількість спеціалізованих редакторів для створення файлів HTML, таких як FrontPage, Macromedia Dreamweaver або Adobe Web Bundle, які володіють можливістю WYSIWYG (What You See Is What You Get - що бачиш, то і отримаєш). З їх допомогою можна легко створювати документи HTML, за допомогою кнопок і елементів меню, а не писати самому теги розмітки. Проте, як вже наголошувалося вище, тим, хто хоче стати технічно грамотним розробником Web, настійно рекомендується використовувати простою текстовий редактор для початкового вивчення HTML.

Надрукуйте наступний текст:

```
<html>
```

```
<head>
<title>Это заголовок сторінки </title>
</head>
<body>
<h1>Доброго дня!</h1>
<p>Це моя перша сторінка HTML. <b>Цей текст виводиться жирним шрифтом.</b></p>
</body>
</html>
```

Приклад виконання даного HTML-коду

Доброго дня!

Це моя перша сторінка HTML. **Цей текст виводиться жирним шрифтом.**

Збережете файл як "page1.htm".

При збереженні файлу HTML можна використовувати розширення .htm або .html. Розширення .htm було прийнято для старих версій операційних систем, які допускали трьохбуквене розширення для файлів. В даний час практично всі операційні системи не мають подібного обмеження і можна використовувати розширення .html.

Тепер Перегляньте, як браузер відображуватиме вашу першу сторінку. Запустите браузер Інтернет. Виберіть "Open" або "Open Page" ("Відкрити" або "Відкрити сторінку") в меню File (Файл) браузера. З'явиться діалогове вікно. Виберіть "Browse" або "Choose File" ("Перегляд" або "Вибрати файл") і знайдіть тільки що створений файл HTML - "page1.htm" - виберіть його і клацніть на кнопці "Open" ("Відкрити"). У діалоговому вікні повинна з'явитися адреса, наприклад "C:\MyDocuments\page1.htm". Клацніть на кнопці ОК, і браузер виведе на екран вашу сторінку.

Розбір прикладу

Ваш перший HTML-документ починається з тега <html>, який повідомляє браузеру про початок документа HTML і закінчується тегом </html>, який інформує браузер про досягнення кінця документа HTML.

Текст між тегами <head> і </head> є інформацією заголовка документа. Ця інформація не виводиться у вікні браузера.

Текст "Це заголовок сторінки" між тегами <title> і </title> є заголовком документа. Цей заголовок виводиться в рядку заголовка вікна браузера.

Текст між тегами <body> і </body> є текстом, який буде виведений у вікні браузера. Текст "Доброго дня!" між тегами <h1> і </h1> відображуватиме стилем заголовка, зазвичай жирним шрифтом більшого розміру.

Тег означає, що починається новий параграф, тег означає кінець параграфа.

Текст "Цей текст виводиться жирним шрифтом." між тегами і буде виведений жирним шрифтом.

Коротко про теги

Теги HTML використовуються для виділення елементів HTML. Зазвичай теги HTML використовуються парами і поміщені між двома символами кутових дужок < (початковий тег)> і </(кінцевий тег)>. **Все що обмежене кутовими дужками "<" і ">" називається тегом.** Теги є командами мови HTML і при прогляданні сторінки в браузері не відображуються. Текст між початковим і кінцевим тегами є **вмістом елементу**. Деякі теги не мають кінцевого, наприклад, тег примусового перенесення рядка
. Для таких тегов рекомендується використовувати наступне написання
.

Регістр символів для відображення тегов не важливий, наприклад, і <P> означає одне і те ж. Проте в цьому курсі використовується нижній регістр для написання тегов. Це пов'язано з тим, що консорціум WWW (W3C), який займається стандартизацією специфікації HTML, рекомендує використовувати теги в нижньому регістрі, оскільки в наступному поколінні стандартів буде саме такий вимога.

Коротко про елементи HTML

Елементом називається пара тегов з однаковим ім'ям, в останньому з яких коштує зворотний слеш "/". Єдиним ***виключенням є елемент*** <!DOCTYPE>, який складається тільки з відкриваючого тега.

Розглянемо той же приклад документа HTML:

```
<html>
<head>
<title>Це заголовок сторінки </title>
</head>
<body>
<h1> Доброго дня!</h1>
<p>Це моя перша сторінка HTML.
 <b>Цей текст виводиться жирним шрифтом.</b></p>
</body>
</html>
```

Приклад виконання даного HTML-коду

Доброго дня!

Це моя перша сторінка HTML. **Цей текст виводиться жирним шрифтом.**

Елементом HTML є:

```
<h1>Доброго дня!</h1>
```

Цей елемент починається з тега <h1>, має вміст "Доброго дня!" і закінчується тегом </h1>.

Також елементом HTML є:

```
<p>Це моя перша сторінка HTML.
<b>Цей текст виводиться жирним шрифтом.</b></p>
```

Цей елемент, починається з початкового тега , закінчується кінцевим тегом і означає, що вміст елементу "Це моя перша сторінка HTML. <b>Этот

текст виводиться жирним шрифтом." є окремим параграфом. При цьому усередині цього елементу знаходиться інший елемент:

```
<b>Этот текст виводиться жирним шрифтом.</b>
```

Цей елемент HTML починається з початкового тега: Вмістом елементу HTML є: Цей текст виводиться жирним шрифтом. Цей елемент HTML закінчується кінцевим тегом . Призначення тега полягає у визначенні елементу HTML, який повинен виводитися жирним шрифтом.

Всі описані елементи HTML містяться в елементі:

```
<body>
<h1>Доброго дня!</h1>
<p>Це моя перша сторінка HTML.
<b>Цей текст виводиться жирним шрифтом.</b></p>
</body>
```

Цей елемент HTML починається з початкового тега <body>, і закінчується кінцевим тегом </body>. Призначення тега <body> полягає у визначенні елементу HTML, який містить основну частку (або тіло) документа HTML.

Атрибути тегов

Теги можуть мати атрибути, які надають додаткову інформацію про елементи HTML. Атрибути завжди використовуються у вигляді пари "ім'я/значення". Спільний формат завдання атрибутів має вигляд:

```
<им'я_тега им'я_атрибута="значення">
```

Наприклад, тег:

```
<body bgcolor="red">
```

означає, що колір фону сторінки має бути червоним.

А тег:

```
<p align="center">
```

означає, що параграф необхідно вирівняти по центру сторінки відображення браузера.

Атрибути завжди поміщаються в початковому теге елементу HTML. Значення атрибутів завжди корисно брати в лапок. Найширше використовуються подвійні лапки, але одиночні лапки також допустимі.

У деяких рідких ситуаціях, коли, наприклад, значення атрибуту само містить лапки, необхідно використовувати одиночні лапки:

```
<html>
<body>
<abbr title='проект "інтернет-університет
  Інформаційних Технологій" - INTUIT.ru'>ИНТУИТ</abbr>
</body>
</html>
```

Приклад виконання даного HTML-коду

ІНТУЇТ

Основні теги HTML

Параграфи

Перш ніж вивчати теги форматування HTML, глянемо як введений текст відображуватиметься, якщо не будуть застосовані ніякі теги, окрім тегов <html> і <body>. Наступний приклад демонструє такий документ HTML

```
<html>
<body>
Цей текст буде показаний у вікні браузера.
</body>
</html>
```

Приклад виконання даного HTML-коду

Цей текст буде показаний у вікні браузера.

Цей простий приклад документа HTML, який містить мінімальну кількість тегов HTML і демонструє, як текст усередині елементу body відображується в браузері.

Якщо ввести великий об'єм тексту в такий спосіб, то читати його буде дуже незручне. Логічніше розбити його на параграфи, як в книзі, які підвищують читабельність тексту, і крім того виділяють смислові блоки.

Наступний приклад показує, як відображуються параграфи

```
<html>
<body>
<p>Это параграф 1.</p>
<p>Это параграф 2.</p>
<p>Это параграф 3.</p>
</body>
</html>
```

Приклад виконання даного HTML-коду

Це параграф 1.

Це параграф 2.

Це параграф 3.

Цей приклад демонструє, як в браузері виводиться текст усередині елементів параграфа. Можна бачити, що за умовчанням текст кожного параграфа виводиться у вигляді окремого блоку. Кожен з таких блоків відділяється від попередніх і подальших блоків сторінки порожнім рядком. Проте відображення параграфа браузером може бути легко змінене за допомогою таблиці стилів.

У різних браузерах на різних моніторах з різним дозволом сторінка відображуватиметься по-різному, тому не варто форматувати за допомогою додавання порожніх рядків і пропусків. ***Будь-яке число пропусків замінюється одним***.

Використання порожніх параграфів <p> для вставки порожніх рядків є поганим стилем, замість цього використовуйте тег
.

Заголовки

Заголовки визначаються за допомогою тегов від <h1> до <h6>. <h1> визначає заголовок найбільшого розміру, а <h6> визначає заголовок найменшого розміру.

```
<h1>Это заголовок першого уровня</h1>
<h2>Это заголовок другого уровня</h2>
<h3>Это заголовок третьего уровня</h3>
<h4>Это заголовок четвертого уровня</h4>
<h5>Это заголовок п'ятого уровня</h5>
<h6>Это заголовок шостого уровня</h6>
```

Заголовки автоматично відділяються додатковими проміжками від решти елементів документа.

Перенесення рядків

Для перенесення усередині параграфа використовується тег
, який виконує примусове перенесення рядка.

```
<html>
<body>
<p>Это <br>пара<br>граф з перенесеннями строк</p>
</body>
</html>
```

Приклад виконання даного HTML-коду

Це
пара
граф з перенесеннями рядків

Тег
 не має закриваючого тега. Тому для сумісності з майбутніми версіями стандарту рекомендується наступне написання тега

Горизонтальна лінійка

Розділяти різні елементи можна за допомогою горизонтальної лінійки, для цього використовуйте тег <hr>:

```
<html>
<body>
<p>Этот параграф відображуватиметься зверху горизонтальної смуги.</p>
<hr>
<p>Этот параграф відображуватиметься знизу горизонтальної смуги.</p>
</body>
</html>
```

Приклад виконання даного HTML-коду

Це параграф відображуватиметься зверху горизонтальної смуги.

Це параграф відображуватиметься знизу горизонтальної смуги.

Тег <hr> не має закриваючого тега. Тому для сумісності з майбутніми версіями стандарту рекомендується наступне написання тега <hr/>. Для цього тега визначений лава атрибутів, але вони є застарілими. І хоча їх вживання можливе, але консорціум W3C їх використовувати не рекомендує. Замість них слід використовувати таблиці стилів.

Коментарі в HTML

Тег коментаря використовується для вставки коментарів в початковий код HTML. Коментарі будуть проігноровані браузером. Коментарі можна використовувати для пояснення коди, що може допомогти при редагуванні початкової коди в майбутньому.

```
<!-- Це коментар -->
```

Ось приклад:

```
<html>
<body>
Цей текст буде показаний у вікні браузера.
<!-- Цей текст не буде показаний, це коментар. -->
</body>
</html>
```

Приклад виконання даного HTML-коду

Цей текст буде показаний у вікні браузера.

Додаткові приклади

Кращим способом вивчення HTML є робота з прикладами. Розглянемо декілька прикладів, які ілюструють деякі елементи форматування документів.

Додаткові параграфи

Цей приклад демонструє деякі особливості поведінки за умовчанням елементів параграфа.

```
<html>
<body>

<p>
Цей параграф
містить багато рядків
у початковому коді
але браузер
це ігнорує.
</p>

<p>
Цей параграф
містить     багато пропусків
у початковому    коді
але     браузер
це ігнорує.
</p>

</body>
</html>
```

Приклад виконання даного HTML-коду

Цей параграф містить багато рядків в початковому коді, але браузер це ігнорує.
Цей параграф містить багато пропусків в початковому коді, але браузер це ігнорує.

Перенесення рядків

Цей приклад демонструє використання перенесення рядків в документі HTML.

```
<html>
<body>

<p>
Щоб виконати перенос<br>строк<br>в
<br>параграфе,<br>используйте тег br.</p>
</body>
</html>
```

Приклад виконання даного HTML-коду

```
Це
пара
граф з перенесеннями рядків
```

Цей приклад демонструє деякі проблеми з форматуванням HTML. Спробуємо відформатувати вірші:

```
<html>
<body>

<p>
  У лісі народилася ялиночка.
  У лісі вона зростала.
  Взимку і літом струнка
  Зелена була.
</p>

<p>Обратите увага, що браузер просто проігнорував використане
форматування!</p>

</body>
</html>
```

Приклад виконання даного HTML-коду

У лісі народилася ялиночка. У лісі вона зростала. Взимку і літом струнка, Зелена була.

Звернете увагу, що браузер просто проігнорував використане форматування!

Заголовки

Цей приклад демонструє теги, які виводять заголовки в документі HTML.

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

```
<h1>Это заголовок рівня 1</h1>
<h2>Это заголовок рівня 2</h2>
<h3>Это заголовок рівня 3</h3>
<h4>Это заголовок рівня 4</h4>
<h5>Это заголовок рівня 5</h5>
<h6>Это заголовок рівня 6</h6>

<h1 align="center">Это заголовок 1, він вирівняний по центру сторінки.</h1>

<p>Используйте теги заголовків тільки для заголовків.
Не використовуйте їх просто для того, щоб виділити щось жирним шрифтом.
Використовуйте для цього інші теги.</p>

</body>
</html>
```

Приклад виконання даного HTML-коду

Це заголовок рівня 1

Це заголовок рівня 2

Це заголовок рівня 3

Це заголовок рівня 4

Це заголовок рівня 5

Це заголовок рівня 6

Це заголовок 1, він вирівняний по центру сторінки.

Використовуйте теги заголовків тільки для заголовків. Не використовуйте їх просто для того, щоб виділити щось жирним шрифтом. Використовуйте для цього інші теги.

Горизонтальна лінійка

Цей приклад демонструє, як використовувати горизонтальну лінійку.

```
<html>
<body>
<p>Тег <hr> визначає горизонтальну лінійку :</p>
<hr>
<p>Это параграф</p>
<hr>
<p>Это параграф</p>
</body>
</html>
```

Приклад виконання даного HTML-коду

Тег

визначає горизонтальну лінійку :

Це параграф

Це параграф

Коментар

Цей приклад демонструє, як використовувати коментар в початковому коді HTML.

```
<html>
<body>

<!--Цей коментар виводитися не буде-->
<p>Это звичайний параграф.</p>

</body>
</html>
```

Приклад виконання даного HTML-коду

Це звичайний параграф.

Фоновий колір

Цей приклад демонструє використання кольорового фону на сторінці HTML. При виборі фону завжди перевіряйте, щоб текст був добре читаний!

```
<html>
<body bgcolor="yellow">
<h2>Смотри: Кольоровий фон!</h2>
</body>
</html>
```

Приклад виконання даного HTML-коду

Дивися: Кольоровий фон!

Таблиця основних тегов HTML

Тег	Опис
<html>	Визначає документ HTML
<body>	Визначає основну частку або тіло документа
<h1> -- <h6>	Визначає заголовки з 1 по 6
<p>	Визначає параграф
 	Вставляє одиничне перенесення рядка
<hr>	Визначає горизонтальну лінійку
<!-->	Визначає коментар

Тема 6. ФОРМАТУВАННЯ HTML-ДОКУМЕНТУ

Розглянемо способи і теги форматування текстової інформації, а також способи виводу на екран спеціальних символів.

Щоб відображувати текст, що не відформатував, досить просто ввести його між тегами зачала і кінця документа <body></body>. При обробці такої сторінки браузер знайде і виведе весь цей текст. Якщо необхідно, щоб до тексту було застосовано яке-небудь форматування, наприклад, виділення напівжирним або курсивом, необхідно використовувати відповідні теги форматування. При цьому текст, що форматується, поміщається між тегами. Розглянемо все вище викладене на прикладах.

Форматування тексту

```
<html>
<body>
<p>
Якщо необхідно щоб до тексту було застосовано яке-небудь
форматування, наприклад, виділення <b>полужирным</b>
або <i>курсивом</i>, необхідно використовувати
відповідні теги форматування.
При цьому текст, що форматується, поміщається між тегами.
</p>
</body>
</html>
```

Приклад виконання даного HTML-коду

Якщо необхідно щоб до тексту було застосовано яке-небудь форматування, наприклад, виділення **напівжирним** або *курсивом*, необхідно використовувати відповідні теги форматування. При цьому текст, що форматується, поміщається між тегами.

Також для виділення тексту використовуються теги і , дані теги є контейнерами і вимагають тега, що закривається. Тег повідомляє браузеру, що на ув'язненому в нім тексті необхідно зробити сильний наголос. Зазвичай візуальні браузеры відображують вміст даного тега напівжирним шрифтом, але це може бути легко змінено за допомогою таблиці стилів. Оскільки даний тег є структурним, він несе смислове навантаження, на відміну від тега <b>, який лише форматує текст напівжирним, то його використання предпочтительней. Тег теж акцентує увагу на укладеному усередині тексті, але він вважається за менш сильний наголос. Відображується він, як правило, курсивом. По тих же причинах використання предпочтительней чим вживання <i>. Порівняєте як відформатує наступний текст.

```
<html>
<body>
<p><strong>Данный параграф відформатував тегом strong</strong></p>
<p><b>А цей тегом b, зовні вони не відрізняються.</b></p>
<p><em>Данный параграф відформатував тегом em</em></p>
<p><i>А цей тегом i, зовні вони не відрізняються.</i></p>
</body>
</html>
```

Приклад виконання даного HTML-коду

**Даний параграф відформатував тегом **

А цей тегом <b>, зовні вони не відрізняються.

*Даний параграф відформатував тегом *

А цей тегом <i>, зовні вони не відрізняються.

Ще однією парою тегов форматування є теги <big> і <small>. Перший виводить текст збільшеним в порівнянні із стандартним, а другий зменшеним. Замість тега <big> рекомендується використовувати теги або теги заголовків, оскільки вони несуть і структурне навантаження. Тег <small> є тегом по сенсу протилежним і , він деакцентирует увага на тексті. З приводу вживання цього тега слід зробити одне зауваження: на різних комп'ютерах встановлений різний дозвіл екрану і в браузері може бути встановлений різний розмір основного шрифту і сильне зменшення тексту може зробити текст не читаним. Наступний приклад демонструє вид тексту, що відформатував за допомогою цих тегов.

```
<html>
<body>
<p><big>Данный параграф відформатував тегом big </big></p>
<p><small>Данный параграф відформатував тегом small </small></p>
<p>А в даному параграфі теги не применяются</p>
</body>
</html>
```

Приклад виконання даного HTML-коду

Даний параграф відформатував тегом <big>

Даний параграф відформатував тегом <small>

А в даному параграфі теги не застосовуються

Іншою парою корисних тегов є <sup> - верхній індекс і <sub> - нижній індекс, які можуть бути корисними при написанні математичних і хімічних формул. Порівняєте формули, набрані різним способом:

```
<html>
<body>
<p>Формула води H2O. У даному параграфі формула набрана
без використання тега sub</p>
<p>Формула води H<sub>2</sub>O.
У даному параграфі формула набрана з використанням тега sub
Формула виглядає звичніше.</p>
<p>2^4=16.
У даному параграфі формула набрана без використання тега sup</p>
<p>2<sup>4</sup>=16. У даному параграфі формула набрана
з використанням тега sup.
Формула виглядає звичніше.</p>
</body>
</html>
```

Приклад виконання даного HTML-коду

Формула води H2O. У даному параграфі формула набрана без використання тега <sub>

Формула води H2O. У даному параграфі формула набрана з використанням тега <sub> Формула виглядає звичніше.

2^4=16. У даному параграфі формула набрана без використання тега <sup>

2^4=16. У даному параграфі формула набрана з використанням тега <sup> Формула виглядає звичніше.

В деяких випадках, наприклад для виведення коди програм, корисним буде використання тега <pre>, який повідомляє браузеру, що текст, що знаходиться усередині, має бути виведений як є. При цьому будуть збережені всі пропуски, перенесення рядків і інші символи, які зазвичай при виводі браузером не відображуються.

Цей приклад показує, як можна управляти перенесеннями рядків і пропусками за допомогою тега <pre>.

```
<html>
<body>

<pre>
Це
заздалегідь форматований текст.
Він зберігає     як пропуски
так і перенесення рядків.
</pre>
</body>
</html>
```

Приклад виконання даного HTML-коду

```
Це
заздалегідь форматований текст.
Він зберігає     як пропуски
так і перенесення рядків.
```

Тег <address> маркірує контактну інформацію для всього документа або його частки. Він може включати імена людей що здійснюють підтримку документа, заслання на сторінки, адреси електронної пошти, телефони і інше. Використання даного тега для маркіровки поштових адрес організацій є не зовсім коректним. Його слід використовувати тільки для позначки адрес для зв'язку з приводу документа. Також допустиме використання тега <address> для виділення контактній інформації з приводу частки документа, зазвичай форми.

```
<html>
<body>
<form method=post action="/cgi-bin/order.cgi">

  <fieldset>
    <legend accesskey=c>Информация про кредитну карте<br></legend>
    <p>
```

```
      <label accesskey=v>
        <input type=radio name=card value=visa> Visa
      </label>
      <label accesskey=m>
        <input type=radio name=card value=mc> Mastercard
      </label>
      <br>
      <label accesskey=n>
        Номер: <input type=text name=number>
      </label>
      <label accesskey=e>
        Термін дії: <input type=text name=expiry>
      </label>
    </p>
  </fieldset>

  <p>
    <input type=submit value="Надіслати замовлення" accesskey=s>
  </p>

  <address>
    Якщо у вас є питання з приводу замовлення, зв'яжіться з нами
    за адресою <a href="mailto:orders@intuit.ru">orders@intuit.ru</a>,
    або по телефону в офісі (+7 495) 253-9312.
  </address>

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

Приклад виконання даного HTML-коду

Інформація про кредитну карте

Visa Mastercard

Номер: Термін дії:

Отправить заказ

Якщо у вас є питання з приводу замовлення, зв'яжіться з нами за адресою, Або телефоном в офісі (+7 499) 253-9312.

Цей приклад показує, як працювати з скороченнями або акронимами.

```
<html>
<body>

<abbr title="Содружество Незалежних Государств">снг</abbr>
<br>
<acronym title="World Wide Web">WWW</acronym>
```

```
<p>При наведенні покажчика миші на акроним або скорочення
показується атрибут title.</p>

<p>Это працює по-різному в різних браузерах.</p>

</body>
</html>
```

Приклад виконання даного HTML-коду

СНД
WWW

При наведенні покажчика миші на акроним або скорочення показується атрибут title.

Це працює по-різному в різних браузерах.

Цей приклад показує, як змінювати напрям виведення тексту.

```
<html>
<body>

<p>
Якщо використовуваний браузер підтримує двонаправлене
уявлення (bdo), то наступний рядок буде записаний
справа наліво (rtl):
</p>

<bdo dir="rtl">
Тут якийсь арабський текст
</bdo>

</body>
</html>
```

Приклад виконання даного HTML-коду

Якщо використовуваний браузер підтримує двонаправлене уявлення (bdo), то наступний рядок буде записаний справа наліво (rtl):

Тут якийсь арабський текст

Цей приклад показує, як використовувати довгі і короткі цитати.

```
<html>
<body>

Тут представлена довга цитата:
<blockquote>
Це довга цитата. Це довга цитата. Це довга цитата.
Це довга цитата. Це довга цитата.
</blockquote>

Тут представлена коротка цитата:
```

```
<q>
Це коротка цитата
</q>

<p>
Для елементу blockquote браузер вставляє додаткові
перенесення рядка, порожні рядки і поля, але елемент q
не змальовується якимсь спеціальним чином.
</p>

</body>
</html>
```

Приклад виконання даного HTML-коду

Тут представлена довга цитата:
Це довга цитата. Це довга цитата. Це довга цитата. Це довга цитата. Це довга цитата.
Тут представлена коротка цитата: Це коротка цитата
Для елементу blockquote браузер вставляє додаткові перенесення рядка, порожні рядки і поля, але елемент q не змальовується якимсь спеціальним чином.

Цей приклад показує, як помітити текст, який видалений або вставлений в документ.

```
<html>
<body>
<p>
дюжина означає
<del>двадцать</del>
<ins>двенадцать</ins>
часток
</p>
<p>
Більшість браузеров закреслюватимуть видалений текст
і підкреслювати вставлений текст.
</p>
<p>
Деякі старі браузеры виводитимуть видалений
або вставлений текст як звичайний текст.
</p>
</body>
</html>
```

Приклад виконання даного HTML-коду

дюжина означає ~~двадцать~~ дванадцять часток
Більшість браузеров закреслюватимуть видалений текст і підкреслюватимуть вставлений текст.
Деякі старі браузеры виводитимуть видалений або вставлений текст як звичайний текст.

Початковий код HTML-документа

Корисно вивчати код Web-сторінк , зроблених іншими. Таку можливість надають всі популярні браузеры. Для цього в меню кнопці View ("Вигляд") браузера

слід вибрати Source ("Початковий код") або Page Source ("Код сторінки"). Відкриється вікно, в якому буде показаний фактичний код HTML сторінки.

Таблиця тегов управління формою відображення	
Тег	**Опис**
<b>	Задає жирний текст
<big>	Відносне збільшення тексту
	Виділяє текст (зазвичай курсив)
<i>	Задає курсив
<small>	Відносне зменшення тексту
	Акцентує текст (зазвичай жирний)
<sub>	Визначає нижній індекс
<sup>	Визначає верхній індекс
<ins>	Вставлений текст
<del>	Видалений текст
<s>	Не рекомендується. Використовуйте замість цього <del>
<strike>	Не рекомендується. Використовуйте замість цього <del>
<u>	Не рекомендується. Використовуйте замість цього стилі (style)
Таблиця тегов управління типом інформації	
Тег	**Опис**
<code>	Визначає текст коди програми

<kbd>	Визначає текст клавіатури
<samp>	Визначає зразок коди програми
<tt>	Визначає текст телетайпу
<var>	Визначає змінну
<pre>	Визначає текст, що заздалегідь відформатував
<listing>	Не рекомендується. Використовуйте замість цього <pre>
<plaintext>	Не рекомендується. Використовуйте замість цього <pre>
<xmp>	Не рекомендується. Використовуйте замість цього <pre>

Таблица тегов цитирования, сносок и определения

Тег	Опис
<abbr>	Визначає скорочення
<acronym>	Визначає акроним
<address>	Визначає елемент address (адреса)
<bdo>	Визначає напрям виведення тексту
<blockquote>	Визначає довгу цитату
<q>	Визначає коротку цитату
<cite>	Визначає виноску на інший матеріал
<dfn>	Використовується для визначення терміну

Тема 7. ГІПЕРПОСИЛАННЯ. ЗАСТОСУВАННЯ ГРАФІКИ

При зберіганні вся інформація може бути представлена або у вигляді одного цілісного блоку (як один великий текстовий документ) або у вигляді декількох незалежних блоків, зв'язаних між собою тільки спеціально

оформленими заславнями. Зберігання і представлення інформації у вигляді незалежних часток (сторінок), зв'язаних між собою заславнями, називається ***гіпертекстом або гіпертекстовим документом***. Заслання між частками називаються ***гіпертекстовими заславнями*** або просто заславнями.

Заслання можуть указувати, як на окремі частки документа, так і на різні документи. Заслання в межах одного документа називаються ***внутрішніми***, а на інші документи - ***зовнішніми***.

У мові HTML передбачений тег <a>, за допомогою якого і створюються гіперпосилання. За допомогою атрибуту name= тега <a> указується місце в документі, на яке потім можна буде перейти. Інший спосіб вказівки точки переходу в документі - за допомогою спільного атрибуту id=. Сам перехід в межах одного документа, або в інший документ, указується за допомогою атрибуту href=. Ім'я точки переходу, вказаної за допомогою атрибуту name= або id=, в атрибуті href= повинно обов'язково зачинатися символом «#». Наприклад, якщо name="labell", то href="#labell".

Зовнішні заслання можуть бути відносними якщо указують на документ в межах одного сайту (або комп'ютера) і ***абсолютними*** - коли указується протокол обміну і повна адреса документа в мережі, наприклад href="http://www.ksame.kharkov.ua/portal.php". Якщо буде вказана тільки тека, без імені конкретного файлу, то передбачається, що це заслання на сайт, і в цій теці буде відкритий файл ***index.html*** або ***index.php***.

Атрибут href= тега <a> дозволяє також створювати ***гіперпосилання на конкретні мітки в зовнішніх файлах.*** В цьому випадку ім'я мітки відділяється від імені файлу символом «#». Наприклад, href="index.html#labell" заслання на мітку label1 в зовнішньому файлі index.html.

Колір гіперпосилань в документі HTML указується в теге <body> за допомогою атрибутів: link= просто заслання, alink= виділене заслання і vlink= вже відвідуване заслання.

За допомогою атрибуту target= тега <a> можна вказати ***місце завантаження заслання***: target="_parent" в поточне вікно браузера, або target="_blank" - в нове.

Атрибут title= тега <a> служить для ***створення підказки***, яка спливає при виділенні гіперпосилання. Така підказка корисна, наприклад, у разі, коли як гіперпосилання використовується картинка.

Гіперпосилання на адреси електронної пошти указується небагато інакше, ніж на інші документи, наприклад

Мой e-mail.

Тут «mailto:» записується обов'язково, і точно так, як вказано. Так само створюються гіперпосилання на інші служби Internet, наприклад, news-конференции, тільки замість «mailto:» пишеться «news:», а замість адреси електронної пошти - ім'я групи новин.

Графічні об'єкти в HTML-документ вставляються за допомогою тега <img>. При цьому можна використовувати такі формати файлів зображень, як ***.GIF, .JPG, .JPEG і PNG.***

Атрибут src= є обов'язковим і указує ***ім'я файлу,*** що містить графічне зображення.

Вирівнювання зображення виконується за допомогою атрибуту align=. Так значення атрибуту align= left вирівнює зображення по лівому краю вікна, align= right - по правому.

Рамка довкола зображення встановлюється за допомогою атрибуту border=. Її товщина указується в пікселях.

Атрибути height= і width= указують ***розмір зображення*** в пікселях по висоті і ширині, відповідно. Завжди бажано явно встановлювати ці атрибути для прискорення завантаження сторінки браузером. Атрибути hspace= і vspace= указують розмір вільного місця в пікселях зліва і справа, і зверху і знизу від зображення, відповідно.

Розглянемо декілька прикладів.

Приклад 1. Створення гіперпосилань в межах однієї теки

1. Створіть в теці для HTML-документов файл hello.html, наприклад, з вітанням.
2. У цій же теці створіть Web-страничку такого вмісту і назвіть її index.html.

```
<html>
<head>
<title>Главная страница</title>
</head>
<body>
Приветствую Вас на своей страничке!<br><br>
Вот мое <a href="hello.html">приветствие</a>
</body>
</html>
```

3. Клацніть по засланню на слові «вітання». В результаті повинен відкритися документ hello.html з п.1.

Приклад 2. Створення гіперпосилань в межах різних тек

1. У теці HTML створіть теку hello і перемістите туди файл hello.html.
2. У файлі index.html замініть рядок:

   «Ось моє приветствие<a>»

   на рядок:

   «Ось моє приветствие<a>

3. Клацніть по засланню на слові «вітання», в результаті повинен знову відкритися файл hello.html.
4. В кінці файлу hello.html додайте такий рядок:

Вернутъся до головної сторінк е

5. Перевірте працездатність заслань.

Приклад 3. Вставка малюнків

1. За допомогою стандартної програми Paint створіть невеликий малюнок, (наприклад, кружок червоного кольору) і збережете його у форматі .gif у теці для HTML-документов під ім'ям imgl.gif .
2. Відредагуйте файл index.html, помістивши після рядка:

   Вітаю Вас на своїй сторіночці!

   тег вставки створеного малюнка imgl.gif:

   <img src="imgl.gif">

.
3. У браузері Перегляньте файл index .html.

Приклад 4. Використання малюнків як гіперпосилання

1. Рядок файлу index.html:

   <img src = "imgl.gif">

   відредагуйте таким чином:

   <img src="imgl .gif">

.
2. Відновить файл index.html у вікні браузера. Звернете увагу на рамку вокруг малюнка – так відбувається тому, що малюнок використовується в качестве гіперпосилання. Для того, щоб прибрати рамку необхідно вказати ее нулевую ширину за допомогою атрибуту border="0" тега <img>.

Приклад 5. Створення гіперпосилань на об'єкти усередині документа

1. У файл index .html внесіть такі зміни:
   а) рядок «Вітаю Вас на своїй сторіночці!» помітьте за допомогою атрибуту name="top" тега <a>
   б) за допомогою тега <Ьг> в кінці тіла сторінки, перед тим, що закривається тегом </body>, вставте 5 порожніх рядків
   в) перед тим, що закривається тегом </body> вставте заслання на початок документа
   г) саме заслання на початок сторінки помітьте за допомогою атрибуту id="bottom".

   Після внесених змін файл повинен виглядати таким чином:

```
<html>
<head>
<title>Главная страница</title>
</head>
<body>
<a name="top">Приветствую Вас на своей страничке!</a><br><br>
<a href="hello/hello.html" title="Перейти к приветствию"><img
src="img1.gif"></a><br><br>
Вот мое <a href="hello/hello.html">приветствие</a>
<br>
<br>
<br>
<br>
<br>
<a id="bottom" href="#top">Перейти в начало документа</a>
</body>
</html>
```

2. Оновіть файл index.html у вікні браузера.
3. Зменшите розмір вікна браузера по висоті на стільки, щоб файл index.html у нього повністю не поміщався.
4. За допомогою вертикальної смуги прокрутки покажіть закінчення документа.
5. Перевірте роботу заслання на початок документа.
6. Відредагуйте файл hello.html, вніcши до нього, перед тегом </p>, що закривається, гіперпосилання на мітку «bottom» у файлі index.html. В результаті файл hello.html матиме такий вміст:

```
<html>
<head>
<title>Мой первый документ HTML</title>
</head>
<body>
<p align="center">
<font color="#008080" size="7"><b>Всем</b></font><br>
<font size="6"><i>огромный привет!</i></font><br>
<a href="../index.html">Вернуться к главной странице</a><br>
<a href="../index.html#bottom">В конец главной страницы</a>
</p>
</body>
</html>
```

7. Перевірте роботу всіх заслань.

Тема 8. ФРЕЙМИ. ТАБЛИЦІ

Використання фреймів

Окрім звичайних HTML-документов, існують також ті, що містять ***фрейми.*** На відміну від звичайних документів, вони не містять ніякій інформації для надання користувачеві, а ***служать для розділення робочої зони вікна браузера на декілька окремих вікон***, в які і завантажуються документи, що несуть інформацію. Тобто, фрейми дозволяють відкрити у вікні браузера не один, а одночасно декілька документів. Наприклад, документ menu.html, який містить меню, logo.html – документ, який містить логотип, шапку сторінки, і content.html – документ з безпосереднім вмістом сайту.

Документ, що містить фрейм, не може мати тіла, тобто тега <body>...</body> – він замінюється тегом <frameset>...</frameset>. При цьому у вікні браузера виводяться документи, вказані в тегах <frame> з контейнера <frameset>.. .</frameset>.

Для розділення робочої зони вікна браузера ***на вертикальні вікна*** (стовпчики) служить атрибут cols= тега <frameset>, а горизонтальні (строчки) – rows=.

Розміри окремих фреймів указуються через кому або в пікселях, або у відсотках від розміру робочої області браузера, або символом «*» – все, що залишилося в робочій області. При цьому, якщо розміри фреймів задаються у відсотках від робочої області браузера, то необхідно стежити, щоб їх сума дорівнювала 100%, а якщо в пікселях – розміру робочої області. Тому, щоб не помилитися розмір один з фреймів указується символом «*». Наприклад, запис rows="*,3*" говорить про те, що нижній фрейм буде в 3 рази вище верхнього. Одночасна вказівка атрибутів cols= і rows= приводить до табличного розділення робочої області на фрейми.

Для того, щоб кількість фреймів по горизонталі і по вертикалі було різним, застосовується вкладений один в одного запис тегов <frameset>...</frameset>.

Атрибут тега <frameset> border= указує ***товщину рамок в пікселях довкола фреймів***, а bordercolor= – їх ***колір***. За умовчанням користувач може змінювати розміри фреймів, перетягуючи мишею їх кордони. Для заборони зміни розміру фрейма служить атрибут noresize тега <frame>.

Атрибут тега <frame> scrolling= управляє створенням ***смуг прокрутки***. За умовчанням він має значення scrolling="auto", тобто якщо вміст документа не поміщається у фрейм, то в нім з'являються смуги прокрутки; значення «yes» указує на те, що смуги прокрутки відображуватимуться завжди, навіть якщо

фрейм більше розміру документа, що відображується в нім, а «no» – смуги прокрутки не відображуватимуться ніколи.

Атрибути тега <frame> marginwidth= та marginheight= служать для вказівки ***відступів в пікселях по горизонталі і по вертикалі***, відповідно, між кордоном фрейма і його вмістом. Атрибут тега <frame> name= служить для привласнення імені фрейму. Це необхідно, зокрема, якщо в нього завантажуватиметься документ по гіперпосиланню, яке відображується в іншому фреймі. Якщо ж ім'я не указується, то за умовчанням документ по гіперпосиланню вантажиться в той же фрейм, що і гіперпосилання.

Плаваючі, або ***вбудовані, фрейми*** дозволяють вбудовувати один документ в інший на зразок матрьошок. Вони створюються за допомогою тега <iframe>.. .</iframe>. При цьому не потрібно створювати окремий фреймосодержащий документ, і прописувати в нім фреймову структуру сторінки — тег <iframe>.. .</iframe> дозволяє вставити один HTML-документ в інший. Якщо браузер користувача не підтримує плаваючих фреймів, то відповідне повідомлення необхідно помістити усередині контейнера <iframe>.. .</iframe>. Розміри плаваючого фрейма указуються в атрибутах width= і height=.

Для роботи із старими браузерами, що не підтримують роботу фреймів, служить тег <noframes>.. .</noframes>. Він записується в самому кінці документа, після останнього тега </frameset>, що закриває. Усередині контейнера <noframes>.. .</noframes> розміщується або альтернативний вміст Web-сторінки, або текст пояснення.

Використання таблиць

У HTML таблиці використовуються не лише традиційно, як метод представлення даних, але і як ***засіб розмітки Web-сторінк***. Опис таблиць повинен розташовуватися внутрі розділу документа <body>. Документ може містити довільну кількість таблиць, причому допускається вкладеність таблиць один в одного.

Кожна таблиця повинна ***зачинатися тегом <table> і завершуватися тегом </table>.*** Усередині цієї пари тегов розташовується вміст таблиці.

Будь-яка таблиця складається з однієї або декількох рядків, в кожній з яких задаються дані для окремих вічок. ***Кожен рядок*** таблиці починається тегом <tr> (Table Row) і завершується тегом </tr>. ***Окреме вічко*** в рядку обрамляється парою тегов <td>.. .</td> (Table Data) або <th>.. .</th> (Table Header). Тег <th> використовується зазвичай для елементів-заголовків таблиці, а <td> – для вічок-даних. Відмінність у використанні полягає лише в типові шрифту,

використовуваного за умовчанням для відображення вмісту вічок, а також в розташуванню даних усередині вічка. ***Вміст вічок типа <th>*** відображується ***напівжирним*** (Bold) шрифтом і розташовується ***по центру*** (align="center", valign="middle"). ***Вічка, визначені тегом <td***>, за умовчанням відображують дані, ***вирівняні вліво*** (align="left") і ***посередине*** (valign="middle") ***у вертикальному напрямі.*** Теги <td> і <th> не можуть з'являтися поза описом рядка таблиці <tr>. Теги </td>, </th> і </tr>, що завершують, можуть бути опущені. В цьому випадку кінцем опису рядка або вічка є початок наступного рядка або вічка, або кінець таблиці. Тег таблиці </table>, що завершує, не може бути опущений.

Кількість рядків в таблиці визначається числом тих тегов <tr>, що відкриваються, а кількість стовпців - максимальною кількістю тегов <td> або <th> серед всіх рядків.

Частка вічок може не містити ніяких даних, такі вічка описуються парою тегов <td>...</td>, які розташовані поспіль. Якщо одна або декілька вічок, розташованих в кінці якого-небудь рядка, не містять даних, то їх опис може бути опущен, а браузер автоматично додасть необхідну кількість порожніх вічок. Звідси витікає, що ***побудова таблиць, в яких в різних строчках розташовується різна кількість стовпців одного і того ж розміру, не дозволяється.***

Таблиця може мати ***заголовок***, який розташовується між парой тегов <caption>...</caption>. Опис заголовка таблиці повинен розташовуватися усередині тегов <table>.. .</table> до першого <tr>. За умовчанням текст заголовка таблиці розташовується над нею (align="top") і центрується по горизонталі. Можна так само вказати розташування заголовка під таблицею – <align="bottom".

За умовчанням розмір таблиці встановлюється залежно від її вмісту так, щоб вона відображувалася щонайкраще. За допомогою атрибутів width= і height= можна явно вказати розміри вічка, рядка і всієї таблиці. Їх значення можуть бути вказані або в пікселях, або у відсотках. Горизонтальне вирівнювання таблиць встановлюється атрибутом align=, а вертикальне – атрибутом valign=.

Перераховані вище теги можуть мати і інші атрибути, число і значення яких різні. Проте в простому випадку теги використовуються без атрибутів, які набувають значень за умовчанням.

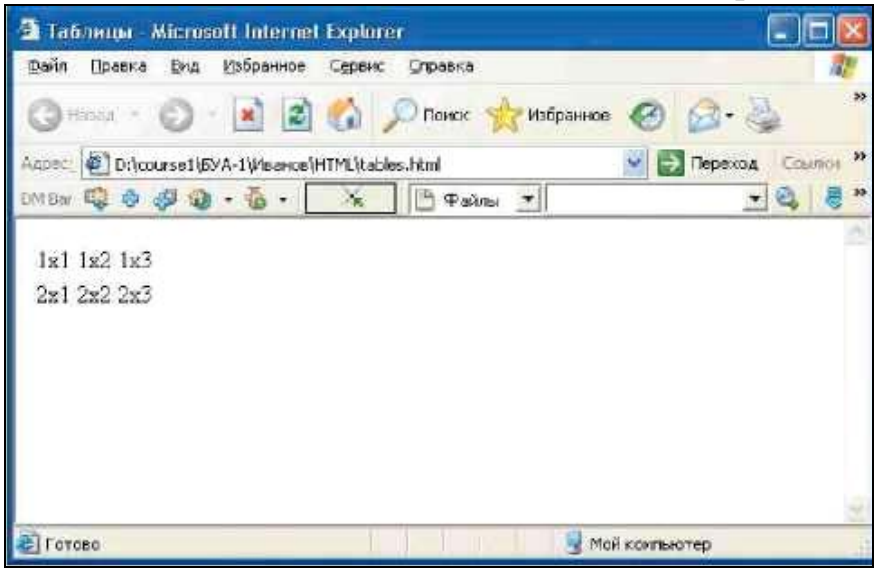
Розглянемо декілька прикладів.

Приклад 1. Створення простої таблиці 2x3

1. У теці HTML-документов створіть файл tables.html, який містить одну просту таблицю, що складається з 2-х рядків і 3-х стовпців, в кожному вічку якої записані її «координати», наприклад 1x1 для – 1-го вічка 1-го рядка, 1x2 – для 2-го вічка 1-го рядка, і т.д.:

```
<html>
<head>
<title>Таблицы</title>
</head>
<body>
<table>
<tr>
<td>1x1</td><td>1x2</td><td>1x3</td>
</tr>
<td>2x1</td><td>2x2</td><td>2x3</td>
</tr>
</table>
</body>
</html>
```

2. Перегляньте створену таблицю в браузері. Якщо не було допущено помилок, то вона повинна виглядати приблизно так:



Невитіюватий вид таблиці обумовлений тим, що так виглядають всі таблиці з параметрами тегов, прийнятими за умовчанням.

Приклад 2. Додавання рамок

1. У файлі tables.html відредагуйте тег створення таблиці <table>, додавши до нього атрибут управління рамкою border=, вказавши її товщину в 1 піксель:
   <table border="l">
2. Збережіть відредагований файл tables.html. Оновіть вміст вікна браузера, клацнувши по кнопці Відновити, і Ви побачите результат.

Приклад 3. Порожні вічка

1. Відредагуйте файл tables.html, «пропустивши» в нім 2-е вічко в 1-му рядку, тобто тег описи 1-го рядка виглядатиме таким чином:
   <td> 1x1 </td><td> 1x3</td>
2. Перегляньте отриманий результат. Він повинен виглядати приблизно так:

1x1	1x3	
2x1	2x2	2x3

3. Знову відредагуйте файл tables.html, вставивши в потрібному місці «порожнє» вічко 1x2. В результаті тег опису 1-го рядка повинен виглядати таким образом: <td>1xl</td> <td></td> <td>1x3</td>.
4. Перегляньте зміни в браузері. В результаті місце під вічко 1x2 буде виділено, але рамки довкола неї не буде.
5. Для того, щоб довкола порожнього вічка відображувалася рамка, вона повинна містити, принаймні, один з символів, що не відображуються. Наприклад, нерозривний пропуск – . В результаті опис 1-го рядка повинен виглядати таким чином <td>lxl</td><td> </td><td>lx3</td>

Приклад 4. Об'єднання вічок

1. Відредагуйте файл tables.html так, щоб 1-е і 2-е вічка 1-ої лави були об'єднані. Для цього необхідно відредагувати опис 1-го рядка наступним образом:

<td colspan="2"> 1xl</td><td>lx2</td>

2. Перегляньте отриманий результат в браузері. Він повинен виглядати приблизно так:

1x1		1x2
2x1	2x2	2x3

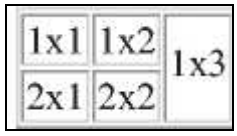
3. Відредагуйте опис 1-го рядка таким чином – тобто ви «забули» видалити вічко 1x3:

<td colspan="2"> 1xl</td><td>lx2</td><td>lx3</td>

4. Перегляньте отриманий результат. Він повинен виглядати приблизно так:

1x1		1x2	1x3
2x1	2x2	2x3	

5. Самостійно, за допомогою атрибуту rowspan=, об'єднайте вічка 1x3 і 2x3 так, щоб отримати таку таблицю:

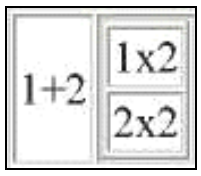


Приклад 5. Вкладені таблиці

1. Створіть таблицю, що складається з 1-го рядка і 2-х стовпчиків. У другій стовпчик помістить таблицю, що складається з 2-х рядків і 1-го стовпчика. Для цього відредагуйте файл tables.html таким чином:

```
<html>
<head>
<title>Таблицы</title>
</head>
<body>
<table border="1">
<tr>
<td>1+2</td>
<td>
<table border="1">
<tr><td>1x2</td></tr>
<tr><td>2x2</td></tr>
</table>
</td>
</tr>
</table>
</body>
</html>
```

2. Перегляньте отриманий результат. Він повинен виглядати приблизно так:



Вкладені таблиці і об'єднання вічок – два способи вирішення однієї задачі. Який з них вважати за краще, залежить від задачі, яка вирішується. Об'єднання вічок – простіший метод, і застосовується в простіших ситуаціях; вкладені таблиці дозволяють гнучкіше конструювати складні таблиці.

Приклад 6. Додавання заголовків

1. Відредагуйте файл tables.html, додавши до таблиці заголовок «Баланс», а до 1-го і 2-го стовпчиків – «Прихід» і «Витрата», відповідно; 3-й стовпчик прибіриь зовсім. В результаті він повинен мати такий вміст:

```
<html>
<head>
<title>Таблицы</title>
</head>
<body>
<table border="1">
<caption><h3>Баланс</h3></caption>
<tr>
<th>Приход</th><th>Расход</th>
</tr>
<tr>
<td>1000</td><td>2000</td>
</tr>
</table>
</body>
</html>
```

Перегляньте отриманий результат в браузері. Він повинен виглядати приблизно так:

Заголовок таблиці додається за допомогою тега <caption>.. .</caption>. За умовчанням він відображується основним текстом таблиці. Для того, щоб його виділити, був використаний тег заголовка <h3>...</h3>. Заголовки стовпчиків виводяться за допомогою тега <th>...</th>. За умовчанням він виводиться напівжирним зображенням, вирівняним по середині вічка. Вміст вічок, створених за допомогою тега <td>.. .</td>, виводиться нормальним шрифтом, вирівняним по лівому краю вічка. Зверніть також увагу, що розмір вічок був підібраний під їх найбільший вміст.

Приклад 7. Використання таблиць для форматування Web-сторінк

1. У теці для HTML-документов перевірте наявність файлу logo.gif - логотипу (кухоль зеленого кольору), або створіть його заново за допомогою графічного редактора Paint.

2. У тій же теці перевірте наявність файлів liza.html, ira.html, katya.html, sveta.html і natasha.html, які містять інформацію про Лізу, Іру, Катю, Світу і Наташу, відповідно. Якщо таких файлів немає, то створіть їх. Наприклад, для Лізи він повинен мати такий вміст:

```
<html>
<head>
<title>Лиза</title>
</head>
<body>
Лиза
</body>
</html>
```

3. Відредагуйте файл tables.html так, щоб він мав наступний вміст:

```
<html>
<head>
<title>Таблицы</title>
</head>
<body>
<table width="400" height="100%">
<tr valign="top">
<td width="80">
<h3>Меню</h3>
<a href="liza.html" target="content">Лиза</a><br>
<a href="ira.html" target="content">Ира</a><br>
<a href="katya.html" target="content">Катя</a><br>
<a href="sveta.html" target="content">Света</a><br>
<a href="natasha.html" target="content">Наташа</a>
</td>
<td>
<table>
<tr><td width="80" height="80">
<img src="logo.gif">
</td></tr>
<tr><td>
<iframe src="liza.html" name="content" frameborder="0">
<font color="ff0000">Чтобы увидеть эту страницу обновите браузер</font>
</iframe>
</td></tr></table></td></tr></table>
</body>
</html>
```

4. Перегляньте відредагований файл в браузері. Він повинен виглядати і працювати так, як і в попередній лабораторній роботі про фрейми.
5. Самостійно видаліть, або змініть, значення атрибутів width=, height= і valign=, які служать для явной установки ширини, висоти і вертикального вирівнювання, відповідно, і перегляньте, як «попливе» таблиця при зміні розмірів вікна браузера.

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. Основи компьютерної графіки: [Навч. посіб.] / В.С.Березовський, В.О.Потієнко, І.О.Завадський. – К.:Вид.група BHV, 2009. – 400 с.
2. Рейнбоу В. Компьютерная графика. Энциклопедия. − СПб.:Изд-во "Питер", 2003. − 768с.
3. А.П.Сергеев, С.В.Кущенко. Основы компьютерной графики. Adobe Photoshop и CorelDRAW – два в одном. Самоучитель. – М.:Диалектика, 2006. – 544 с.
4. Web-дизайн с нуля!: книга + видеокурс / П.П.Константинов [и др.]. – М.:Лучшие книги, 2009. − 304 с.
5. Матюшок В.М. Информатика для экономистов Учебник. – Москва, ИНФРА-М, 2006. – 880 с.
6. Бєльков Д.В., Єдемська Є.М., Незамова Л.В. Створення веб-документів засобами мови HTML. Методичні рекомендації по дисципліні «Економічна інформатика». Частина 1. – Донецьк: ДонНТУ, 2010. – 107 с.
7. Маслова Н.О., Масло С.В., Павлиш В.М. Методичні вказівки та завдання до лабораторних робіт з курсу „Комп’ютерна графіка”. – Донецьк: ДонНТУ, 2008. – 56 с.
8. Б.И.Погребняк, А.В.Белогурова. Методические указания для выполнения лабораторных работ по курсу „Экономическая інформатика”. Основы WEB-дизайна.. − Харьков: ХНАГХ, 2008. − 47 с.
9. http://www.intuit.ru/department/internet/htmlbasics