МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Державний заклад "Південноукраїнський національний педагогічний університет

імені К.Д. Ушинського"

Кафедра прикладної математики та інформатики

Мазурок Тетяна Леонідівна

Царенко Микола Олександрович

СИСТЕМИ УПРАВЛІННЯ НАВЧАННЯМ

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ ДО ЛАБОРАТОРНИХ РОБІТ

Перша частина

Для студентів напряму підготовки 014 Середня освіта (Інформатика) другий(магістерський) рівень

Фізико-математичного факультету

Рекомендовано методичною радою університету

ОДЕСА 2021

Рекомендовано вченою радою Державного закладу "Південноукраїнський національний педагогічний університет

імені К.Д. Ушинського" протокоп№* від 28.12.2020 року

Мазурок Т. Л., Царенко М. О. «СИСТЕМИ УПРАВЛІННЯ НАВЧАННЯМ»

Методичні рекомендації Одеса - ПНПУ імені К.Д. Ушинського;- 87с.

ПРАКТИЧНА РОБОТА № 1

Тема: **Розробка моделі освоєння навчального матеріалу: Розробка структурно-логічної схеми навчальної теми**

1. ЗАГАЛЬНІ ВІДОМОСТІ

Мета роботи: Вивчити закономірності складання діагностичних цілей і отримати практичні навички вивчення теми, розділяти її на навчальні елементи і виділяти структурні зв'язки.

Оснащення: Індивідуальні методичні вказівки для самостійної роботи, навчальна програма з предмету, науково-методична література, періодична преса.

2. ОСНОВНІ ТЕОРЕТИЧНІ ПОЛОЖЕННЯ

Слід враховувати, що важливим моментом в описі цілей навчання є аналіз самих об'єктів діяльності фахівця, названих навчальними елементами (НЕ). Саме ними окреслюється і гарантується область можливої діяльності в структурі навчального предмета або системи навчальних предметів.

Під *навчальними елементами* (НЕ), як відомо, розуміються існуючі поза і незалежно від суб'єкта, що пізнає об'єктивні *явища* і *предмети* навколишнього світу, пізнані людством у вигляді їх властивостей, зв'язків і відносин і відображені у вигляді наукових понять і теорій, а також *способи*, і *методи* використання того чи іншого, тобто методи конкретної *діяльності* людей. З НЕ складається зміст навчання. Виступаючи по відношенню один до одного в об'єктивному взаємозв'язку, НЕ становить певну структуру, яка може бути представлена наочно у вигляді графа рисунок 1.

Як приклади НЕ можна назвати різні явища, процеси, що відбуваються в машині, на виробництві, громадському житті; НЕ являються різні частини

машин, технологічні інструменти і способи роботи та інше. При цьому різні НЕ в сукупності утворюють різні поєднання і знаходяться на певній градації графа - структури предмета.

Для кожного виду діяльності, а значить і для кожного навчального предмета можна виділити *експертним* шляхом цілком певну кількість НЕ, в першому наближенні характеризують *обсяг* навчання або *обсяг* засвоєння.

Дуги графа показують зв'язки, що існують між НЕ, і їх взаємозалежності. Числом зв'язків даного НЕ з іншими НЕ визначається його значимість для засвоєння даного виду діяльності. Прийнято в графі указувати лише прямі, ієрархічні зв'язки між НЕ і не вказувати опосередковані зв'язки, тому що вони враховуються структурою графа і мають значення лише в розробці послідовності вивчення НЕ.

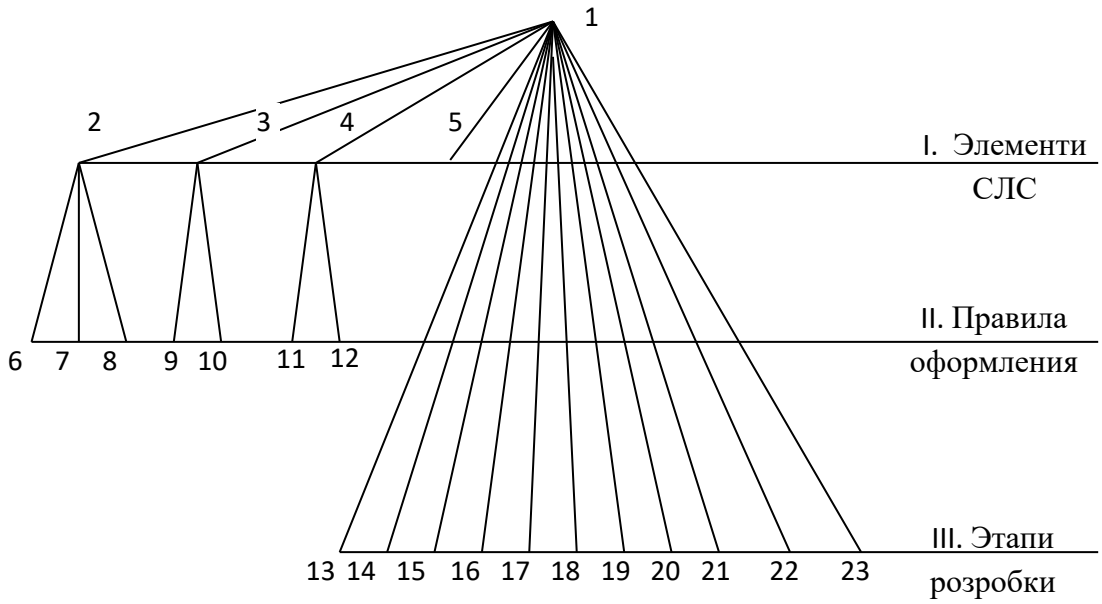


Рисунок 1.
Граф теми «Розробка структурно-логічної схеми теми».

Побудувавши граф навчальної дисципліни, легко порахувати *загальне число навчальних елементів* і ввести деякі характеристики навчання.

Слід звертати увагу на необхідність точної відповідності змісту підстави і змісту НЕ на відповідній градації графа.

У графі показана узагальнена логічна структура видів і методів діяльності якими повинен опанувати кожен викладач. Ці види і методи діяльності є навчальними елементами, які в свою чергу можуть бути диференційованими залежно від спрямованості в підготовці викладача. Завдання зводиться до того,

щоб розкрити зміст того досвіду, який буде формуватися в учнів, що навчаються діяльності по предмету, що викладається.

При побудові графа дотримуються правил побудови ієрархічних деревовидних структур:

- граф має тільки один корінь, один НЕ - назва теми;
- відсутні окремі (висячі) вершини, не пов'язані з вищестоячими НЕ, крім кореня;
- зв'язок здійснюється тільки зверху - вниз;
- нижчий НЕ може бути пов'язаний тільки з одним вищим НЕ;
- угрупування НЕ на одному рівні здійснюється за якою-небудь спільною ознакою (загальної основи);
- вищі НЕ не повинні бути пов'язані менш ніж з двома нижчестоячими НЕ.

Паралельно з побудовою графа складають таблицю НЕ, в яку вносять найменування НЕ (табл. 1).

Таблиця 1. Специфікація до графу теми «Розробка структурно-логічної схеми навчальної теми».

№ п / п	Навчальні елементи	Діагностичні цілі навчання за якістю засвоєння (рівень засвоєння)
1	2	3
1.	Розробка графа теми	1
2.	Вершини - навчальні елементи	1
3.	Ребра	1
4.	Порядки, основи	1
5.	Специфікація	1
6.	Позначення навчальних елементів	2
7.	Розміщення навчальних елементів	2
8.	Запис навчальних елементів	2
9.	Показ зв'язків	2
10.	Окреслення ребер	2
11.	Позначення порядків	2
12.	Виділення основ	2
13.	Виписка з програми ЧПУ	3
14.	Складання робочого варіанта	3
15.	Складання тез теми	3
16.	Виписка основ	3
17.	Визначення кількості і послідовності порядків	3

18.	Підготовка бланка для графа і форми специфікації	3
19.	Розміщення і запис НЕ. Оформлення ребер	3
20.	Перевірка правильності оформлення	3
21.	Звірка із програмою ЧПУ	3
22.	Призначення цілей навчання	3
23.	Експертна перевірка графа	3

Рекомендується наступна технологія практичної роботи. Беруть два аркуші паперу. На одному аркуші будують граф (зверху - вниз), на іншому - послідовно вписують рядки таблиці НЕ. Аналогом цього процесу являється складання змісту навчального посібника, коли його зміст попередньо дроблять на глави, параграфи і т.д. Однак при побудові графа вмісту навчального матеріалу, на відміну від складання змісту, немає потреби піклуватися про послідовність викладу НЕ. Важливо відобразити лише ієрархічну структуру навчального матеріалу.

Уміти уявляти навчальні теми в графовій формі можна успішно використовувати при підготовці доповідей, виступів, дискусій і «круглих столів», при написанні рефератів.

3. ПОРЯДОК ВИКОНАННЯ РОБОТИ

- Вибрати будь-яку тему предмета за запропонованою навчальною програмою, розраховану на 4-8 годин навчального часу.
- Виконати наступні етапи, посилаючись на рисунок 1 (навчальні елементи порядку III):

НЕ 13. Виписати з програми відповідного навчального предмета копію тексту обраної теми.

НЕ 14. Скласти робочий варіант тексту теми. Для цього потрібно співставити виписку з професійною (кваліфікованою) характеристикою фахівця що готується і проаналізувати, чи міститься в тексті виписки все, що з даної теми необхідно для підготовки кваліфікованого фахівця, чи немає дублювання з іншими темами даного предмета або з темами раніше або паралельних предметів, що вивчаються. Разом з тим, потрібно забезпечити скрупульозну селекцію прийнятої до вивчення інформації за правилом геологів: «в рюкзак не те, що може стати в нагоді, а без чого не обійтися».

У робочому варіанті повинні бути виявлені (виділені) підстави графа.

НЕ 15. Скласти тези або конспект робочого варіанту теми.

НЕ 16, 17. Користуючись робочим варіантом, виписати підстави, визначити кількість порядків і їх послідовність.

НЕ 18. Підготувати бланк для чернетки графа: у верхній частині листа 200 х 300 мм, розташованого горизонтально, приблизно по середині показують

вершину першого НЕ і наносять на однакових інтервалах горизонталі по числу порядків. На горизонталях підписують номери і назви базових основ.

На окремому аркуші за формою таблиці 1 лекції «Проектування педагогічних програмних засобів. Відбір і структурування навчального матеріалу педагогічних програмних засобів» підготувати бланк специфікації.

НЕ 19. Вести запис навчальних елементів теми в специфікації з одночасним розміщенням їх вершин на відповідних порядках графа, записом номерів, оформленням зв'язків (ребер).

НЕ 20. Перевірити правильність оформлення графа і специфікації з робочим варіантом програми.

НЕ 21. Звірити підготовлену чернетку графа і специфікації з робочим варіантом програми.

НЕ 22. Призначити і зафіксувати по кожному навчальному елементу індексами (як в табл. 1) цілі навчання.

НЕ 23. Чистові екземпляри графа і специфікації слід піддати експертизі. В якості експерта в даному випадку виступає викладач.

По кожному НЕ теми (див. Специфікацію, виконану Вами за завданням), призначити необхідний рівень засвоєння.

4. Контрольні питання

- Що таке граф?
- Для чого використовується графова форма подання навчального материалу?
- Що мається на увазі під навчальними елементами?
- У чому полягає суть побудови графа теми?
- Що така підстава графа?
- Що таке порядок графа?
- Що таке специфікація до графу, для чого вона потрібна?

Практична робота № 2

Тема: **Розробка алгоритмів засвоєння знань**

1. ЗАГАЛЬНІ ВІДОМОСТІ

Мета роботи: Вивчити алгоритми засвоєння знань і отримати практичні навички використання психологічних закономірностей засвоєння знань, що дозволяють підвищити ефективність процесу навчання.

Оснащення: індивідуальні методичні вказівки для самостійної роботи, науково-методична література, періодичні видання.

2. ОСНОВНІ ТЕОРЕТИЧНІ ПОЛОЖЕННЯ

При розробці навчальної роботи доцільно враховувати психологічні закономірності засвоєння знань, встановлені в педагогічній психології і дозволяють підвищити ефективність процесу навчання. У даній практичній роботі розглядаються деякі найбільш відомі та "технологічні" теорії засвоєння знань.

Бихевиористска теорія навчання. У біхевіоризмі (від лат. Behavior - поведінка) не розглядаються внутрішні процеси людського мислення. Вивчається поведінка, яке трактується як сума реакцій на будь-які ситуації. Один з основоположників біхевіоризму Е.Л. Торндайк (1874-1948) вважав, що навчання людини повинно будуватися на базі чисто механічних, а не свідомих принципів. Тому він намагався описати навчання людини за допомогою простих правил, справедливих одночасно і для тварин. Серед цих правил виділимо два закони, які послужили платформою для подальшого розвитку теорії навчання.

Перший з них, названий законом тренування, говорить про те, що, чим частіше повторюється певна реакція на ситуацію, тим міцніше зв'язок між ними, а припинення тренування (повторення) призводить до ослаблення цих зв'язків.

Другий закон був названий законом ефекту: якщо зв'язок між ситуацією і реакцією супроводжується станом задоволеності (задоволення) індивіда, то міцність зв'язку з цим зростає і навпаки: міцність зв'язку зменшується, якщо результат дії призводить до стану незадоволеності. Спираючись на ці закони, послідовник Торндайка Б.Ф. Скіннер розробив на початку 50-х років досить технологічну методику навчання, названу надалі лінійним програмуванням. В основу своєї методики Скіннер поклав універсальну формулу:

С → Р → П

де ***С*** - ситуація;
Р - реакція;
П - підкріплення.

Навчальний матеріал Скіннер пропонував розбивати на дрібні дози, кожна з яких повинна містити одну ситуацію. Ситуації повинні бути настільки простими (що майже автоматично забезпечувалося малою кількістю доз навчального матеріалу), щоб реакції на них практично завжди були правильними. На думку Скіннера, правильне виконання навчального завдання вже само по собі є позитивним підкріпленням і призводить слухача в стан задоволеності.

У текстах програмованих навчальних посібників Скіннера мали місце пропуски (ситуації) - один пропуск на фразу з 2-3 рядків. Пропущені слова росташовувались на полях сторінки. Учень, вивчаючи таку допомогу, спочатку закривав поля, читав текст, вставляючи пропущені слова, і відразу ж перевіряв себе, відкриваючи відповіді. Тексти навчальних посібників були написані таким чином, щоб в процесі їх читання забезпечувалося багаторазове повторення всіх істотних елементів навчального матеріалу.

Застосування програмованих посібників Скіннера в професіонально-технічних училищах США виявилося успішним: істотно скоротився час навчання, підвищилася кваліфікація учнів робочих. Однак тут же знайшлися і недоліки методики лінійного програмування:

нудність і механістичність програмованих текстів;

- відсутність системності, цілісності в сприйнятті навчального материалу (велика кількість дрібних доз не сприяє узагальненям);
- правильність виконання простих завдань є позитивним підкріпленням лише на перших порах читання посібника, в подальшому правильне виконання простих ситуацій вже не приносить почуття задоволення;

- відсутність адаптації (всі учні виконують одну і ту ж програму, йдуть по одній лінії).

Значна частина цих недоліків була усунена в запропонованій Н. А. Краудером схемою розгалуженого програмування (рис. 1). Краудер запропонував збільшити дозу інформації (***І1*** , ***І2*** на рис. 1) з 2-3 рядків у Скіннера до приблизно половини сторінки. Типова ситуація (завдання) у Краудера складалася з питання (***П*** і трьох варіантів відповідей: ***01*** - правильна відповідь, ***02*** - неточна відповідь, ***03*** - неправильна відповідь. При неточній відповіді учень звертався до корегуючої інформації (***К***), при неправильній - йому давалося роз'яснення, допомога (***Р***). При правильній відповіді учень отримував позитивне підкріплення (***П***) і переходив до наступної дози інформації (І***2***). Таким чином, схема розгалудженого програмування мала три шляхи: для сильних, середніх і слабких учнів.

Незважаючи на гостру критику за принципове невтручання в мислення учня (біхевіористи керують лише його поведінкою), біхевіорістска теорія навчання набула широкого поширення і була реалізована в ряді технічних навчальних пристроїв. І в даний час універсальна схема цієї теорії (ситуація - реакція - підкріплення) в її лінійній або розгалуженої формі є стрижневим фрагментом багатьох комп'ютерних навчальних програм (рис. 2).

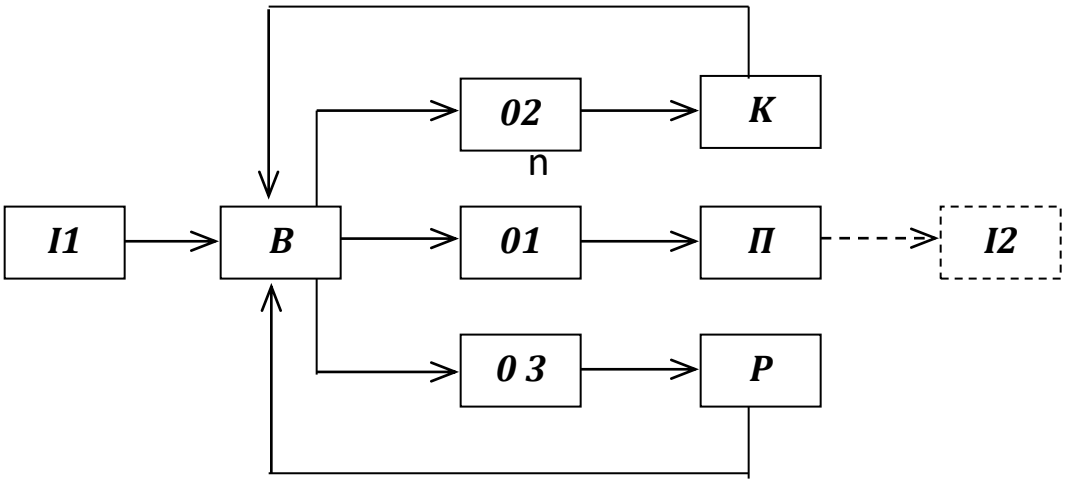


Рисунок 1. Схема розгалудженого програмування.

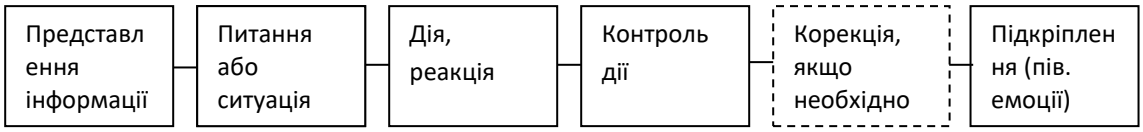


Рисунок 2. Схема засвоєння знань в теорії програмованого навчання.

Асоціативно-рефлекторна теорія засвоєння. Асоціацію в даній теорії визначають як зв'язок між психічними явищами, при наявності якої актуалізація одного явища викликає появу іншого. Таким чином, навчання в асоціативно-рефлекторній теорії трактується як встановлення зв'язків між різними елементами знання. Зв'язки прийнято ділити на зовнішні і внутрішні. Зовнішні зв'язки дають чисто механічне заучування. Наприклад, правило для запам'ятовування колірного спектра: «Кожен Мисливець Бажає Знати, Де Сидить Фазан». Внутрішні ж, логічні зв'язки дозволяють з одних елементів знання отримувати (виводити) інші елементи.

Необхідними умовами для застосування асоціативно-рефлекторної теорії засвоєння є наявність у учнів певного фундаменту знань і володіння ними логічними операціями, які дозволяють пов'язувати між собою раніше вивчені і нові елементи знань. Методику асоціативно-рефлекторного навчання можна представити у вигляді схеми з шести наступних етапів (рис. 3):

1. Актуалізація раніше засвоєних елементів знань (контроль, нагадування).
2. Встановлення зв'язків між раніше засвоєними і новими елементами знань.
3. Фіксація і осмислення нових елементів знань.
4. Закріплення нових знань.
5. Узагальнення раніше засвоєних та нових елементів знань в єдину систему.
6. Закріплення узагальненого знання.

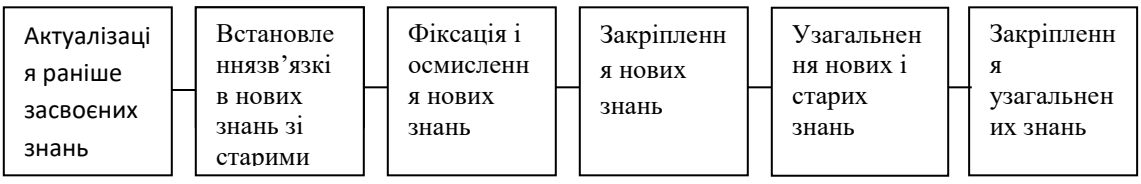


Рисунок 3. Концептуальна схема асоціативно-рефлекторної теорії навчання.

При конкретній реалізації цієї схеми в глобальному сценарії навчальної роботи з навчальною програмою локальні сценарії кожного етапу можуть бути побудовані на основі універсальної біхевіорістськой формули (рисунок 1).

Теорія поетапного формування розумових дій. Основи цієї теорії були закладені П.Я. Гальперіним і надалі були розвинуті в роботах Н.Ф. Тализіна та інших його послідовників. Відповідно до цієї теорії процес навчання доцільно планувати у вигляді схеми, яка складається з шести наступних етапів (рис. 4).

1. Створення мотивації для вивчення навчального матеріалу.
2. Формування орієнтовної основи діяльності, наприклад, вивчення загальної структури навчального матеріалу.
3. Матеріальна чи матеріалізована форма діяльності. На цьому етапі організується навчальна діяльність безпосередньо з досліджуваними матеріальними об'єктами або з їх замінниками: макетами, кресленнями, схемами і т.п.
4. Абстрагуватися від матеріальних об'єктів зовнішньомовної діяльності. Це може бути не тільки промовляння вголос, але і письмо.
5. Абстрагована діяльність, що протікає в формі внутрішньої промови (зовнішня мова подумки).
6. Навчальна діяльність, що протікає в абстрагованій згорнутій, розумовій формі.

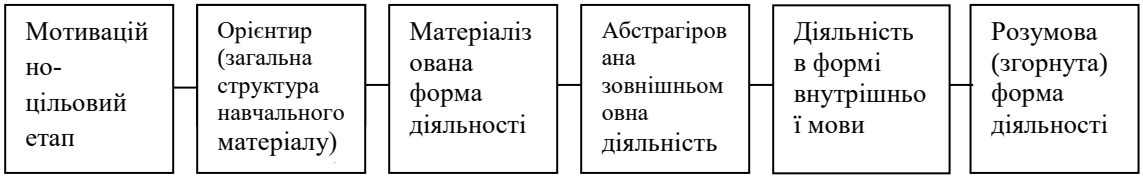


Рисунок 4. Алгоритм поетапного формування розумових дій.

Концепція алгоритмізації. Основна сфера застосування цієї теорії засвоєння - вивчення алгоритмів розв'язання задач. Технологічна схема навчальної роботи за цією теорією складається з п'яти етапів (рис. 5).

1. Усвідомлення області застосування засвоюваних способів.
2. Ознайомлення з алгоритмом рішення задачі в цілому.
3. Навчальна діяльність за алгоритмом з зовнішньою опорою (алгоритм перед очима).
4. Навчальна діяльність за алгоритмом з епізодичною зовнішньою опорою (алгоритму перед очима немає, але є можливість зазирнути в його опис).
5. Навчальна діяльність за алгоритмом без зовнішньої опори.

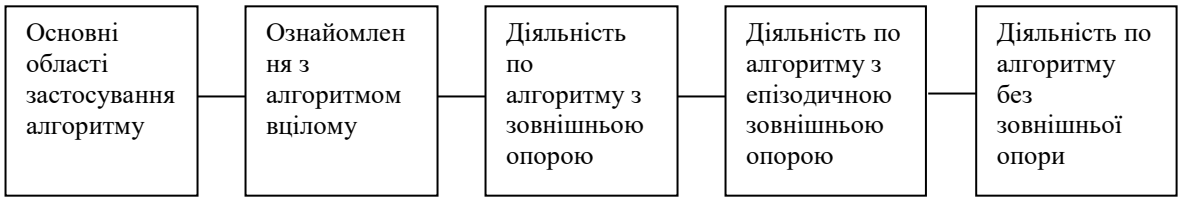


Рисунок 5. Концепція алгоритмізації.

Рекомендації щодо застосування психологічних теорій засвоєння. При проектуванні педагогічних програмних засобів доцільно планувати на початку навчальної роботи створення в учнів мотивації, ознайомлення із загальною структурою навчального матеріалу (теорії алгоритмізації або поетапного формування розумових дій), нагадування, якщо це необхідно, раніше вивченого матеріалу (асоціативно-рефлекторна теорія). При розробці послідовності виконання вправ в ході вивчення окремих навчальних елементів спочатку плануються до виконання вправи зі схемами, кресленнями та іншими графічними ілюстраціями (матеріалізована форма діяльності), а слідом за ними - більш абстрактні вправи. Сценарії кожної вправи доцільно планувати відповідно до

універсальної схеми програмованого навчання. З огляду на подрібнений, порційний характер цієї процедури навчання, необхідно також передбачати проміжні і завершальний узагальніючі етапи.

3. ПОРЯДОК ВИКОНАННЯ РОБОТИ

- Ознайомитися з основними теоретичними положеннями;
- На основі вивченого теоретичного матеріалу проаналізувати схеми засвоєння знань. Результати аналізу занести в таблицю 1.
- Навчальний матеріал, підготовлений в практичній роботі: «Розробка моделі освоєння навчального матеріалу: Розробка структурно-логічної схеми навчальної теми» розібрати і уявити за кожною схемою засвоєння знань;
- Відповісти на контрольні питання.

Таблиця 1.

Теорія засвоєння знань	Суть теорії засвоєння знань	Переваги теорії засвоєння знань	Недоліки теорії засвоєння знань	Відмінності теорії засвоєння знань одне від одного
Бихевиористска теорія навчання:				
Лінійне програмування				
Розгалуджене програмування				
Асоціативно-рефлекторна теорія засвоєння				

Теорія поэтапного формування розумових дій				
Концепція алгоритмізації				

4. Контрольні питання

- У чому полягає суть біхевіорістськой теорії навчання?
- Що таке лінійне програмування (суть, переваги і недоліки)?
- Що таке розгалуджене програмування (суть, переваги і недоліки)?
- У чому полягає суть засвоєння знань в асоціативно-рефлекторній теорії навчання?
- У чому полягає суть засвоєння знань поетапного формування розумових дій?
- У чому полягає суть засвоєння знань в теорії алгоритмізації?

ПРАКТИЧНА РОБОТА № 3

Тема: **Розробка моделі освоєння навчального матеріалу: Розробка матриці відносин черговості навчальних елементів**

1. ЗАГАЛЬНІ ВІДОМОСТІ

Мета роботи: Вивчити послідовність розробки моделі освоєння навчального матеріалу, формування матриці відносин черговості навчальних елементів, і отримати практичні навички обробки матриці віносин черговості і

побудови послідовності вивчення навчального матеріалу у вигляді списку навчальних елементів; навчитися формувати матрицю логічних зв'язків навчальних елементів.

Оснащення: індивідуальні методичні вказівки для самостійної роботи, науково-методична література, періодичні видання.

2. ОСНОВНІ ТЕОРЕТИЧНІ ПОЛОЖЕННЯ

Модель освоєння навчального матеріалу

Серед процедур конструювання матеріалу навчального курсу провідні позиції належать матриці логічних зв'язків дисципліни. Ця матриця являє собою таблицю з двома входами: нульові стовпець і рядок в наявності переліки тем дисципліни, для яких встановлюються логічні зв'язки, причому в клітинах перетину рядків і стовпців проставляється відмітка лише в разі взаємозв'язку тем. Матриця дозволяє не тільки системно відобразити у вигляді наочної таблиці мережу взаємозв'язків тем навчального курсу, а й виділити вузлові теми курсу, значимі в теоретичному та прикладному плані (в цьому випадку рядок має найбільшу кількість відміток).

Матриця логічних зв'язків може бути складена, наприклад, одночасно для всіх спеціальних дисциплін, що визначають зміст професійної підготовки випускників профшкіл. Створення такої росширеної матриці дозволяє дати відповідь на складне питання про достатність, наприклад, загальнонаукового апарату для вивчення учнями спеціальних дисциплін. Дійсно, якщо в стовпці якоїсь теми немає відміток взаємозвязків (або їх мало), то або в даній темі не використовується загальнонаукових матеріалів, або загальнонауковий апарат, необхідний учням для вивчення цієї теми, не включений в програму аналізованого курсу. Нарешті, використання матриці логічних зв'язків для формування системи ключових тем курсу і призводить до подання провідних знань цього курсу у вигляді мережевого гіпертексту - орієнтованого графа, в вершинах якого знаходяться вузлові теми курсу, наочні і технічні засоби, а на ребрах - перехресні посилання між цими системними фрагментами курсу.

Модель змісту навчального матеріалу не містить відповідей на питання про те, в якій послідовності повинні вивчатися навчальні елементи (НЕ) і які логічні зв'язки між ними. Ці питання розглядаються при формуванні моделі освоєння навчального матеріалу. Будемо ілюструвати побудову цієї моделі на фрагменті матеріалу даного розділу (рис. 1 і табл. 1).

До складу моделі освоєння входять матриці відносин черговості і логічних зв'язків НЕ, послідовність вивчення НЕ, граф логічних зв'язків НЕ (рис. 1). Побудову моделі проводять в чотири етапи:

- формування матриці відносин черговості НЕ;
- обробка матриці відносин черговості і побудова послідовності вивчення навчального матеріалу у вигляді списку НЕ;
- формування матриці логічних зв'язків НЕ;

- побудова графа логічних зв'язків НЕ.

Перший і третій етапи є неформальними і виконуються на основі аналізу навчального матеріалу. Матриці відносин черговості і логічних зв'язків НЕ є квадратними. Розмір матриць дорівнює кількості НЕ. Спочатку будують осередки матриць і нумерують їх рядки і стовпці у відповідності зростання НЕ (рис. 1, а і б). Далі построково заповнюють чарунки матриць нулями і одиницями.

При заповненні осередків матриці відносин черговості аналізують просте бінарне відношення черговості між двома НЕ. Одиницю ставлять в клітинку, якщо НЕ, зазначений в номері рядка, повинен вивчатися після НЕ, зазначеного в номері стовпця. Протилежне становище черговості позначають нулем або залишають відповідному полі матриці порожній. Всі осередки головної діагоналі матриці відносин черговості заповнюють одиницями. Осередки матриці, симетричні відносно головної диагонали, повинні мати протилежні відносини (0 або 1). Тому неформальний аналіз парних відносин черговості можна проводити лише для лівого нижнього або для правого верхнього трикутника матриці, заповнюючи її частину, що залишилася формально на основі властивості антисиметрії.

При заповненні матриці логічних зв'язків НЕ ставлять одиницю в клітинку, якщо навчальний матеріал НЕ, зазначеного в номері рядка, логічно пов'язаний з навчальним матеріалом НЕ, зазначеного в номері стовпця. Складання матриці логічних зв'язків зручно вести на основі матриці відносин черговості шляхом виключення одиниць з тих осередків, для яких відсутні логічні, опорні зв'язки між елементами (рис. 1, *а* і *б*).

Процес заповнення матриць доцільно вести, маючи перед очима таблицю НЕ і тексти з навчальним матеріалом по всьому НЕ, якщо вони є. Аналіз змісту навчального матеріалу дозволяє більш об'єктивно виявляти парні відносини черговості і логічні зв'язки між НЕ.

Проаналізуємо як приклад деякі осередки матриць на малюнку. 1, *а* і *б*. Так, одиниця в другій позиції 3-го рядку обох матриць означає, що 3-й НЕ спирається на 2-й НЕ і навчальний матеріал по моделі освоєння повинен викладатися і вивчатися після викладу і вивчення поняття моделі змісту (табл. 1). Навчальний матеріал 3-го НЕ безпосередньо не спирається на поняття цільових показників в 7-у НЕ (0 в осередку 3-7 матриці логічних зв'язків), але, оскільки 7-й НЕ входить в поняття моделі змісту 2-го НЕ, у часовій послідовності вивчення навчального матеріалу 3-й НЕ повинен розглядатися пізніше 7-го (одиниця у відповідній клітинці матриці відносин черговості).

	1	2	3	4	5	6	7	8	9	10	Σ
1	1	1	1	1	1	1	1	1	1	1	10
2		1			1	1	1	1	1	1	7

	1	2	3	4	5	6	7	8	9	10
1		1	1	1						
2					1	1				

3		1	1		1	1	1	1	1	1	8
4		1	1	1	1	1	1	1	1	1	9
5					1		1	1	1	1	5
6					1	1	1	1	1	1	6
7							1				1
8							1	1			2
9							1	1	1		3
10							1	1	1	1	4

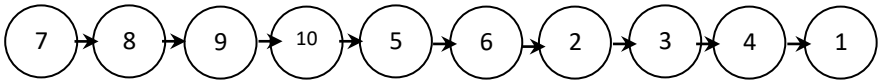
1 2 3 4 5 6 7 8 9 10 кроки навчання

а

3		1			1	1				
4		1	1			1			1	
5										
6					1		1	1	1	1
7										
8							1			
9							1			
10							1			

Порядок вивчення НЕ

б



в

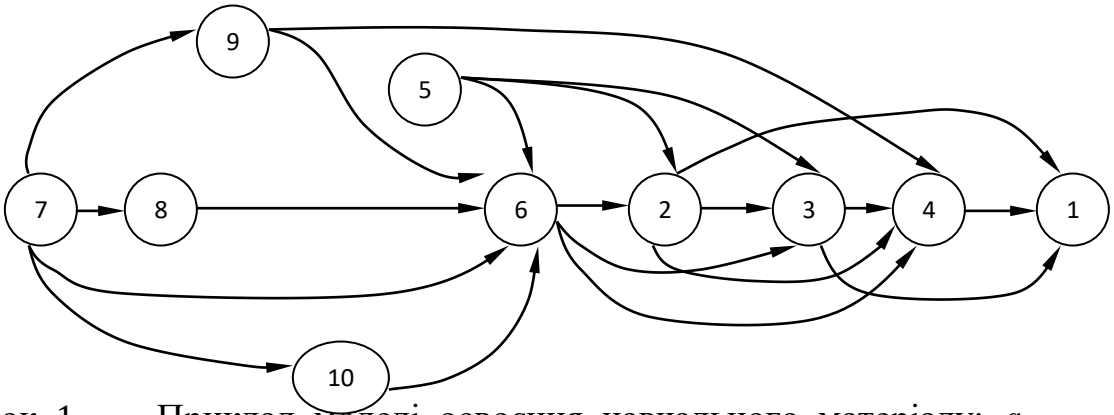


Рисунок 1. Приклад моделі освоєння навчального матеріалу: *а* - матриця відносин черговості НЕ; *б* - матриця логічних зв'язків НЕ; *в* - послідовність вивчення НЕ; *г* - граф логічних зв’язків.

Чи не для всіх НЕ може бути очевидний вибір послідовності: від загального - до приватного або навпаки. Наприклад, поняття проектування навчальних комплексів (НЕ номер 1 в розглянутому прикладі) може викладатися не як узагальнення-резюме в п. 1.7 даного розділу, а як загальне поняття з перерахунком всіх етапів проектування на початку розділу після п. 1.1. В

принципі, з точки зору логіки викладу, байдужа черговість розгляду НЕ під номерами 7, 8, 9. Тому на вигляд матриць відносин черговості і логічних зв'язків, а, отже, в подальшому і на форму представленя навчального матеріалу впливають не тільки об'єктивні, а й суб'єктивні фактори: смаки розробника комплексу, його звички, інтуїтивне уявлення, склад мислення і т.п. Це природна ситуація і, звичайно ж, боятися або соромитися її не слід.

Послідовність вивчення НЕ в покроковій процедурі навчання визначають в процесі формальної обробки матриці відносин черговості, підсумовуючи коефіцієнти кожного рядка матриці. Отримані суми записують в колонці праворуч від матриці (рис. 1, *а*). Величини сум вказують порядкові номери відповідних НЕ в списку послідовності вивчення навчального матеріалу (рис. 1, *в*).

Логічні зв'язки НЕ відображають для наочності у вигляді орієнтованого графа (рис. 1, *г*). Будують граф по матриці логічних зв'язків НЕ, яка є для нього транспонованою матрицею суміжності. Доцільно розташовувати цей граф під списком послідовності НЕ, зберігаючи вказаний в списку порядок освоєння навчального матеріалу.

Ребра графа логічних зв'язків вказують на опорні зв'язки між НЕ. Так, ребра, що зв'язують НЕ номер 2 з НЕ під номерами 5 і 6 (рис. 1, *г*), вказують, що для освоєння поняття моделі змісту навчального матеріалу в тій формі, в якій це поняття вводиться в даному розділі посібника, необхідно мати уявлення про поняття графа змісту і таблиці навчальних елементів.

Модель освоєння навчального матеріалу комплексу визначає послідовність його викладу в навчальному посібнику, варіанти траєкторій його засвоєння в АУК, логічні зв'язки при побудові гіпертексту.

3. ПОРЯДОК ВИКОНАННЯ РОБОТИ

- Ознайомитися з основними теоретичними положеннями;
- Розробити матрицю відносин черговості елементів навчального матеріалу, підготовленого в практичній роботі: «Розробка моделі опанування навчального матеріалу: Розробка структурно-логічної схеми навчальної теми»;
- Відповісти на контрольні питання.

4. Контрольні питання

- Які елементи входять до складу моделі освоєння? Перерахуйте етапи побудови моделі.
- Опишіть алгоритм побудови матриці.

ПРАКТИЧНА РОБОТА № 4
Тема: **Розробка етапів змісту педагогічних програмних засобів**

1. ЗАГАЛЬНІ ВІДОМОСТІ

Мета роботи: Вивчити етапи проектування педагогічних програмних засобів і отримати практичні навички розробки змісту педагогічних програмних засобів.

Оснащення: індивідуальні методичні вказівки для самостійної роботи, науково-методична література, періодичні видання.

Тривалість роботи: 4 години

2. ОСНОВНІ ТЕОРЕТИЧНІ ПОЛОЖЕННЯ

Педагогічний програмний засіб - електронний підручник можна визначити як програмно-інформаційну систему, що складається з програм для ПК, що реалізують сценарії навчальної діяльності, і певним чином підготовлених знань (структурованої інформації і системи вправ для її осмислення та закріплення). Звідси випливають ключові проблеми проектування електронного підручника: підготовка інформаційного опису теоретичного матеріалу (навчальних текстів, ескізів графічних ілюстрацій, сценаріїв демонстраційно-ілюструючих програм і анімацій, відеокліпів і т.п.), створення вправ для активізації процесу засвоєння теорії, розробка сценаріїв (алгоритмів управління) для організації ефективної цілеспрямованої пізнавальної діяльності учнів.

Коментар Для розробки електронного підручника нерідко використовують спеціальні педагогічні інструментальні програмні засоби, звані іноді авторськими системами. Ступінь досконалості тієї чи іншої авторської системи визначається сервісними можливостями по введенню, редагуванню, компонуванню текстової частини навчального матеріалу, наявністю шрифтів для математичної

символіки, використанням графіки, типами вправ (з множинним вибором, з числовими відповідями, з конструйованими відповідями), включенням елементів гіпертексту, мультимедіа і т.п. Однак всі ці «хитрощі» творців авторських систем надають розробникам електронного підручника лише потенційні можливості для реалізації їх дидактичних ідей. Проектування електронного підручника ведеться за «столом» і є свого роду мистецтвом, внаслідок чого електронні підручники, підготовлені різними авторами навіть в одному авторському середовищі, можуть істотно відрізнятися за їх дидактичної ефективності .

Елементи управління в сценаріях навчальних програм

Відповідно до постулатів загальної теорії управління в будь-яких циклічних замкнутих системах управління, в тому числі і в педагогічних, повинні бути реалізовані наступні функції:

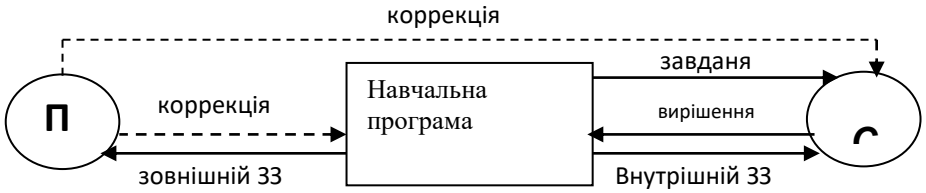
1. формування цілей управління;
2. встановлення вихідного стану об'єкта управління;
3. визначення програми дій, яка передбачає основні перехідні стани об'єкта управління;
4. систематичний збір інформації зворотного зв'язку;
5. переробка інформації зворотного зв'язку з метою вироблення та реалізації коригувальних впливів.

Зупинимося детальніше на особливостях поняття зворотного зв'язку, властивих педагогічним системам. Зворотний зв'язок (ЗЗ) у тріаді "Педагог - Навчальна програма - Той, якого навчають" можна розділити на два види: зовнішня і внутрішня ЗЗ (рис. 1).

Внутрішній ЗЗ - це інформація, яка надходить від навчальної програми до учня у відповідь на його дії при виконанні вправ. Вона призначена для самокорекції учнем своєї навчальної діяльності.

Поняття внутрішнього ЗЗ має виключно важливе значення для автоматизації процесу навчання. **Внутрішній ЗЗ** дає можливість учневі зробити усвідомлений висновок про успішність або

помилковость навчальної діяльності. Він спонукає учня до рефлексії, є стимулом до подальших дій, допомагає оцінити і скорегувати результати навчальної діяльності. **Розрізняють консультативний і результативний внутрішній ЗЗ**. Консультація може бути різною: допомога, роз'яснення, підказка, наштовхування і т.п. Результативний ЗЗ також може бути різним: від «вірно-невірно» до демонстрації правильного результату або способу дії.



П - педагог

С - студент

ЗЗ - зворотній зв'язок

Рисунок 1. Схема взаємодії в тріаді «Педагог-АУК-Якого навчають».

Інформація **зовнішнього ЗЗ** в розглянутій тріаді (див. Рис. 1) надходить до педагога і використовується ним для корекції діяльності учня і навчальної програми.

Склад типового фрагмента електронного підручника

На початковому етапі проектування електронного підручника декомпозують його на окремі фрагменти. Кожен фрагмент відповідає одному навчальному елементу. Розташування фрагментів і їх логічні зв'язки відповідають теорії засвоєння навчального матеріалу. Кілька додаткових фрагментів на початку електронного підручника повинні бути присвячені створенню мотивації і загального орієнтування в

навчальному матеріалі. В кінці електронного підручника, враховуючи дробовий характер покрокової процедури програмованого навчання, повинні бути узагальнюючі фрагменти.

До складу типового фрагмента електронного підручника можуть входити його назва, інформаційний блок, блоки вправ і коментарів до них.

Інформаційний блок (ІБ) містить теоретичний матеріал, викладений на заданому для розглянутого навчального елемента рівні уявлення. Що розміщувати в ІБ? Тут можуть бути різні підходи, що відрізняються обсягом інформації.

1. ІБ містить тільки найменування навчального елемента, за яким далі йдуть вправи. Передбачається, що інформація з даного навчального елементу викладена в посібнику, підручнику.

2. ІБ містить коротку інформацію (нагадування) за навчальним елементом, викладеним в повному обсязі на паперовому носії.

3. ІБ містить всю інформацію по даному навчальному елементу, замінюючи або дублюючи паперовий носій.

Вибір того чи іншого підходу визначається конкретними обставинами: наявністю доступного навчального посібника, змістом навчального матеріалу, смаками викладача, можливостями інструментальних засобів для підготовки електронного підручника, об'ємом навчального матеріалу і т.п.

Інформаційний блок складається з сторінок. Сторінками можуть бути текстові та графічні екрани, анімаційні ролики, відеокліпи, демонстраційні розрахункові програми і т.п. Зручно, коли інформаційний блок містить 3-5 сторінок. Тоді їх можна «перегортати» вперед і назад, осмислюючи представлену на них інформацію.

При підготовці інформаційних блоків доцільно планувати застосування технологій гіпертексту, мульти- і гіпермедіа. Інструментальні засоби гіпертексту дозволяють

розробнику електронного підручника позначати, підсвічуючи яким-небудь певним кольором, окремі ключові слова або поєднання і пов'язувати їх з фрагментами тексту в інших ІБ, де дається детальний опис цих понять. Якщо учневі незрозумілий позначений термін в тексті, то досить підвести до нього курсор, натиснути певну клавішу і отримати на екрані більш детальну інформацію по ньому, а потім повернутися до початкового тексту. Таким чином, здійснюється довільна навігація по всьому тексту, причому кожен учень вибирає підходящий для нього шлях самостійно. Зауважимо, що при підготовці гіпертексту необхідно спиратися на модель освоєння навчального матеріалу - матрицю і граф логічних зв'язків між навчальними елементами (практичні роботи: «Розробка моделі освоєння навчального матеріалу: Розробка структурно-логічної схеми навчальної теми» і «Розробка моделі освоєння навчального матеріалу: Розробка матриці відносин черговості навчальних елементів »).

Технологія мультимедіа дозволяє оживити текст, оснащувати його графічними ілюстраціями (статичними і динамічними), фотографіями, відеокліпами, фрагментами аудіоінформації. Поєднання технологій гіпертексту і мультимедіа отримало назву гіпермедіа. При цьому з'являється можливість пов'язувати з позначеними термінами не тільки елементи тексту, а й графічні ілюстрації, анімаційні ролики, фрагменти оцифрованої аудіо- та відеоінформації.

Застосування таких технологій істотно активізує навчальну інформацію, робить її в порівнянні з поданням на паперовому носії більш наочною для сприйняття і зручною для засвоєння.

Блок вправ типового фрагмента АУК повинен містити вправи по кожному рівню засвоєння. Для кожного рівня необхідно не менше 2-5 вправ, щоб забезпечити засвоєння.

Розрізняють тренуючі і контрольні вправи. Перші використовують для осмислення і закріплення інформації, другі - для

діагностики та вимірювання $K_{\text{п}}$, $K_{\text{і}}$, на початку і в кінці роботи учня з електронним підручником.Тренують вправи нерозривно пов'язані з коментарями, які є інформацією зворотного зв'язку. Вправи, що не супроводжуються внутрішнім ЗЗ, є контрольними.

Підготовка вправ - це найбільш трудомістка справа в створенні електронного підручника, яка потребує високої педагогічної майстерності від викладача-розробника. Для кожного електронного підручника необхідно придумати не тільки відповідні завдання для його засвоєння, а й певним чином розташувати і ранжувати їх, вибрати форму вправ (з вибірковими, числовими, конструйованими відповідями), підготувати зразки відповідей і передбачити типові помилки.

Блок коментарів може містити різні види інформації внутрішнього ЗЗ для реакцій на дії учнів при виконанні вправ - від найпростіших (вірно, невірно, неточно) до докладних роз'яснень типових помилок. Нерідко в коментарях використовують відповідні сторінки або набір сторінок інформаційного блоку.

3. ПОРЯДОК ВИКОНАННЯ РОБОТИ

- Ознайомитися з основними теоретичними положеннями;
- Підготувати навчальний матеріал для фрагмента педагогічного програмного засобу, використовуючи підготовлений матеріал практичних робіт: «Розробка моделі освоєння навчального матеріалу: Розробка структурно-логічної схеми навчальної теми» і «Розробка моделі освоєння навчального матеріалу: Розробка матриці відносин черговості навчальних елементів» і «Розробка алгоритмів засвоєння знань »:
  - схема сценарію;
  - зміст інформаційного блоку (1-2 сторінки тексту, 1-2 ілюстрації, тощо);

– блок вправ (5 тестів);

– блок коментарів (якщо необхідно);

- Відповісти на контрольні питання.

4. Контрольні питання

1. Визначте сутність поняття «електронний підручник».
2. Перелічіть існуючі проблеми проектування електронного підручника.
3. Визначте поняття "внутрішнього зворотного зв'язку" в електронному підручнику.
4. Визначте поняття "зовнішнього зворотного зв'язку" в електронному підручнику.
5. Опишіть схему взаємодії в тріаді "Педагог - Електронний підручник - Той, якого навчають".
6. У чому сутність інформаційного блоку електронного підручника?
7. Для чого потрібен блок вправ електронного підручника?
8. У чому сутність блоку коментарів електронного підручника?
9. Перерахуйте етапи проектування електронного підручника.

ПРАКТИЧНА РОБОТА № 5

Тема: **Створення тестів при проектуванні педагогічних програмних засобів**

1. ЗАГАЛЬНІ ВІДОМОСТІ

Мета роботи: Вивчити види тестів і їх особливості, отримати практичні навички створення тестів.

Оснащення: індивідуальні методичні вказівки для самостійної роботи, науково-методична література, періодична література.

Тривалість роботи: 4 години

2. ОСНОВНІ ТЕОРЕТИЧНІ ПОЛОЖЕННЯ

Тести

При проектуванні електронного підручника значна частина роботи припадає на створення тестів. Вони використовуються в тренувальних і контрольних вправах. Тренує вправу - це тест, обов'язково супроводжується внутрішнім зворотним зв'язком (ЗЗ). Контрольна вправа - це теж тест, але вже не супроводжуваний внутрішнім ЗЗ. Розрізняють тести для оцінки якостей особистості, розумових здібностей, спеціальних здібностей, тести досягнень. Будемо розглядати тільки тести досягнень.

Структура тесту: Тест = завдання + еталон .

Якщо в тесті відсутній еталон, то оцінка правильності тесту схильна до ілюзій окоміру і суб'єктивним судженням. А без оцінки правильності виконання тесту неможливо провести діагностику і вимір при контролі, сформувати внутрішій ЗЗ при навчанні.

1. Відповідність джерел інформації, якими користуються учні (вимога відповідності змісту й обсягу отриманої учнями інформації).

2. Завдання повинно вимагати від учня вирішення тільки одного питання (вимога простоти). Виконання цієї вимоги перешкоджає непотрібного ускладнення процедури контролю.

3. Формулювання питання тесту повинно вичерпно роз’яснювати поставленого перед учнями завдання, причому мова і терміни, способи і індексація позначень, графічні зображення та ілюстрації завдання і відповідей повинні бути безумовно і однаково (однозначно) зрозумілі всіма учнями (вимога однозначності завдання).

4. Подробиця питання (завдання) і лаконічність відповідей тестів.
5. Граматична і логічна відповідність відповідей питання (завдання).

Крім перерахованих вимог виділяють вимоги:

- до валідності;
- до визначеності;
- до простоти;
- до однозначності;
- до надійності.

Валідність тесту - це адекватність. Розрізняють змістовну і функціональну валідність: перша - це відповідність тесту змісту контрольованого навчального матеріалу, друга - відповідність тесту оцінюваному рівню діяльності.

Виконання вимоги *визначеності* (загальнодоступності) тесту необхідно не тільки для розуміння кожним учнем того, що він повинен виконати, але і для виключення правильних відповідей, що відрізняються від еталону.

Вимога *простоти* тесту означає, що тест повинен мати одне завдання одного рівня, тобто не повинен бути комплексним і складатися з кількох завдань різного рівня засвоєння.

Однозначність визначають як однаковість оцінки якості виконання тесту різними експертами. Для виконання цієї вимоги тест повинен мати еталон. Для вимірювання ступеня правильності використовують коефіцієнт:

$$K_{\alpha} = P_1 / P_2$$

де P_1 - кількість правильно виконаних істотних операцій в тесті або батареї тестів;

P_2 - загальна кількість істотних операцій в тесті або батареї тестів.

Істотними вважають ті операції в тесті, які виконуються на підприємстві, що перевіряється рівень засвоєння. Операції, що належать до нижчого рівня засвоєння, в число істотних не входять. При $K_{\alpha} \geq 0.7$ вважають, що діяльність на даному рівні засвоєна.

Поняття *надійності* тестування визначають як ймовірність правильного виміру величини K_∞. Кількісний показник надійності $r \in [0, 1]$. Вимога надійності полягає в забезпеченні стійкості результатів багаторазового тестування одного і того ж випробуваного. Надійність тесту або батареї тестів зростає зі збільшенням кількості суттєвих операцій P. Так, для $\alpha = 1$ при $P = 20$ ймовірність правильного виміру (надійність тесту) $r = 0.5$; при $P = 80$ $r = 0.9$; $P = 100$ $r = 0.99$.

Тести першого рівня . Нагадаємо, що діяльність першого рівня (впізнання, $\alpha = 1$) - це репродуктивна діяльність за допомогою (з зовнішньою опорою). У наведених нижче прикладах (табл. 1) зовнішньої опорою є представлені явно самі об'єкти, по яким задаються питання.

Таблиця 1.
Приклади тестів першого рівня засвоєння.

№ п / п	завдання	еталони	Р
	розпізнавання		
1.	Чи відноситься структура тесту: Тест = завдання + еталон, до тестів досягнення	да	1
	розрізнення		
2.	Однаковість оцінки якості виконання тесту різними експертами відноситься до 1. валідності тесту 2. визначеності тесту 3. однозначності тесту 4. простоти тесту 5. надійності тесту	однозначність	5
	Класифікація		
3.	Вкажіть змістовну і функціональну валідність тесту 1. відповідність тесту однаковості якості виконання різними експертами 2. відповідність тесту оцінюваному рівню діяльності 3. відповідність тесту змісту контрольованого навчального	Змістовна - 3 Функціональна - 2	4

	матеріалу 4. відповідність тесту на розуміння кожним учнем того, що він повинен зробити		

Тести другого рівня. Нагадаємо, що діяльність другого рівня (відтворення, $\alpha = 2$) - це відтворення раніше засвоєної інформації по пам'яті від буквальної копії до застосування в типових ситуаціях (табл. 2).

Таблиця 2.
Приклади тестів другого рівня.

№ п / п	завдання	еталони	Р
тести підстановки			
1.	Поняття ... тестування визначають як ймовірність правильного виміру величини	надійність	1
конструктивні тести			
2.	Яке поняття тестування визначають як ймовірність правильного виміру величини	надійність	1
3.	Дайте визначення ...	Ключові слова, символи, порядок їх розташування	За кількістю ключових слів
4.	Напишіть формулу ...		
5.	Перерахуйте ознаки (властивості) ...		
Типові завдання			
6.	Визначте ступінь правильності тесту якщо $P_1 = 17$; $P_8 = 20$	1. $K_{\alpha} = P_1 / P_8$ 2. = 17/20 = 0,85	1
7.	Визначте величину струму в мережі з напругою U = 150 В і опором R = 50 Ом	1. I = U / R 2. I = 150/50 = 3 А	1

Тести третього рівня . При досягненні третього рівня засвоєння матеріалу ($\alpha = 3$) учень здатний самостійно відтворювати і перетворювати засвоєну інформацію для обговорення відомих фактів і продукування про них

суб’єктивно нової інформації (нової для нього), а також для застосування її в різноманітних нетипових, реальних ситуаціях. Строго кажучи, ряд нетипових завдань може бути в процесі навчання переведений в розряд типових.Однак можуть бути навчальні завдання, які за своєю природою завжди залишаються нетиповими, скільки б ми не вправлялися в їх вирішенні, наприклад, формулювання проектної задачі в термінах математичного програмування.Учень вивчив об'єкт проектування, володіє математичними методами оптимізації. Якщо об'єкт досить складний, то його проектування розпадається на ряд різноманітних приватних проектних завдань. Розглянути всі можливі ситуації в ході навчання, тобто перевести їх в розряд типових, часто неможливо з огляду на їх різноманіття. Тому декомпозиція загальної задачі на приватні, приведення приватних завдань до стандартного вигляду, використовуваному в оптимізації, є практично завжди нетипової ситуацією ($\alpha = 3$).

Необхідно розрізняти тип і форму тесту. Тип тесту будемо пов'язувати з рівнем засвоєння: впізнання, розрізнення, класифікація - типи тестів першого рівня; тести підстановки, конструктивні тести, типові завдання - типи тестів другого рівня; нетипові завдання - тести третього рівня. Тип тесту визначається характером внутрішньої розумової діяльності, яку повинен виконати учень при вирішенні тесту.

Форма тесту визначає його зовнішнє уявлення. Сучасні інструментальні середовища для створення електронних підручників дозволяють будувати тести з вибірковими, числовими, конструюються відповідями. На практиці найчастіше застосовують тести з вибірковими відповідями. Вони простіше в підготовці (не потрібно створювати безліч еталонів правильних відповідей, забезпечити повноту якого вкрай важко) і, що найголовніше, простіше у використанні. У тестах з вибірковими відповідями учні основні зусилля витрачають на виконання завдання, а не на набір відповідей.

Нерідко викладачі пов'язують тести з вибірковими відповідями тільки з першим рівнем засвоєння (впізнання, розрізнення, класифікація). На жаль, це досить широко поширене дидактичний оману є результатом поверхневого судження. Для визначення типу тесту важлива не його форма, а вид розумової діяльності, яку виконує учень при вирішенні тесту. Якщо учень аналізує подані варіанти відповідей, виконуючи операції впізнання, розрізнення або класифікації, то це тест першого рівня. Якщо ж учень спочатку конструює

відповідь, згадуючи раннє засвоєну інформацію або застосовуючи її для вирішення типової або нетипового завдання, і лише після цього вибирає відповідь з представлених варіантів, то це тест відповідно другого або третього рівня засвоєння. Причому, якщо число варіантів відповідей більше трьох (5-9), то ймовірність вгадування невелика.

3. ПОРЯДОК ВИКОНАННЯ РОБОТИ

- Ознайомитися з основними теоретичними положеннями;
- Скласти по 2 приклади тестів першого і другого рівнів засвоєння навчального матеріалу;
- Розробити тест, що відповідає вимогам валідності, визначеності, однозначності і надійності за матеріалом підготовленим в процесі виконання практичних робіт;
- Відповісти на контрольні питання.

4. Контрольні питання

1. Визначте сутність поняття «тест».
2. Перелічіть існуючі види тестів.
3. Розкрийте сутність поняття «валідність тесту».
4. Що таке простота тесту?
5. Розкрийте сутність поняття «однозначність тесту».
6. У чому сутність надійності тесту?
7. У чому сутність тестів першого рівня?
8. У чому сутність тестів другого рівня?
9. У чому проявляється відмінність тестів першого рівня від другого?
10. Що таке тип тесту?

11. Розкрийте сутність поняття «форма тесту»?

ПРАКТИЧНА РОБОТА № 6

Тема: **Правила створення HTML документів: Структура HTML документа. Функціональні розділи. Кольори елементів.**

1. ЗАГАЛЬНІ ВІДОМОСТІ

Мета роботи: Вивчити правила створення HTML документів, ознайомитися зі структурою HTML документа його функціональними розділами, отримати практичні навички і досвід їх створення.

Оснащення: індивідуальні методичні вказівки для самостійної роботи, науково-методична література, періодична преса, матеріал, підготовлений в процесі виконання попередніх практичних робіт, електронний підручник «Основи проектування педагогічних програмних засобів», електронний підручник віртуального факультативу.

2. ОСНОВНІ ТЕОРЕТИЧНІ ПОЛОЖЕННЯ

Термін HTML (Hyper Text Markup Language) означає «мова маркування гіпертекстів». Першу версію HTML розробив співробітник Європейської лабораторії фізики елементарних частинок Тім Бернерс-Лі.

HTML-документ є текстовим файлом з розширенням htm (Unix-системи можуть містити файли з розширенням html). HTML документи можуть бути створені і відредаговані в будь-якому текстовому редакторі.

Мова HTML представляє з себе безліч команд, укладених між знаками «<» і «>», наприклад, <html> - це називається міткою (англійською - tag, читається «тег»). Більшість HTML-міток - парні, тобто на кожну мітку що відкривається виду <tag> є закриває мітка вигляду </ tag> з тим же ім'ям, але з додаванням знака слеш - «/». Багато мітки, крім імені, можуть містити атрибути - елементи, що дають додаткову інформацію про те, як browsers повинен обробити поточну мітку. У найпростіших HTML документах таких атрибутів немає.

Таким чином, мітка або тег HTML складається з наступних один за одним в певному порядку елементів:

- лівої кутової дужки "<" (такого ж, як символ «менш ніж»);

- слеша «/», який означає, що тег є кінцевим, що закриває деяку структуру, тобто в цьому контексті ви можете читати символ «/», як кінець.

- імені тега, наприклад TITLE або PRE

- необов'язкових, якщо навіть тег може мати їх, атрибути.

Тег може бути без атрибутів або супроводжуватися одним або декількома атрибутами, наприклад: ALIGN = CENTER

- правої кутової дужки «>» (такий же, як символу «більше ніж»), які відкривають <...>, і закривають </ ...> дії які відповідають даним командам. Це можуть бути команди створення будь-яких елементів сторінки, як, наприклад вбудована таблиця або зображення. Текст, укладений між тегами, при виведенні на екран за допомогою browsers підпорядковується правилам притаманним для даних тегів (як, наприклад розмір або колір). Деякі з тегів не вимагають закриття, і припиняють свою дію з появою аналогічних команд які їх відкривають. (Наприклад, рядок в списку переривається там, де дана команда створення наступного рядка списку.) Однак, «зайвий» тег який закриває не зашкодить. Browsers всі незрозумілі для нього команди пропускає повз, що вельми важливо для недосвідчених творців HTML документів. Тому, краще закривати теги - це полегшить читання документів.

Теги можуть бути вкладені один в одного, для надання тексту відразу декількох властивостей. В цьому випадку, *порядок закриття тегів повинен бути строго протилежний порядку їх відкривання.* Наприклад, якщо ви спочатку виділили текст жирним шрифтом, а потім ще й курсивом, то закривати слід спочатку внутрішній тег (курсив) і тільки потім жирний шрифт.

Наприклад:

<B> Вищеописаний <I> текст </ I> </ B> <I> буде мати такий вигляд. </ I>

При перегляді в browsers матимемо:

Попередній *текст* *буде мати такий вигляд.*

HTML не є чутливою до клавіатури, тобто якими буквами набрані команди мови: <html> сприймається аналогічно <HTML> або <hTmL>. Виняток становлять спеціальні символи.

Обов'язкові структурні елементи HTML-документа

<Html> ... </ html>

Тег <html> повинен відкривати HTML-документ. Аналогічно , тег </ html> повинен завершувати HTML-документ.

«Голова» документа: <head> ... </ head>

Ця пара тегів вказує на початок і кінець заголовка документа. Крім найменування документа (див. Опис тега <title> нижче), в цей розділ може включатися безліч службової інформації.

<Title> ... </ title>

Все, що знаходиться між мітками <title> і </ title> , тлумачиться browsers як назва документа. *Netscape Navigator* , наприклад, показує назву поточного документа в заголовку вікна і друкує його в лівому верхньому куті кожної сторінки при виведенні на принтер.

Назва - необов'язкова частина в документі, але крім відображення в заголовку browsers імені сторінки, може включати необмежену кількість дуже корисних META-інструкцій. Зазвичай вони розташовуються між двома першими мітками <head> і <title>. META-інструкція це стандартний опис теми документа (для пошукових систем) або ж пряма вказівка для browsers.

Приклад:

<META HTTP-EQUIV = "Content-Type" CONTENT = "text / html"> - інструкція дає вказівку браузеру інтерпретувати документ як HTML-текст.

<META HTTP-EQUI V = "Refresh" CONTENT = "17; URL = http: //www.abcd.ru"> (або URL = music.mid ">) - Така інструкція через 17 секунд почне завантаження вузла з указаним URL (або почне відтворення файлу music.mid в звуковому форматі, якщо той підтримується browsers.)

Простір між мітками що закриваються </ title> і </ head> часто використовується для зберігання операторів JavaScript і VBScript використовують глобальні змінні і функції, а також при впровадженні Каскадних таблиць стилів (команда, вставлена між тегами </ title> і < / head>. У ній описані елементи сторінки, які мають однаковий вигляд, тобто до яких будуть застосовані певні стильові рішення) або для оголошення, наприклад, нестандартного розміру шрифту сторінки за допомогою тега <basefont size = "5" >

Примітка: Рекомендується назва не довше 64 символів.

«Тіло» документа: <body> ... </ body>

Ця пара тегів вказує на початок і кінець **тіла** HTML-документа, яке власне, і визначає зміст документа.

Основний текст сторінки знаходиться після необов'язкового заголовка, між так же необов'язковими мітками: **<BODY>** ... тіло сторінки ... **</ BODY>** . (Сучасні браузери самі розпізнають, де кінчається назва і починається тіло документа ...). Якщо елемент BODY не містить атрибути, які розміщені всередині мітки, використання його не дає явного ефекту в безпосередньому відображенні документа. Даний тег можна використовувати для завдання кольору або фонового малюнка для сторінки.

Атрибути використовуються в тегах <body>

BGCOLOR - визначає колір фону документа

TEXT - застосовується безпосередньо для визначення кольору тексту документа

LINK -використовується для кольору невідвіданого гіпертекстового зв'язку, тобто визначає колір виділеного елементу тексту, при натисканні на який відбувається перехід по гіпертекстовому посиланню

VLINK - застосовується для кольору відвіданого гіпертекстового зв'язку, тобто визначає колір посилання на документ, який вже був переглянутий раніше

ALINK - даний атрибут використовується для кольору активного гіпертекстового звязку, тобто використовується для виділення тексту при натисканні

BACKGROUND - використовується для URL фонового образу

BOTTOMMARGIN - встановлює кордон нижнього поля документа в пікселах

TOPMARGIN - встановлюється кордон верхнього поля в пікселах

BGPROPERTIES - якщо встановлено значення FIXED, фонове зображення не прокручується

LEFTMARGIN - встановлює кордон лівого поля документа в пікселах

RIGHTMARGIN - встановлює кордон правого поля документа в пікселах

SCROLL - Yes / No встановлює наявність або відсутність полос прокрутки вікна браузера

Наприклад:

<BODY BGCOLOR = GREEN> - зелений колір сторінки

Кольори елементів

Фоновий колір і колір тексту для всієї сторінки призначаються параметрами тега BODY:

<body bgcolor = "колір" text = "колір"> *... тіло сторінки* **... </ body>**

Колір шрифту для окремих ділянок тексту (тег також може керувати розмірами шрифту):

<font color = "колір"> *... фрагмент тексту ...* **</ font>**

Колір фону таблиці: (аналогічно і для окремих осередків, тільки table змінюється на tr, td або th)

<table bgcolor = "колір"> *... тіло таблиці (осередки) ...* **</ table>**

Колір базового шрифту для всієї сторінки також можна визначити тегом:

<basefont color = "колір"> - розташованим в «голові» сторінки (частіше застосовується для установки розмірів шрифту)

Сучасні браузери (від IE 4.0.) Розпізнають 140 визначених кольорів.

Задання кольору можливо як в іменному, так і в цифровому вираженні, тобто у вигляді: <BODY BGCOLOR = tomato »або« font color = FF34C6>

Таблиця з назвами іменних кольорів представлена на рисунку 1

antiquewhite	aqua	aquamarine
black	chartreuse	darkblue

blue	blueviolet	brown
coral	cornflowerblue	cornsilk
darkgoldenrod	darkgray	cyan
beige	darkgreen	darkkhaki
darkred	darksalmon	darkseagreen
darkturquoise	darkviolet	deeppink
floralwhite	forestgreen	fuchsia
goldenrod	indigo	lightblue
green	greenyellow	honeydew
limegreen	deepskyblue	dimgray
khaki	lavender	lavenderblush
lightcyan	lightgoldenrodyellow	lightgreen
lightskyblue	lightslategray	lightsteelblue
magenta	maroon	mediumaquamarine
mediumseagreen	mediumslateblue	mediumspringgreen
mediumorchid	lime	lightpink
maccasin	novajowhite	navy
mintcream	livedrab	palevioletred
orangered	orchid	palegoldenrod
peachpuff	peru	pink
rosybrown	royalblue	saddlebrown
powderblue	paleturquoise	snow
sienna	silver	skyblue
steelblue	tan	teal
turquoise	slategray	tomato
wheat	white	whitesmoke

Малюнок 1. Таблиця кольорів.

У цифровому вираженні кольоровість шифрується інтенсивністю трьох основних кольорів: червоного, зеленого і синього (RGB) в шістнадцятирічній системі насиченості заданого кольору одним із трьох основних кольорів в діапазоні від нуля (00) до 255 (FF). 00- нульова інтенсивність, FF- максимальна насиченість. Відповідно: 000000- чорний колір, FF0000- червоний, 00FF00 - зелений, 0000FF - синій, FFFFFF- білий максимальної інтенсивності.Виходячи з цього правила, кольори з одинаковими значеннями всіх трьох параметрів (Red = Green = Blue) будуть сірим, але різної інтенсивності. Розберемо декілька прикладів.

bgcolor = # FFFFFF

Колір фону . Насиченість червоним, зеленим і синім однакова - FF (це шістнадцятирічне уявлення числа 255). Результат - білий колір.

t ext = # 000000

Колір тексту. Насиченість червоним, зеленим і синім однакова - 00 (нуль). Результат - чорний колір.

link = # FF0000

Колір гіпертекстового посилання. Насиченість червоним - FF (255), зеленим і синім - 00 (нуль). Результат - червоний колір.

Приклад кольорів в RGB представлений на малюнку 2 Верхня частина, колонками по три, була виставлена як приклад правильного підбору кольорів, в керівництві Артемія Лебедева.

0	ffc200	003562	0069a3
2	ff0000	006d93	000000
d	000000	e8aa00	ffffff
Приклади кольорів в RGB			
d	9595c6	9999ff	009933
9	968187	cc0033	007299
c	ae88b8	960018	e96b9e
f	c699bd	999966	a9 834f
5	ff99cc	33cc66	6699cc

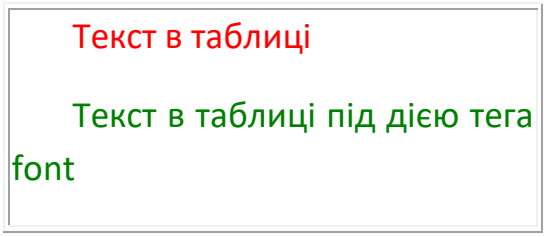
Малюнок 2. Приклад кольорів в RGB.

Розглянемо ієрархію колірних тегів:

<Html> <head> <basefont color = yellow> </ head>
<Body bgcolor = white text = red>
< Font color = blue >
Текст під дією тега font
<t able border> <tr> <td>
Текст в таблиці < br >
< Font color = green > Текст в таблиці під дією тега font </ font >
</ Table >
</ Font >
</ Body > </ html >

Наочно даний код буде виглядати, так як показано на малюнку 3

Текст під дією тега font



Малюнок 3. Вид ієрархії колірних тегів в HTML –документі.

Жовтий колір, заданий тегом basefont перепризначувався параметром text тега body на червоний (тег <basefont> проте залишається актуальним через можливості керувати розміром шрифта, що неможливо для тега BODY.). Будь-який текст, введений зараз без додаткових колірних тегів, буде відображений червоним. Але в запропонованому прикладі такий тег є: - <font color = blue> . Він забарвлює верхній рядок прикладу в блакитний колір. В межі тега потрапляє і таблиця, але на текст який знаходиться всередині таблиці дія цього тегу не розповсюджується! Таким чином, текст таблиці буде виведений кольором, зазначеним в параметрі text (червоним). Якщо ж ми застосуємо для тексту всередині таблиці свій власний тег <font color = green> то він буде справно працювати.

Примітка: текст всередині таблиці слід наповнювати власними тегами <FONT> (це стосується і розмірів).

Крім того, тег <BODY> може містити атрибут background = "[ім'я файлу]" , який задає зображення, що служить фоном для тексту та інших зображень. Як і будь-яке інше зображення, фон повинен бути представлений в форматі GIF (файл з розширенням * .gif) або JPEG (файл з розширенням * .jpg або * .jpeg). Браузери заповнюють множинними копіями зображення-фону весь простір вікна, в якому відкрито документ, подібно до того, як при будівництві великий простір стін покривають маленькими (і однаковими) плитками. Важливо відзначити, що колір фону і зображення-фон ніяк не відображаються на папері, якщо розпечатати HTML-документ. З цього витікає важливий практичний наслідок: **намагайтеся не використовувати текст білого кольору** .

Отже, **<body background = Image.gif>** - команда для використання фонового малюнка, припустимо з імя’м Image.gif

Якщо сторінка займає місця більше ніж вікно браузера, то, здвигаючи смугу прокрутки, буде здвигатися так само і фоновий малюнок. Для того щоб він не «відповзав» слід використовувати наступний синтаксис:

<body bgcolor = "red" background = "fon.jpg" bgprope r ties = "fixed">

В даному прикладі ще до завантаження фонового зображення (fon.jpg) сторінка прийме червоний колір (підбирається в тон фонової картинки, адже зображення може мати великий час завантаження, а може і зовсім загубитися

при поганому зв'язку. Досить важко буде прочитати блідо жовтий текст на білому тлі.), при пропусканню фоновий малюнок буде стояти на місці (bgproperties = fixed) а текст з картинками буде повзти по нім, як титри в кінофільмі.

Примітка : подвійні лапки, так само необов'язкові для сучасних браузерів (а в деяких випадках навіть шкідливі у зв'язку з марним збільшенням розміру сторінки). Їх використання необхідно, в тому випадку, коли значення параметрів складається з декількох слів розділених пробілами або є певним виразом як **style.**

Приклад текстового і числового значення кольору представлений в таблиці 1.

Таблиця 1. Співвідношення текстового і числового значень кольорів.

колір	**текстове значення**	**числове значення**
1	2	3
чорний	Black	# 000000
Темно синій	Navy	# 000080
синій	Blue	# 0000FF
зелений	Green	# 008000
Синьо-зелений	Teal	# 008080
лимонний	Lime	# 00FF00
морської хвилі	Aqua	# 00FFFF
Темно-бордовий	Maroon	# 800000
фіолетовий	Purple	# 800080
оливковий	Olive	# 808000
сірий	Gray	# 808080
Срібний	Silver	# C0C0C0
червоний	Red	# FF0000
фуксин	Fuchsia	# FF00FF
жовтий	Yellow	# FFFF00
білий	White	#FFFFFF

Приклад WEB-сторінки

Виходячи з вищесказаного, приклад мінімально допустимого коду для сторінки яка правильно працює в мережі виглядає наступним чином.

```
<Html>
<Head>
<Meta http-equiv = Content-Type content = text / html>
< Title > Приклад </ title >
</ Head >
```

< Body > Місце для основного коду сторінки </ body >
</ Html >

Заголовки

Для виділення заголовків в тексті існує тег <розмір> *... заголовок ...* </ розмір> (від head- голова). Де розмір заголовка змінюється від 1 до 6 в порядку убування. Заголовок першого рівня - найбільший, шостого рівня, природньо - найдрібніший. У заголовках закриває тег обов'язковий.

< H 1> ... </ H 1> - < H 6> ... </ H6>

Наведений тег виділяє свій заголовок відступом рядків. До нього можна застосовувати атрибут align - вирівнювання:

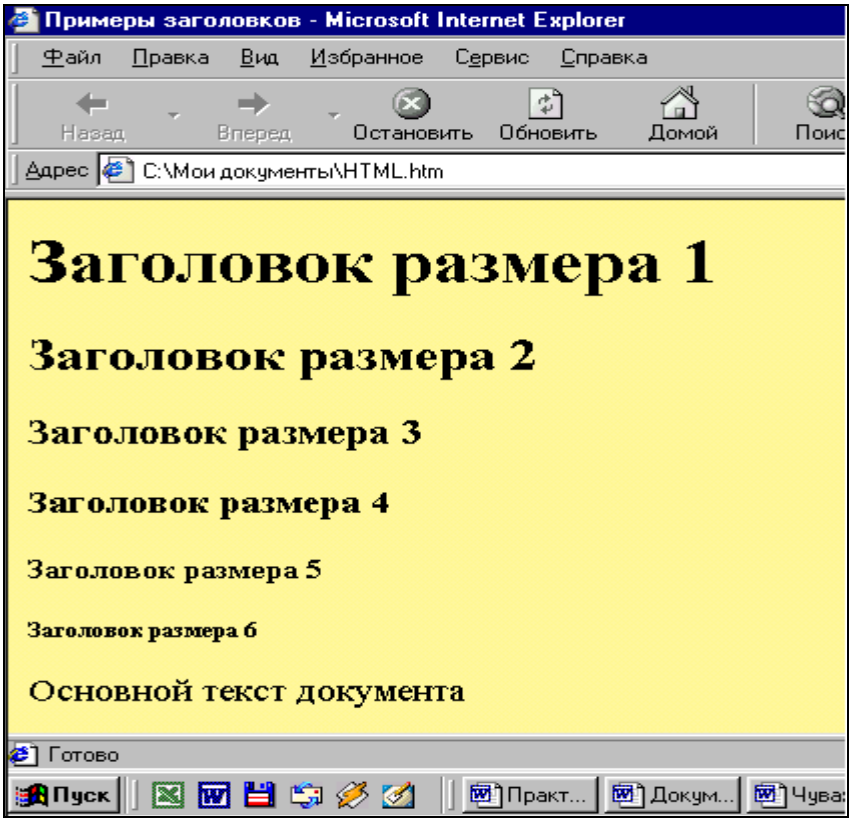
<h5 align = "right"> Заголовок п'ятого рівня **</ h5>**

При цьому заголовок документа буде вирівняний по правому краю

Приклад HTML-команди:

```
< HTML >
< HEAD >
<TITLE> Приклади заголовків </ TITLE>
</ HEAD>
<BODY BGCOLOR = fff99d>
<H1> Тема розміру 1 </ H1>
<H2> Тема розміру 2 </ H2>
<H3> Заголовок розміру 3 </ H3>
<H4> Заголовок розміру 4 </ H4>
<H5> Заголовок розміру 5 </ H5>
<H6> Тема розміру 6 </ H6>
Основний текст документа
</ BODY>
</ HTML>
```

Наочний вид прикладу представлений на малюнку 4.



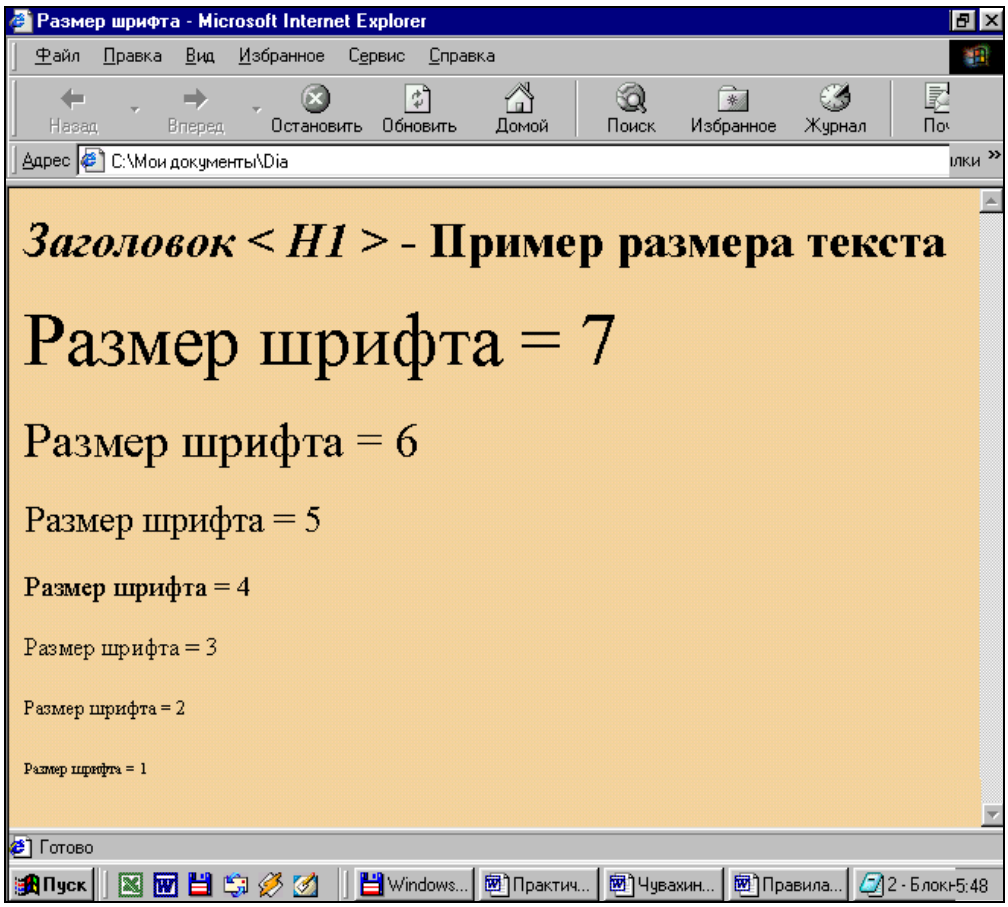
Малюнок 4. Наочний вид заголовків HTML – сторінки.

Розмір шрифту

Розмір шрифту в іншому тексті задається за допомогою команд <font size = "n"> ... </ font> Теги можуть бути вкладені одна в одну в незліченній кількості. Однак при цьому необхідно їх правильно закривати! Значення для ***n*** можуть лежати в межах від 1 до 7 по зростанню розміру шрифту (протилежно розмірам заголовків). Наприклад, наведений нижче код дає на екрані наступне (малюнок 5).

```
< Html >
< Head >
<title> Розмір шрифту </ title>
</ Head>
< Body Bgcolor = fad 69 f >
<h1> <i> Тема <H1> </ i> - Приклад розміру тексту </ h1>
<p> <font size = "7"> Розмір шрифту = 7 </ font>
  <p> <font size = "6"> Розмір шрифту = 6 </ font>
  <p> <font size = "5"> Розмір шрифту = 5 </ font>
  <p> <font size = "4"> Розмір шрифту = 4 </ font>
  <p> <font size = "3"> Розмір шрифту = 3 </ font>
  <p> <font size = "2"> Розмір шрифту = 2 </ font>
  <p> <font size = "1"> Розмір шрифту = 1 </ font>
```

</ Body> </ html>



Малюнок 5. Наочний приклад розміру шрифту тексту.

Також можливе завдання розміру шрифту щодо шрифту «за замовчуванням» у вигляді: **size = + 2** або **size = -1** . Звичайний експлоереровский розмір це 3, але у кожного користувача можуть бути і свої налаштування. Розмір шрифту «за замовчуванням» для конкретної сторінки можна задати тегом **<basefont size = 5>** -Розташування в «голові» сторінки, для установки даного розміру відповідно рівному 5.

Для тексту що знаходиться в кожній з комірок таблиці слід застосовувати власний тег <FONT> (це стосується і кольору шрифта)

Розміщення тексту буває і зовсім без міток, в цьому випадку стає неможливим його вирівнювання (що в принципі буває і непотрібно), а шрифт приймає розмір відповідний <font size = "3">.

Колір шрифту задається параметром color = "колір" наприклад: **<font color = green size = + 2>** Розмір шрифту = 5 **</ font>** видасть той же рядок що і в попередньому прикладі тільки зелений. При цьому черговість в завданні параметрів будь-якого тега неважлива.

Розмір, тип і колір основного шрифту можна встановити за замовчуванням тегом < **BASEFONT** > - базовий шрифт

Basefont встановлює основний розмір шрифту, який застосовується до звичайного і попередньо відформатованого тексту, але не до заголовків, за винятком тих, які модифікуються з використанням елемента FONT із зазначеним відносним розміром шрифту (наприклад, FONT SIZE = + 1), а також тип і колір шрифту. Дія цього елемента розповсюджується не на все. В Netscape, наприклад, BASEFONT не впливає на розмір шрифту в межах таблиці. *Таким чином, що би встановити розмір шрифту в межах таблиці, необхідно вставити елемент зміни шрифту в кожну клітинку!*

Даний тег використовує такі атрибути:

SIZE - розмір шрифту (від 1 до 7)

COLOR - колір вмісту елемента FONT

FACE - тип шрифту, яким програма перегляду буде виводити текст

Таким чином, тег виглядає так:

< BASEFONT SIZE = n >

або

<BASEFONT COLOR = колірна специфікація>

або

<BASEFONT FACE = тип шрифту >

Абзац

На відміну від текстових документів переривання рядків в HTML-файлах не суттево. При перегляді HTML-джерела в текстовому редакторі обрив рядка може відбуватися в будь-якому місці, але при перегляді в браузері це переривання буде проігноровано. Замість 68 прогалин браузер покаже в тексті тільки одну, якщо тільки ви не використовуєте Авторський стиль.

Розбиття документа на складові задається за допомогою таблиць, міток форматування: <p> і </ p> (від **p** aragraf) а також < br > (від **br** eak). (Крім цього можлива вставка примусових прогалин за допомогою команди: & nbsp; (**n**o**b** reak **sp** ace) для кожного пробілу). Звичайний текст рекомендується розташовувати в абзацах: <p> текст </ p>.

У цьому випадку він буде виділений новим рядком. (Закриваючий тег необов'язковий. У разі якщо його пропущено, наступний абзац почнеться з наступної мітки яка відкриває <P>, проте його застосування іноді доцільно, наприклад для переривання вирівнювання).

Редагування тексту усередині абзацу може проводиться із застосуванням Логічних і Фізичних стилів форматування. При логічному форматуванні, Ви повідомляєте browsers що бажаєте виділити фрагмент тексту. При цьому браузери різних виробників будуть виділяти цей текст на свій розсуд. При физичному форматуванні Ви самі ставите стильове рішення шрифту - жирний,

під нахилом, закреслений і т.д. в цьому випадку всі browsers будуть підкорятися Вашим наказам). Таким чином,

<P> ... </ P>

ця пара тегів описує абзац. Все, що укладено між <P> і </ P> , сприймається як один абзац.

Відступ від краю

Абзаци можуть бути виконані з відступом від краю за допомогою тегів: <dl> і </ dl>. Ці теги добре підходять для виділеня важливих абзаців з одноманітного тексту. Структура взаємодії тегів досить складна, і має кілька внутрішніх команд. В межах цих тегів наприклад, не повинно бути тегів <p>.

Примітка: якщо виділеного абзацу не виходить, то треба вставити після тега який відкривається команду «Червоного рядка»: <dd> який потребує закриття.

Вирівнювання

Велике значення в HTML має можливість вирівнювання. Наприклад, наступний рядок: <P ALIGN = "CENTER"> Вирівнювання по центру </ P> видасть на екрані:

Для align (читається «Елайн», від англійського «вирівнювати») допустимі значення: «center» (вирівнювання по центру), «right» (вирівнювання по правому краю) і «justify» (рівномірний розподіл по всій довжині рядка **тільки для тексту**). За замовчуванням, браузер сам рівняє всі елементи по лівому краю. Вирівнювання може застосовуватися до тегів: <p>, <table>, до картинок і заголовків.

Так само в HTML існує спеціальний тег вирівнювання: ... </ center>. Все, що розташоване між цими мітками буде вирівняно, так само як і по команді <p align = "center">

Тобто теги <Hi> і <P> можуть містити додатковий атрибут ALIGN , наприклад:

<H1 ALIGN = CENTER> Вирівнювання заголовка по центру </ H1>

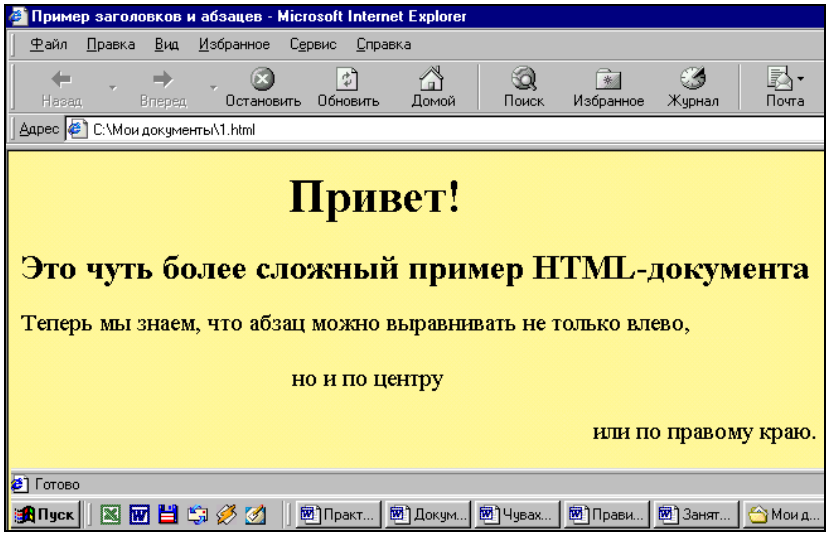
або

<P ALIGN = RIGHT> Зразок абзацу з вирівнюванням по правому краю </ P>

Все вищеописане можна побачити на наступному прикладі:

```
< Html >
< Head >
< Title > Приклад заголовків і абзаців </ title >
</ Head>
<body BGCOLOR = fff99d>
<H1 ALIGN = CENTER> Привіт ! </ H1>
<H2> Це трохи складніший приклад HTML-документа </ H2>
<P> Тепер ми знаємо, що абзац можна вирівнювати не тільки вліво, </ P>
```

<P ALIGN = CENTER> але і по центру </ P> <P ALIGN = RIGHT> або по правому краю. </ P>
</ Body>
</ Html>
Наочний вид прикладу представлений на малюнку 6.



Малюнок 6. Приклад HTML -сторінки з заголовками, абзацами і вирівнюванням.

3. ПОРЯДОК ВИКОНАННЯ РОБОТИ

• Ознайомитися з основними теоретичними положеннями;

• Виконати завдання

- Створіть HTML-документ з наступним заголовком: «Це моя перша Web-сторінка». Документ повинен спиратися тільки на основні елементи і виводити наступний текст: « **Сленг** - це слова, які часто розглядаються як порушення норм стандартної мови. Це дуже виразні, іронічні слова, які послуговують для позначення предметів, про які говорять в повсякденному житті ».

Примітка : для створення Web -сторінки необхідно створити документ з розширенням файлу * .txt (наприклад, Student .txt) з мінімальним кодом (див. Приклад 1). Потім поміняйте розширення файлу * .txt на * .htm.Закрийте його.

- Клацніть мишею по значку створеної вищеописаним способом WEB-сторінки. Після її відкриття поставте курсор миші в будь-яку точку сторінки яка відкрилася в Explorere і клацніть правою кнопкою. У меню виберете «перегляд вHTML ».

Відредагуйте текст, у вікні текстового редактора, додавши в нього наступний абзац: « Сленг складається зі слів і фразеологізмів, які виникли і спочатку вживалися в окремих соціальних групах і відображав цілісну орієнтацію цих груп ». Текст абзацу повинен бути темно-синім або темно-

зеленим. Для фонового малюнка використовуйте будь який графичний файл у форматі JPG або GIF .

Закрийте редактор.

На питання про збереження дайте відповідь позитивно.

У Explorere клацніть «оновити».

• Створити WEB-сторінку використовуючи матеріал інформаційного блоку практичної роботи №4. Найважливішу інформацію виділіть іншим кольором. Текст документа зробіть темно-синім або темно-зеленим кольором. Фоновий малюнок повинен бути нейтральним.

• Відповісти на контрольні питання.

4. Контрольні питання

- Що таке HTML-документ? Що являє собою мова HTML?
- З яких частин складаються теги?
- Які обов'язкові теги повинен включати в себе будь-який HTML-документ?
- Для чого застосовуються теги <HEAD> і <TITLE>?
- З чого може складатися тіло HTML-документа?
- Які атрибути використовуються в тезі <body>?
- Якими тегами задається колір елемента? Які атрибути він може включати?
- Що таке метаінформація? Для чого вона використовується?
- Для чого застосовуються заголовки в HTML-документах? Якими тегами вони описуються?
- За допомогою, яких команд задається розмір шрифту в тексті?
- Яким параметром задається колір шрифту в документі?
- Яким тегом задається розмір, тип і колір основного шрифта? Які атрибути при цьому використовуються?
- Яким чином в HTML-документах встановлюються абзаци в тексті?
- Яким чином в HTML-документах задається вирівнювання текста?

ПРАКТИЧНА РОБОТА № 7

Тема: **Правила створення HTML документів: Организация списків. Авторський стиль редагування**

1. ЗАГАЛЬНІ ВІДОМОСТІ

Мета роботи: Вивчити організацію списків в HTML-документах і їх особливості, отримати практичні навички їх створення.

Оснащення: індивідуальні методичні вказівки для самостійної роботи, науково-методична література, періодична преса, матеріал, підготовлений в процесі виконання попередніх практичних робіт, електронний підручник «Основи проектування педагогічних програмних засобів», електронний підручник виртуального факультативу.

2. ОСНОВНІ ТЕОРЕТИЧНІ ПОЛОЖЕННЯ

HTML дозволяє визначати зовнішній вигляд цілих абзаців тексту. Абзаци можна організовувати в списки, виводити їх на екран у відформатованому вигляді, або збільшувати ліве поле. Розберемо все по порядку.

Маркери: <UL> ... </ UL>

Текст, розташований між мітками <UL> і </ UL> , сприймається як маркований список. Кожен новий елемент списку слід починати з мітки <LI> .

<UL> <LI> яблука <LI> банани </ UL>

Дає на екрані:

- яблука
- банани

У тегах рядків «LI» списків, при введенні параметра title = "Підказка" можлива вставка пояснень, що спрацьовує при наведенні на рядок.

Закривають малі теги </ li> необов'язкові, але, як правило, їх завжди пишуть.

Нумеровані списки: <OL> ... </ OL>

Нумеровані списки влаштовані точно так же, як марковані, тільки замість символів, що виділяють новий елемент, використовують цифри.

До нумерованого списку < OL > можливо додавання атрибуту TYPE , який дає можливість задавати значення нумерації. Наприклад: 1 - звичайні (арабські) числа (1,2,3, ... n); a - латинські підрядкові літери (a , b , c , ... n); A - латинські надмалі літери (A , B , C , ... n); i - римські маленькі цифри (i , ii , iii , ... n); I - римські великі цифри (I , II , III , ...).

< OL TYPE = I >

<LI> олівець <LI> блокнот <LI> ручка

</ OL>

Дає на екрані:

I. олівець
II. блокнот
III. ручка

Якщо немає ніяких додаткових атрибутів Browser автоматично нумерує елементи такого списку арабськими цифрами.

<OL>

<LI> олівець <LI> блокнот <LI> ручка

</ OL>

Дає на екрані:

1. олівець
2. блокнот
3. ручка

Закривають малі теги </ li> необов'язкові, але, як правило, їх завжди пишуть.

Списки визначень: <DL> ... </ DL>

Список визначень дещо відрізняється від інших видів списків. Замість тегів <LI> в списках визначень використовуються теги <DT> (від англійського definition term - кожний термін) і <DD> (від англійського definition definition – визначення самого визначення). Розберемо це на прикладі. Припустимо, у нас є наступний фрагмент HTML-тексту:

<Html>

<Head>

<Title> Приклад списків визначень </ title>

</ Head>

<Body BGCOLOR = fff99d>

<DL>

<DT> HTML

<DD> Термін HTML (HyperText Markup Language) означає мову маркування гіпертекстів. Першу версію HTML розробив співробітник Європейської лабораторії фізики елементарних частин Тім Бернерс-Лі.

<DT> HTML-документ

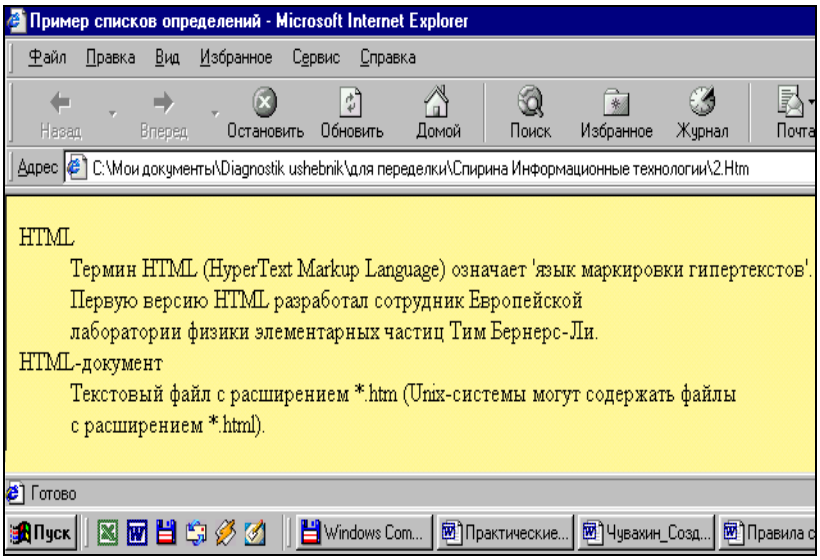
<DD> Текстовий файл з розширенням * .htm (Unix-системи можуть містити файли з розширенням * .html).

</ DL >

</ Body >

</ Html >

Цей фрагмент буде виведений на екран як показано на рисунку 1.



Малюнок 1. Наочний приклад списку визначень.

Зверніть увагу: точно так же, як тег <LI> , у тегів <DT> і <DD> закривають малі теги </ DT > і </ DD > необов'язковий.

Вкладені списки

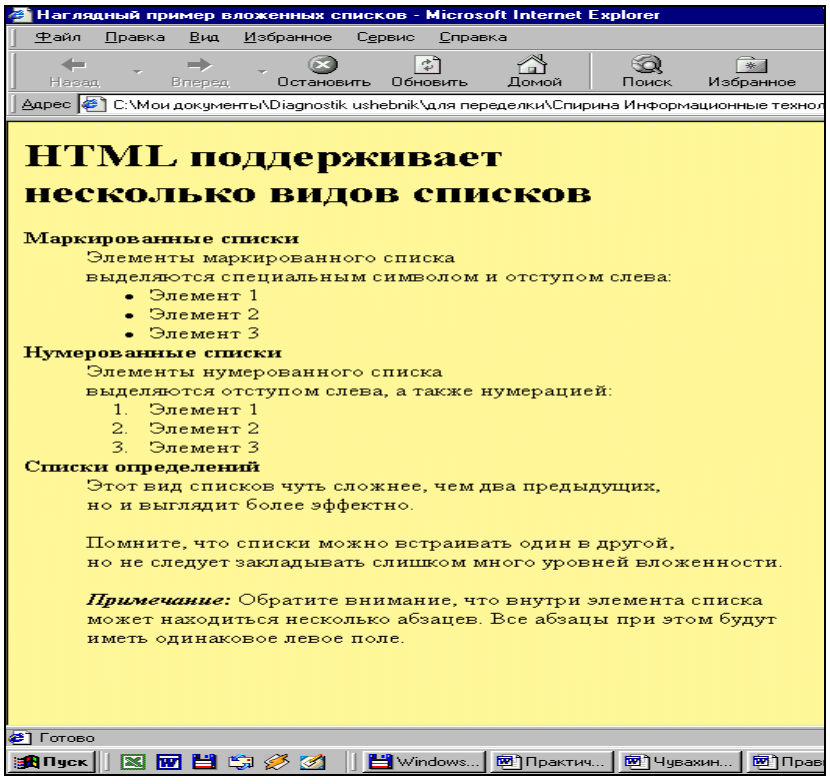
Елемент будь-якого списку може містити в собі цілий список будь-якого виду. Число рівнів вкладеності в принципі не обмежено, проте зловживати вкладеними списками все ж не слід. Вкладені списки дуже зручні при підготовці різного роду планів і змістів. наприклад:

```
< Html >
< Head >
< Title > Наочний приклад вкладених списків </ title >
</ Head >
< Body BGCOLOR = fff 99 d >
<H1> HTML підтримує кілька видів списків </ H1>
<DL>
<DT> < b > Маркери </ b >
<DD> Елементи маркованого списку виділяються спеціальним символом і відступом зліва:
<UL>
<LI> Елемент 1
<LI> Елемент 2
<LI> Елемент 3
</ UL>
<DT> < b > Нумеровані списки </ b >
```

<DD> Елементи нумерованого списку виділяються відступом зліва, а також нумерацією:

<OL>
<LI> Елемент 1
<LI> Елемент 2
<LI> Елемент 3
</ OL>
<DT> < b > Списки визначень </ b >
<DD> Цей вид списків трохи складніше, ніж два попередніх, але і виглядає більш ефектно.
<P> Пам'ятайте, що списки можна вбудовувати один в інший, але не слід закладати занадто багато рівнів вкладеності. </ P>
<P> Зверніть увагу, що всередині елемента списку може знаходитися кілька абзаців. Всі абзаци при цьому будуть мати однаково ліве поле. </ P >
</ DL >
</ Body >
</ Html >

Наочний приклад вкладених списків представлений на рисунку 2.



Малюнок 2. Зовнішній вигляд HTML -сторінки з вкладеними списками.

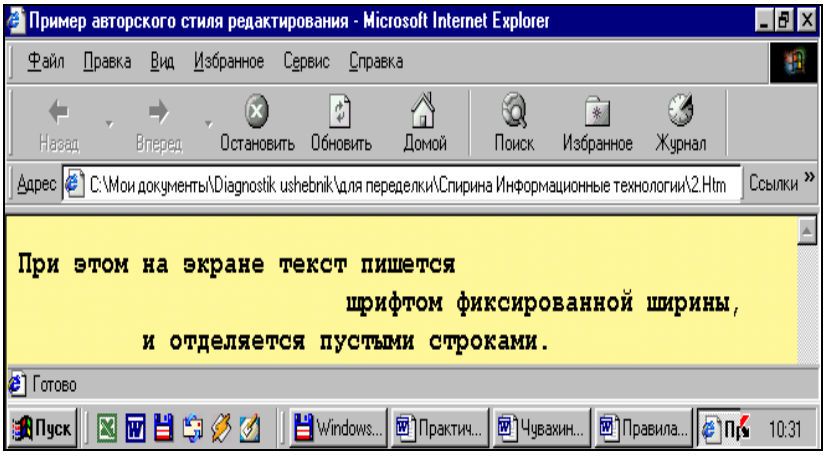
Авторський стиль редагування

Текст документа формується browser, які ігнорують множинні пробіли і символи кінця рядка. Однак, якщо використовувати <pre> ... </ pre> (від англійського preformatted - попередньо форматований), виводиться браузером на екран як є - з усіма пробілами, символами табуляції і кінця рядка. Це дуже зручно при створенні простих таблиць.

наприклад:

< Html >
< Html >
< Head >
<Title> Приклад авторського стилю редагування </ title>
</ Head>
<Body Bgcolor = fff99d>
<Pre> <H3>
При цьому на екрані текст пишеться шрифтом фіксованої ширини,
і відділяється порожніми рядками. </ pre>
</ Body>
</ Html>

Наочний вид HTML -сторінки представлений на малюнку 3.



Малюнок 3. Зовнішній вигляд HTML -сторінки з авторським стилем редагування.

В межах <PRE> можуть використовуватися гіперпосилання. Про те, в межах <PRE> , необхідно уникати використання інших методів форматування.

Текст з відступом: <BLOCKQUOTE> ... </ BLOCKQUOTE>

Текст, укладений між мітками <BLOCKQUOTE> і </ BLOCKQUOTE> , виводиться браузером на екран зі збільшеним лівим полем.

3. ПОРЯДОК ВИКОНАННЯ РОБОТИ

- Ознайомитися з основними теоретичними положеннями;
- Створити HTML-документ з заголовком «Я вчуся організовувати списки», який буде виводити нумерований список структурно-логічної схеми вивчення матеріалу, підготовленого в практичній роботі № 1. Використовувати для тксту заголовок вашої теми розробленого уроку 1 рівня. Вирівняти заголовок по центру, а решта тексту по лівому краю. При нумерації використовувати римські надрядкові цифри. Колір тексту повинен бути різнобарвним.
- Створити HTML-документ з заголовком «Списки визначень», який буде виводити список визначень за розробленим вами матеріалом до уроків. Використовувати для списку заголовок 1 рівня.
- Створити HTML-документ з заголовком розробленої теми до уроку. Документ повинен виводити вкладений список розробленого вами матеріалу уроку, розділеного на модулі і блоки. В якості основного списку (модулі) використати нумерований список римськими цифрами надрядкового рівня, як вкладеного списку (блоки) використовувати нумерований список арабськими цифрами, а в якості вкладеного списку (подблоки) використовувати маркований список. В якості фону використовувати будь яке фонове зображення.
- Відповісти на контрольні питання.

4. Контрольні питання

- Що таке список?
- Які використовуються списки?
- Що таке маркований список? Привести приклади таких списків.
- Що таке нумерований список? Привести приклади таких списків.
- Яким чином можна змінити вигляд нумерованого списка в HTML-документі?
- Для чого використовуються списки визначень? Які теги при цьому викортстовують?
- Що являє собою вкладений список?
- Для чого використовується тег <PRE>?
- Що таке авторський стиль редагування?

ПРАКТИЧНА РОБОТА № 8

Тема: **Правила створення HTML документів: Стилі форматування.**

1. ЗАГАЛЬНІ ВІДОМОСТІ

Мета роботи: Ознайомитися зі стилями форматування HTML-документів, навчитися виконувати різні види форматування при їх створенні.

Оснащення: індивідуальні методичні вказівки для самостійної роботи, науково-методична література, періодична преса, матеріал, підготовлений в процесі виконання попередніх практичних робіт, електронний підручник «Основи проектування педагогічних програмних засобів», електронний підручник віртуального факультативу.

2. ОСНОВНІ ТЕОРЕТИЧНІ ПОЛОЖЕННЯ

HTML допускає два підходи до шрифтового виділення фрагментів тексту. З одного боку, можна прямо вказати, що шрифт на деякій ділянці тексту повинен бути жирним або *похилим* , тобто змінити фізичний стиль тексту. З іншого боку, можна помітити певну частину тексту яка має відмінний від нормального логічний стиль, залишивши інтерпретацію цього стилю браузера.

Фізичні стилі форматування

Під фізичним стилем форматування прийнято розуміти пряму вказівку браузеру на модифікацію поточного шрифту. Наприклад, все, що знаходиться між тегами <B> і </B> , буде написано **жирним шрифтом** . Текст між тегами <I>і </I> буде написаний *похилим шрифтом* . Текст, розміщений між тегами <TT> і </TT> , буде написаний шрифтом, що імітує ю друкарську машинку , тобто мають фіксовану ширину символу. Розглянемо види фізичних стилів Форматування:

<b> Жирний шрифт </ b> **Жирний шрифт**
<i> Похилий шрифт </ i> *Похилий шрифт*
<s> Перекреслити шрифт </ s> ~~Перекреслити шрифт~~
<u> Підкреслений шрифт </ u> Підкреслений шрифт
<tt> шрифт фіксованої ширини </ tt> fixed шрифт
<sup> Верхній математика. шрифт </ sup> Верхній шрифт (індекс)
<sub> Нижній математика. шрифт </ sub > Нижній шрифт (індекс)
<big> Збільшений шрифт </ big> Збільшений шрифт
<small> Зменшений шрифт </ small> зменшений шрифт

Теги форматування можуть бути вкладені одна в одну в будь-якій кількості. Пам'ятайте про їх своєчасне закриття:

<i> <u> <b> <s> Приклад </ s> показує </ b> властивості </ u> вкладених </ i> тегів

~~Приклад~~ показує *властивості вкладених* тегів.

Логічні стилі форматування

При використанні логічних стилів автор документа не може знати заздалегідь, що побачить на екрані читач. Різні браузери тлумачать одні й ті ж мітки логічних стилів по-різному. Деякі браузери ігнорують деякі мітки взагалі і показують нормальний текст замість виділеного логічним стилем. Тому використовувати логічний стиль форматування слід дуже обережно.

Ось найпоширеніші логічні стилі.

<EM> ... </ EM>

Від англійського emphasis - акцент. Це тег виділеного шрифту - в основному виділення представляється курсивом.

 Тег виділеного шрифту </ em> *Тег виділеного шрифту*

<STRONG> ... </ STRONG>

Від англійського strong emphasis - сильний акцент . При використанні цього тега відбувається виділення напівжирним шрифтом. Крім цього браузери можуть використовувати і підкреслювання.

<CODE> ... </ CODE>

Рекомендується використовувати для фрагментів вихідних текстів.

< SAMP> ... </ SAMP>

Від англійського sample - зразок. Рекомендується використовувати для демонстраційних зразків повідомлень, що виводяться на екран програмами.

<samp> Тег шрифту зразків </ samp> Тег шрифту зразків

<KBD> ... </ KBD>

Від англійського keyboard - клавіатура. Рекомендується використовувати для вказівок того, що потрібно ввести з клавіатури.

<VAR> ... </ VAR>

Від англійського variable - змінна. Рекомендується використовувати для написання імен змінних, тобто в основному служить для оформлення назв програмних змінних або обраних користувачем параметрів комп'ютерної команди.

<Ins> Тег шрифту вставки </ ins> Тег шрифту вставки

<Del> Тег віддаленого шрифту </ del>

<acronym title = "Підказка"> Підказка </ acronym> Підказка

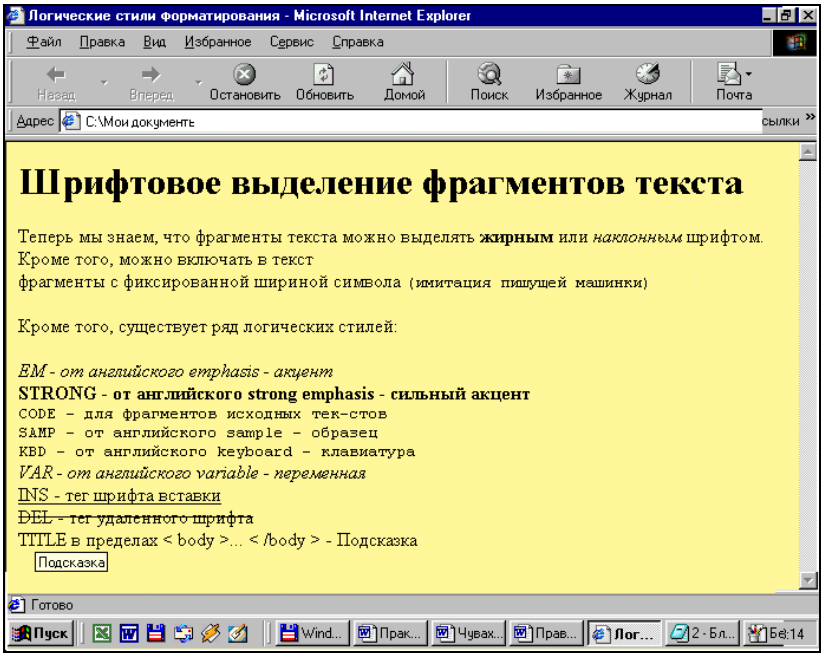
 Тег важливих фрагментів </ strong> Тег важливих фрагментів.

В основному в електронних підручниках тег підказки використовується в лівій стороні (фреймі) підручників для опису гиперпосилань (підказка спливає

при утриманні на ньому курсору миші). Останні доопрацювання специфікації html внесли суттєві зміни в актуальності використання даного тега в якості коментаря. Всі нові браузери повинні виводити випливаючу підказку для будь-якого елемента в межах <BODY> ... </ BODY> за допомогою вставленого в тег параметра title = "Текст підказки."

Підсумуємо знання про логічні і фізичні стилі за допомогою прикладу. На малюнку 1. можна побачити, як браузер показує ті чи інші логічні стилі.

< Html >
< Head >
< Title > Логічні стилі форматування </ title >
</ Head >
< Body Bgcolor = fff 99 d >
<H1> Шрифтове виділення фрагментів тексту </ H1>
<P> Тепер ми знаємо, що фрагменти тексту можна виділяти



Малюнок 1. Зовнішній вигляд HTML -сторінки з логічними стилями форматування.

<B> жирним </ B> або <I> похилим </ I> шрифтом. Крім того, можна включати в текст фрагменти з фіксованою шириною символу

<TT> (імітація друкарської машинки) </ TT> </ P>
<P> Крім того, існує ряд логічних стилів: </ P>
<P> <EM> EM - від англійського emphasis - акцент </ EM>
<STRONG> STRONG - від англійського strong emphasis - сильний акцент </ STRON G>
<CODE> CODE - для фрагментів вихідних текстів </ CODE>

<SAMP> SAMP - від англійського sample - зразок </ SAMP>
<KBD> KBD - від англійського keyboard - клавіатура </ KBD>
<VAR> VAR - від англійського variable - змінна </ VAR> </ P>
<INS> Тег шрифту вставки </ INS>
<DEL> Тег віддаленого шрифту </ DEL>
<ACRONYM TITLE = "Підказка"> Підказка </ ACRONYM>
</ Body >
</ Html >

Примусове переривання рядка

Використовуючи тег
 можна перейти на новий рядок всередині абзацу наприклад текст такого вигляду:

<p> Лабораторія оновлення змісту професійного навчання
 Кафедри загальнотехнічних дисциплін і методики трудового навчання
 Карагандинського державного університету ім. Е.А. Букетова </ p>

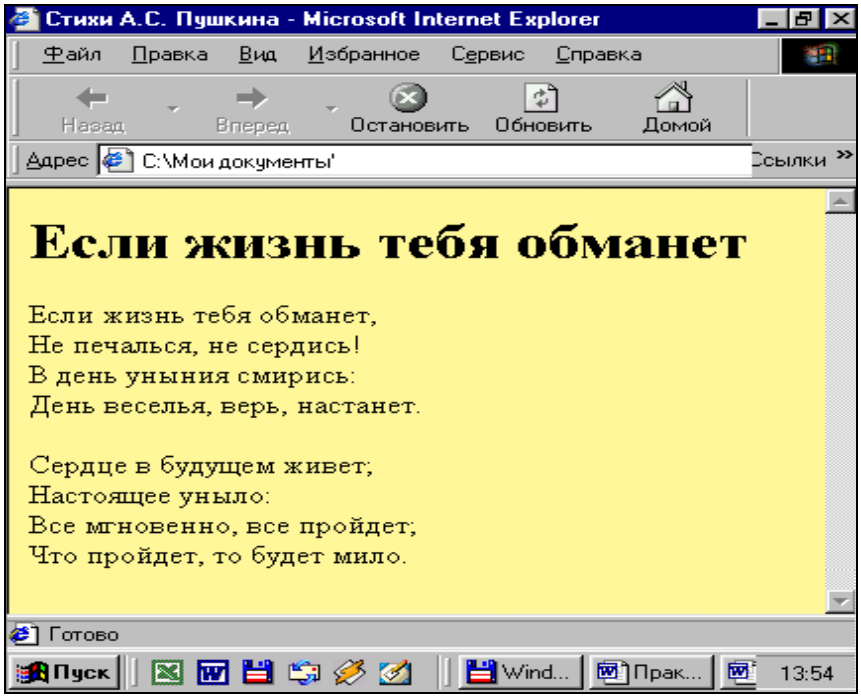
Отримаємо на екрані:

Лабораторія оновлення змісту професійного навчання
Кафедри загальнотехнічних дисциплін та методики трудового навчання
Карагандинського державного університету ім. Е.А. Букетова

Для заборони розриву браузером особливо важливого рядка (рекомендується використовувати в словах типу: чи-що, як-то, як-небудь і т.п.) відокремте потрібний уривок тексту тегами: <nobr> Ваш нерозривний текст </ nobr >

Примусове переривання рядка дуже зручно використовувати при публікації віршів, приклад представлений на малюнку 2.

<Html>
<Head>
<title> Вірші А . З . Пушкіна </ title>
</ he ad>
<body Bgcolor = fff99d>
< H 1> Якщо життя тебе обдурить </ H 1>
<P> Якщо життя тебе обдурить,
Не журися, не сердься!
В день зневіри упокорся:
День веселощів, вір, настане. </ P>
<P> Серце в майбутньому живе;
Справжнє понуро:
Все миттєво, все пройде;
Що пройде, то буде мило. </ P>
</ Body > </ html >



Малюнок 2. Зовнішній вигляд HTML -сторінки з примусовим перериванням рядків

Горизонтальна лінія

Використовуючи тег <HR> Ви можете розділити абзаци горизонтальною лінією. Тега що закриває немає. Використовуючи атрибути size (висота), width (довжина), можна вказувати висоту в пікселях і довжину в пікселях або відсотках. Використовуючи атрибут color (колір), можна задати колір лінії, а атрибут align зі значеннями left , right , center вирівнюють горизонтальне місцезнаходження риски. Без цього атрибута риска автоматично вирівнюється посередині. Наочний вид прикладу наведеного нижче коду представлений на малюнку 3: <hr size = "8" width = "40%" color = "red"> Даний код намалює таку лінію:

Малюнок 3. Горизонтальна лінія.

Примітка : якщо колір не вказано, то за замовчуванням встановлюється зазвичай сірий.

Рядок що біжить

Звичайний рядок що біжить (працює тільки в Internet Explorer).

Він може бути створений виключно засобами HTML без використання будь-яких скриптів, за допомогою тега < marquee > і наступних атрибутів:

BGCOLOR - фон;
WIDTH - ширина поля рядка в% або пікселях;
HEIGHT - визначає висоту поля в пікселях;
HSPACE - вертикальний інтервал від краю поля до тексту;
VSPACE - горизонтальний інтервал від краю поля до тексту;
ALIGN - визначає центрування тексту в поле рядка (значення: top - вгорі, bottom - внизу, middle - посередині);
BEHAVIOR - тип руху рухомого рядка (значення: scroll - текст з'являється біля одного краю і зникає за іншим, slide - біжучий рядок виходить з одного краю профіля і зупиняється біля протилежного краю, alternate - задає змінну напрямку - від одного краю до іншого і назад;
DIRECTION - напрямок руху рухомого рядка (значення: right - зліва на право, left - справа наліво);
SCROLLAMOUNT - переміщення в пікселях за одну ітерацію;
SCROLLDELAY - пауза між ітераціями в тисячних частках секунди;
LOOP - число повторень тексту в рухомому рядку (значення: або відповідне число, або infinite - рядок, що біжить повторюється необмежену кількість разів.

Для створення рядка, що біжить в IE необхідно написати наступне: <marquee> Текст </ marquee> Оформити рядок можна наступним чином:

<marquee behavior = "alternate" bgcolor = "yellow" scrolld e lay = "145" width = "50%"> <font size = "5" color = "red"> Текст рядка що біжить . </ Font > </ marquee >

behavior = "alternate" - команда, яка змушує рядок рухатися між краями. За замовчуванням даної команди, рядок рухається справа наліво. BGCOLOR - колір фону. Scrolldelay - швидкість переміщення. Width - довжина рядка, що біжить (необовязково у відсотках). font size = "5" - розмір шрифту що біжить, color = " red " - відповідного кольору шрифту що біжить. Останні два параметри не належать до атрибутів < marquee >

Швидкість переміщення рухомого рядка можна регулювати, змінюючи значення атрибутів SCROLLAMOUNT і SCROLLDELAY. Необхідно відзначити те, що тег <marquee> дозволяє переміщувати по горизонталі так само і графічні об'єкти. Для цього просто замість тексту треба помістити картинку за допомогою звичайного тега <img src>.

Спеціальні символи

Оскільки символи «<» і «>» сприймаються браузерами як початок і кінець HTML-тегів, виникає питання: а як показати ці символи на екрані? В HTML це робиться за допомогою & послідовностей (їх ще називають символьними обєктами або ескейп-послідовностями). Браузер показує на екрані символ «<», коли зустрічає в тексті послідовність & lt; (по першим літерам англійських слів less than - менше, ніж). Знак «>» кодується послідовністю & gt; (за першими

літерами англійських слів greater than - більше, ніж). Символ «&» (амперсанд) кодується послідовністю & amp; Подвійні лапки «" »кодуються послідовністю & quot; Пам’ятайте: крапка з комою - обов'язковий елемент & послідовності. Крім того, всі букви, складові послідовності, повинні бути в нижньому регістрі (тобто, маленькі). Використання тегів типу & QUOT; або & AMP; не допускається. Взагалі кажучи, & послідовності визначені для всіх символів з другої половини ASCII-таблиці (куди, природно, входять і російські літери). Справа в тому, що деякі сервери не підтримують восьмибітну передачу даних , і тому можуть передавати символи з ASCII- кодами вище 127 тільки у вигляді & послідовностей.

& Lt; - ліва дужка <

& Gt; - права дужка>

& Amp; - амперсанд &

& Quot; - лапки "

Також можна вводити примусові пробіли в будь-якій кількості за допомогою команди: & nbsp;.

Символи, наведені в цій таблиці не є знаками форматування HTML і їх використання говорить лише про культуру мови web дизайнера.

& # 167;	§
& # 169;	©
& # 174;	®
& # 176;	°
& Laquo ;	«
& Raquo ;	»
& # 133;	...
& # 145;	'
& # 146;	'

& # 132;	"
& # 147;	"
& # 148;	"
& # 149;	•
& # 150;	-
& # 151;	-
& # 153;	™
& # 8470;	№
& # 177;	±

3. ПОРЯДОК ВИКОНАННЯ РОБОТИ

- Ознайомитися з основними теоретичними положеннями;
- У створеному HTML-документі із заголовком «Списки визначень» практичної роботи № 7, відкоригувати виведені визначення в такому порядку: першим визначенням задати курсивний шрифт і вирівняти його по правому краю; для другого визначення задати напівжирний шрифт і вирівняти його по центру; в третьому визначенні саме поняття вивести телетайпним шрифтом, а текст задати підкресленим шрифтом. В якості фону сторінки використовувати будь яке фонове зображення.
- У скопійованому HTML-документі, який був створений в практичній роботі № 7 з вкладеним списком вивести, назви модулів напівжирним шрифтом з вирівнюванням по середині, всі назви блоків вивести курсивним напівжирним шрифтом вирівнюючи їх по лівому краю, а назви підблоків вивести підкресленим шрифтом також вирівнюючи текст по лівому краю. Весь текст

абзаців підблоків вивести звичайним розміром шрифту, який використовується за замовчуванням (розмір шрифту = 3). Після кожного блоку вивести горизонтальну лінію з будь якими заданими параметрами. В кінці документа створити рядок що біжить з резюме викладеного матеріалу (кількість слів не повинно перевищувати десяти). В якості фону використовувати будь-який фоновий малюнок або колір фону.

- Створити HTML-документ з заголовком «Я вмію застосовувати спеціальні символи». В даному документі повинен бути відображений текст, в якому будуть зустрічатися різні специальні символи. Для виведення цих символів використовувати escape-послідовності. В якості фону використовувати будь-який фоновий малюнок або колір фону.
- Відповісти на контрольні питання.

4. Контрольні питання

- Що можна задавати за допомогою тегів фізичного форматування?
- Перелічіть види фізичних стилів форматування? Якими тегами вони задаються?
- У чому відмінність логічних стилів форматування від фізичних? Для чого вони служать?
- Який тег використовується для виведення «підказки»?
- Для чого використовується тег примусового переривання рядка, і чим він позначається?
- Для чого використовують горизонтальну лінію? Яким тегом її позначають?
- Які атрибути використовують в тезі горизонтальної лінії? Перерахуйте їх.
- Яким тегом задається рядок, що біжить? Перерахуйте його основні атрибути.
- Для чого існують спеціальні символи?
- Які escape-послідовності можна додавати в HTML-документ і як?

ПРАКТИЧНА РОБОТА № 9

Тема: **Правила створення HTML документів: Зв'язування документів або гіперпосилання. Створення таблиць. Фрейми**

1. ЗАГАЛЬНІ ВІДОМОСТІ

Мета роботи: Вивчити з'єднання документів між собою певними зв'язками, навчитися використовувати ці зв'язки на практиці, отримати практичні навики створення таблиць в HTML -документах і створювати документи з різною фреймовою структурою.

Оснащення: індивідуальні методичні вказівки для самостійної роботи, науково-методична література, періодична преса, матеріал, підготовлений в процесі виконання попередніх практичних робіт, електронний підручник «Основи проектування педагогічних програмних засобів», електронний підручник виртуального факультативу.

2. ОСНОВНІ ТЕОРЕТИЧНІ ПОЛОЖЕННЯ

Зв'язування документів або гіперпосилання

Як ви вже знаєте, скорочення HTML означає «мова маркування гіпертекстів». Що ж таке гіпертекст? На відміну від звичайного тексту, який можна читати тільки від початку до кінця, гіпертекст дозволяє здійснювати миттєвий перехід від одного фрагмента тексту до іншого. Системи допомоги багатьох популярних програмних продуктів влаштовані саме за гіпертекстовим принципом. При натисненні лівої кнопки миші на деякий виділений фрагмент поточного документу відбувається перехід до деякого заздалегідь призначеного документа або фрагменту документа. В HTML перехід від одного фрагмента тексту до іншого задається за допомогою тега виду:

<A HREF="[адрес перехода]"> виділений фрагмент тексту </A>

Як параметр [адреса переходу] може використовуватися кілька типів аргументів. Найпростіше - це задати ім'я іншого HTML-документа, до якого потрібно перейти. Наприклад:

<A HREF="pr.htm"> Перейти до змісту </A>

Такий фрагмент HTML-тексту призведе до появи виділеного фрагмента Перейти до змісту , при натисканні на який в поточне вікно буде завантажений документ pr.htm .

Примітка: якщо в адресі переходу не вказано каталог (папка), перехід буде виконаний всередині поточного каталогу (папки). Якщо в адресі переходу не вказано сервер, перехід буде виконаний на поточному сервері. З цього випливає одне дуже важливе практичне міркування. Якщо Ви підготували до публікації певну групу HTML-документів, які посилаються один на одного тільки по імені

файла і знаходяться в одному каталозі (папці) на Вашому комп'ютері, вся ця група документів буде працювати точно так само, якщо її помістити в будь-який інший каталог (папку) на будь-якому іншому комп'ютері, на локальній мережі або ... в Інтернет! Таким чином, у Вас з'являється можливість розробляти цілі колекції документів без підключення до Інтернет, і тільки після остаточної готовності, підкріпленими випробуваннями, розміщувати колекції документів в Інтернет в цілому. Такий спосіб визначення шляху називається відносним посиланням, оскільки шлях до файлу що звязується вказується відносно місця розташування поточного файлу.

Перед завантаженням сторінок перевірте відповідність імен файлів. Якщо ви хочете дати посилання на файл NJStats.htm, що лежить у вкладеній папці AtlanticStates можна описати використовуючи слеш:

<A HREF="A tlanticStates/NJStats.htm"> New Jersey </A>

Якщо Ви перебуваєте в файлі ttt .html і посилаєтеся на вихідний документ Sk .html, то посилання буде виглядати так:

<A HREF="../Sk.html"> Казки </A>

Однак для файлів, з якими немає безпосереднього зв'язку, слід використовувати абсолютні адреси. Розглянемо, наприклад, групу документів, що утворюють вказівки для користувача. Посилання всередині цієї групи повинні бути відносними. У посиланнях на інші документи (наприклад, на схоже програмне забезпечення) повинні використовуватися абсолютні адреси файлів. В цьому випадку при перенесенні вказівок в іншу директорію посилання не потрібно буде оновлювати. Абсолютним посиланням називається посилання, в якому точно вказані комп'ютер, каталог і файл. У той час як відносні посилання застосовуються для Web-сторінок однієї машини, посилаючись на Web-сторінки інших комп'ютерів, необхідно використовувати абсолютні посилання.

Наприклад, якщо Ви хочете дати посилання на документ, що знаходиться в мережі то необхідно ввести в свій HTML-документ приблизно такий фрагмент:

 New Jersey

Таким чином, посилання, які ви даєте на документи, що знаходяться в мережі повинні бути абсолютними. Всі внутрішні посилання (забезпечують роботу вашої сторінки) повинні бути відносними, щоб уникнути постійних виправлень сторінки при перенесенні її в іншу папку.

Гіперпосиланнями можуть служити і графічні файли. Наприклад:

<A HREF="example.html"> <IMG SRC = "picture.gif"> </A>

За принципом дії графічні посилання нічим не відрізняються від текстових. Вони не підкреслюються і не виділяються кольором, а для їх виділення браузери зазвичай навколо такого зображення малюють рамку.

Крім цього, гіперпосилання можна давати не тільки на зображення і текст, а абсолютно на будьякі файли. Наприклад, якщо ви хочете розмістити у себе

посилання на піратський архів з грою Doom2 знаходиться в одній директорії з вашою сторінкою, то пишіть приблизно так:

 Doom2

В цьому випадку браузер відкриває вікно з питанням про збереження даного файлу на диску користувача.

При необхідності можна задати перехід не просто до деякого документу, але і до певного місця всередині цього документа. Для цього необхідно створити в документі, до якого буде поставлено перехід, деяку опорну точку, або **анкер**. Розберемо це на прикладі. Припустимо, що необхідно здіснити перехід з файлу 1.htm до слів "Перехід закінчений" в файлі 2.htm (файли знаходяться в одній папці). Перш за все, необхідно створити ось такий анкер в файлі 2.htm:

<A NAME=" AAA "> Глава 1 </A>

Слова «Глава 1» при цьому ніяк не будуть виділені в тексті документа. Потім у файлі 1.htm (або в будь-якому іншому) можна визначити перехід на цей анкер:

<A HREF="2.htm#AAA"> Глава 1 </A>

Таким чином, слово Глава 1 в документі перетворюється в анкер з ім'ям AAA.

Крім того, перехід до цього анкера можна визначити і всередині самого документа 2.htm - досить тільки включити в нього ім'я «AAA» зі знаком дієз «#» при цьому опускається ім'я файлу. наприклад:

<A HREF="#AAA"> Перехід до анкера AAA </A>

Примітка: знак дієз «#» обов'язковий !

На практиці це дуже зручно при створенні великих документів. На початку документа можна помістити зміст, що складається з посилань на анкери, розташовані в заголовках розділів документа. Щоб уникнути непорозумінь рекомендується ставити імена анкерів латинськими буквами. Слідкуйте за написанням імен анкерів: більшість браузерів відрізняють великі літери від маленьких. Якщо ім'я анкера визначено як AAA , посилання на анкер aaa або AaA не виведе Вас на анкер AAA , хоча документ, швидше за все, буде завантажений корректно крім посилань на HTML-документи. Можливі посилання і на інші види ресурсів:

<A HREF="ftp://server/directory/file.doc"> вивантажити файл </A>

Таке посилання, якщо ним скористатися, запустить протокол передачі файлів і почне вивантаження файлу file. doc , що знаходиться в каталозі directory на сервері server , на локальний диск користувача.

<A HREF="mailto:user@mail.box"> Послати лист </A>

Якщо користувач зробить перехід по такому посиланню, у нього на екрані відкривається вікно введення вихідного повідомлення його поштової програми. У рядку To: ("Куди") вікна поштової програми буде вказано user@mail.box .

Розберемо все, що, ми знаємо про зв'язування, за допомогою прикладу.

< HTML >
< HEAD >
< TITLE > Приклад зв'язування документів </ TITLE >
</ HEAD >
< BODY >
< H 1> Зв'язування документів </ H 1>
<P> За допомогою посилань можна переходити до інших файлів (наприклад, до <A HREF="pr.htm"> перехід до змісту вказівок </A>). </ P>
<P> Можна вивантажувати файли (наприклад, < A HREF = " ftp : // server / directory / file . Doc "> це керівництво в форматі Microsoft Word 2.0 </A>) по FTP. </ P>
<P> Можна дати користувачеві можливість послати пошту (наприклад, <A HREF="mailto:nc@iname.com"> автору цих вказівок </A>). </ P>
</ BODY>
</ HTML>

Таблиці

Таблиці потрібні для подання інформації в табличному вигляді. Є, однак, і менш очевидні відповіді. До тепер ми мали справу з документами, в яких існував тільки один "потік" тексту. На практиці іноді дуже хочеться розташувати текст в декілька колонок. Таблиця може в цьому допомогти. Крім того, таблиця, що складається з одного осередку, може дуже ефектно виділити фрагмент тексту, на який Ви хочете звернути увагу читача.Таблиця розміщується між тегами <table> ... </ table> . Теги < table > ... </ table > можуть містити наступні атрибути:

<TR> ... </ TR> - визначає рядок таблиці.
<TD> ... </ TD> - визначає осередок даних таблиці.
<TH> ... </ TH> - визначає осередок заголовка таблиці.

Дані теги записуються наступним чином: <TR> Заголовок осередків (TH елементи) і дані осередків (TD елементи) </ TR>

Як влаштована таблиця

У пристрої таблиці найлегше розібратися на простому прикладі.

<HTML>
<HEAD>
<TITLE> Приклад таблиці </ TITLE>
</ HEAD>
<H1> Найпростіша таблиця </ H1>
<TABLE BORDER = 1> <! - Це початок таблиці ->
<CAPTION> <! - Це заголовок табліци ->
У таблиці може бути заголовок

```
</ CAPTION>
<TR> <! - Це початок першого рядка ->
<TD> <! - Це початок першої чарунки ->
Перший рядок, перша колонка
</ TD> <! - Це кінець першого осередку ->
<TD> <! - Це початок другої чарунки ->
Перший рядок, друга колонка
</ TD> <! - Це кінець другого осередку ->
</ TR> <! - Це кінець першого рядка ->
<TR> <! - Це початок другого рядка ->
<TD> <! - Це початок першої чарунки ->
Другий рядок, перша колонка
</ TD> <! - Це кінець першого осередку ->
<TD> <! - Це початок другої чарунки ->
Другий рядок, друга колонка
</ TD> <! - Це кінець другого осередку ->
</ TR> <! - Це кінець другого рядка ->
</ TABLE> <! - Це кінець таблиці ->
</ BODY>
</ HTML>
```

Таблиця починається з тега **<TABLE>** і закінчується тегом **</ TABLE> .** Тег <TABLE> може включати кілька атрибутів.

Розглянемо їх:

ALIGN - Встановлює розташування таблиці по відношенню до полів документа. Можна вибрати зі значень ALIGN = LEFT (вирівнювання вліво), ALIGN = CENTER (вирівнювання по центру), ALIGN = RIGHT (вирівнювання вправо).

WIDTH - Ширина таблиці. Її можна задати в пікселях (наприклад, WIDTH = 400) або у відсотках від ширини сторінки (наприклад, WI DTH = 80%).

BORDER - Встановлює ширину зовнішньої рамки таблиці і осередків в пікселах (наприклад, BORDER = 4). Якщо атрибут НЕ встановлений, таблиця показується без рамки, деякі браузери допускають просто написання BORDER, яке сприймається, як BORDER = 1

C ELLSPACING - Встановлює відстань між осередками таблиці в пікселах (наприклад, CELLSPACING = 2).

CELLPADDING - Встановлює відстань між вмістом комірки і рамкою навколо осередку в пікселях («набивка») -

BGCOLOR - -встановлює колір фону під таблицею.

BORDERCOLOR - змінює колір сітки таблиці -

Таблиця може мати *заголовок* між тегами **<CAPTION>** ... **</ CAPTION> ,** хоча заголовок не є обов'язковим. Тег <CAPTION> може містити

атрибут ALIGN . Допустимі значення: <CAPTION ALIGN = TOP> (заголовок розміщується над таблицею) і <CAPTION ALIGN = BOTTOM> (заголовок розміщується під таблицею).

Кожен *рядок таблиці* починається з тега **<TR>** і закінчується тегом **</ TR>** . Тег <TR> може включати наступні Атрибути:

ALIGN - Встановлює вирівнювання тексту в осередках рядка. Можна вибрати зі значень ALIGN = LEFT (вирівнювання ліворуч), ALIGN = CENTER (вирівнювання по центру), ALIGN = RIGHT (вирівнювання праворуч).

VALIGN - Встановлює вертикальне вирівнювання тексту в осередках рядка. Можна вибрати зі значень VALIGN = TOP (вирівнювання по верхньому краю), VALIGN = MIDDLE (вирівнювання по центру), VALIGN = BOTTOM (вирівнювання по нижньому краю).

Кожна *клітинка таблиці* починається з тега **<TD>** і закінчується тегом **</ TD>** . За замовчуванням текст в осередку вирівнюється по лівому краю і центрується по вертикалі. Осередки даних можуть описуватися додатковими параметрами, що визначають характеристики осередку і / або її вмісту.

Тег <TD> може включати наступні атрибути:

NOWRAP - Присутність цього атрибуту означає, що вміст комірки повинен бути показаний в один рядок.

COLSPAN - Встановлює «розмах» осередка по горизонталі. Наприклад, COLSPAN = 3 означає, що осередок простягається на три колонки.

ROWSPAN - Встановлює «розмах» осередка по вертикалі. Наприклад, ROWSPAN = 2 означає, що осередок займає дві строки.

ALIGN - Встановлює вирівнювання тексту в комірці. Можна вибрати зі значень ALIGN = LEFT (вирівнювання ліворуч), ALIGN = CENTER (вирівнювання по центру), ALIGN = RIGHT (вирівнювання праворуч).

VALIGN - Встановлює вертикальне вирівнювання тексту в комірці. Можна вибрати зі значень VALIGN = TOP (вирівнювання по верхньому краю), VALIGN = MIDDLE (вирівнювання по центру), VALIGN = BOTTOM (вирівнювання по нижньому краю).

WIDTH - Встановлює ширину осередку в пікселах (наприклад, WIDTH = 200).

HEIGHT - Встановлює висоту комірки в пікселах (наприклад, HEIGHT = 40).

Якщо комірка таблиці порожня, навколо неї не малюється рамка. Якщо осередок порожній, а рамка потрібна, в клітинку можна ввести символьний об'єкт & nbsp; (non-breaking space - нерозривний пробіл). Осередок як і раніше буде порожній, а рамка навколо неї буде. Крім цього будь-яка комірка таблиці може містити в собі іншу таблицю.

Осередок таблиці може містити заголовок (строки чи стовбці) осередки таблиці розміщуються між тегами **<TH>** ... **</ TH>** .

Тег <T H > може включати наступні атрибути:

NOWRAP - Присутність цього атрибуту означає, що вміст комірки повинно бути показано в один рядок.

COLSPAN - Встановлює «розмах» осередку по горизонталі. Наприклад, COLSPAN = 3 означає, що осередок простягається на три колонки.

ROWSPAN - Встановлює «розмах» осередку по вертикалі. Наприклад, ROWSPAN = 2 означає, що осередок займає дві строчки.

ALIGN - Встановлює вирівнювання тексту в комірці. Можна вибрати зі значень ALIGN = LEFT (вирівнювання ліворуч), ALIGN = CENTER (вирівнювання по центру), ALIGN = RIGHT (вирівнювання праворуч).

VALIGN - Встановлює вертикальне вирівнювання тексту в комірці. Можна вибрати зі значень VALIGN = TOP (вирівнювання по верхньому краю), VALIGN = MIDDLE (вирівнювання по центру), VALIGN = BOTTOM (вирівнювання по нижньому краю).

WIDTH - Встановлює ширину осередку в пікселах (наприклад, WIDTH = 200).

HEIGHT - Встановлює висоту комірки в пікселах (наприклад, HEIGHT = 40).

Примітка : параметри, які стоять всередині осередків < TH > ... </ TH > or < TD > ... </ TD>, скасовують вирівнювання за замовчуванням, завдання в <TR> ... </ TR>.

Фрейми

Подібно таблицям фрейми ділять екран браузера на частини. Відмінність полягає в тому, що сторінка, що містить фрейми, взагалі не є сторінкою, так як не має тіла, тобто тега BODY і відповідно не містить будь-якої інформаці і цінного навантаження. У ній повинні розташовуватися теги FRAMESET, які створюють розмітку для завантаження на екран одночасно декількох WEB-сторінок.

Кожне підвікно, або Фрейм, може мати такі властивості:

- Кожен Фрейм має свій URL, що дозволяє завантажувати його незалежно від інших фреймів
- Кожен Фрейм має власне ім'я (параметр NAME), що дозволяє переходити до нього з іншого фрейму
- Розмір фреймів може бути змінений користувачем прямо на екрані за допомогою миші (якщо це не заборонено вказівками спеціального параметра)

Дані властивості фреймів дозволяють створювати просунуті інтерфейсні рішення, такі як:

- Розміщення статичної інформації, яку автор вважає за необхідне постійно показувати користувачу, в одному статичному фреймі. Це може бути графічний логотип фірми, copyright, набір керуючих кнопок

- Розміщення в статичному фреймі змісту всіх або частини WEB-документів, що містяться на WEB-сервері, що дозволяють користувачеві швидко знаходити потрібну йому інформацію
- Створювати вікна результатів запитів, коли в одному фреймі знаходиться власне запит, а в іншому результати запиту
- Створювати форми типу «майстер-деталь» для WEB-додатків, які обслуговують бази даних

<HTML>
<HEAD> ... </ HEAD>
<FRAMESET> ... </ FRAMESET>
</ HTML >

Програмно розбиття вікна браузера на фрейми реалізуємо так:

1. Створюється html файл (зазвичай це перша сторінка сервера в назві index.htm) в якому створюються розміри і кількість фреймів, а також імена файлів відповідних фреймів і деякі атрибути для кожного фрейму.

2. . Створюється окремі html сторінки для кожного фрейма.

Наприклад, для створення html-файлу який реалізує розбиття екрану на дві частини. Необхідно створити два звичайних html-файлу з іменами homepage.htm і menu.htm. Головний файл назвемо, наприклад, index.htm, ось як він повинен виглядіти:

< HTML >
< TITLE > Назва вашої сторінки </ TITLE >
<FRAMESET cols = "*, 140">
<FRAME SRC = "homepage.htm" NAME = "frame1">
<FRAME SRC = "menu.htm" NAME = "frame2">
</ FRAMESET >
</ HTML >

Розглянемо кожен тег окремо:

<HTML> </ HTML> і <TITLE> <TITLE> - стандартні теги для всіх html файлів

<FRAMESET> в цьому тегу задається кількість рядів або стовпців ROWS і COLS відповідно, а також їх розміри і розташування. Існує три способи завдання їх розміру:

- по пікселям - просто треба написати висоту або ширину в пікселях.

- відсотками треба написати скільки відсотків від вікна браузера віддається кадру і після цифр ставиться знак% Необхідно, щоб всі% -ки в сумі становили 100%. Якщо при додаванні значень результат не дорівнює 100%, то браузер сам розрахує розміри фреймів пропорційно заданим значенням.

- зірочка - все, що залишилося у вікні дорівнює значку *. Наприклад, можна написати 20%, 20%, 60% або 20%, 20%, * і ніякої різниці не буде.

У цьому ж тезі можна задати товщину розмежувальної лінії і окаймляючої рамки командами FRAMEBORDER = "X" і BORDER = "Y" де x і y товщина в пікселях.

У нашому випадку (<FRAMESET cols = "*, 140">) тим самим поділяємо вікно на два стовпці, праве шириною в 140 пікселів, а ліве шириною у весь екран.

Крім вищерозглянутих атрибутів можливе застосування і інших, наприклад таких як:

BORDERCOLOR - задає колір рамки

FRAMEBORDER - встановлює кордон рамки. Можливі такі значеня: YES - є межа, NO - немає межі

FRAMESPACING - встановлює ширину кордону

Тег <FRAME> задає атрибути для кожного фрейму персонально. Його називають одиночним фреймом. Йому властиві такі атрибути

SRC - вказується URL-адресу об'єкта завантажується в даний фрейм. Команда SRC задає ім'я файлу, який завантажиться в цьому фреймі, в нашому випадку ім'я файла homepage.htm (<FRAME SRC = "homepage.htm" ...).

SCROLLING - параметр для регулювання смуги прокрутки задається наступними значеннями: YES , NO , AUTO

MARGINWIDHT, MARGINHEIGHT - керують відступом всередині рамок, служать для вирівнювання графічного зображення всередині рамки. Також в цьому тегу можна задати величину межі фрейму, за яку нічого крім бекграунду не може заходити. Це робиться командами MARGINWIDTH = "x" і MARGINHEIGHT = "y", де x і y - величина в пікселах.

NORESIZE - параметр, який вказує на те, що розмір рамки-фрейма ніколи не мінятиметься.

NAME - визначає ім'я фрейма, яке може використовуватися для посилання до нього. Команда NAME задає ім'я даного фрейму, в нашому випадку ім'я "frame1". Ім'я необхідне, для того щоб надалі вказати до якого фрейму застосовувати посилання. Наприклад треба щоб натискаючи на посилання в фреймі який містить файл menu.htm вміст файлу посилання показувалось в фреймі що містить файл homepage.htm. Для цього необхідно відкорегувати html код посилання:

<A HREF="file.htm"> file </A> - що було

<A HREF="file.htm" TARGET="frame1"> file </A> - що повинно бути . А якщо треба щоб файл завантажився в головному вікні браузера то напишіть на посиланні TARGET = "_ top"

У гіпертекстовому посиланні використовується атрибут TARGET, який посилається на ім'я фрейма. Якщо атрибут TARGET не вказаний, то при виборі посилання вміст файлу, вказаного в атрибуті HREF, відображається в поточному вікні або фреймі. А якщо атрибут TARGET зазначений, то вміст

файлу, вказаного в атрибуті HREF, відображається у вікні або фреймі, зазначеному в атрибуті TARGET.

Значення атрибута Target:

- _BLANK - завантажує вміст сторінки, заданим посиланням, в нове порожнє вікно < A HREF = " page . Htm " TA R GET = "_ blank ">
- _SELF - вміст сторінки, заданим посиланням, у вікно, яке містить посилання < A HREF = " page . Htm " TA R GET = "_ self ">
- _PARENT - завантажує вміст сторінки, заданим посиланням, у вікно, яке є безпосереднім власником набору фреймів < A HREF = " page . Htm " TARGET = "_ parent ">
- _TOP - вміст сторінки, заданим посиланням, у вікно, ігноруючи фрейми які використовуються.

Примітка : якщо фрейму не присвоїти імені, то в ньому не можна буде відкрити документ або зображення по посиланню з інших фреймів. Тому треба присвоювати імена всім фреймам, вміст яких буде змінюватися при виборі посилання в іншому фреймі. Імена фреймів повинні складатися з букв і цифр. В якості першого символу імені фрейму не можна використовувати символ підкреслення (_), крім чотирьох зарезервованих імен, наведених вище. У всьому іншому обмежень на імена немає.

FRAMEBORDER - межа рамки: значення YES - є межа, NO - немає границі

BORDERCOLOR - задає колір рамки

BORDER - вказує ширину бордюру

</ FRAMESET> закриває тег.

Ось кілька прикладів створення фреймів:

*	140

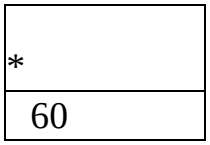
```
<FRAMESET cols = "*, 140">
<FRAME SRC = "homepage.htm" NAME = "frame1">
<FRAME SRC = "menu.htm" NAME = "frame2">
</ FRAMESET>
```

100	*

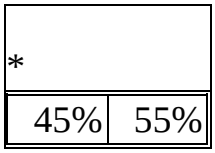
```
<FRAMESET cols = "100, *">
<FRAME SRC = "homepage.htm" NAME = "Frame1">
<FRAME SRC = "menu.htm" NAME = "Frame2">
</ FRAMESET>
```

100
*

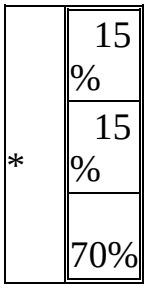
```
<FRAMESET rows = "100, *">
<FRAME SRC = "homepage.htm" NAME = "Frame1">
<FRAME SRC = "menu.htm" NAME = "Frame2">
</ FRAMESET>
```



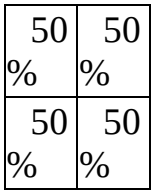

```
<FRAMESET rows = "*, 60">
<FRAME SRC = "homepage.htm" NAME = "Frame1">
<FRAME SRC = "menu.htm" NAME = "Frame2">
</ FRAMESET>
```




```
<FRAMESET rows = "*, 60">
<FRAME SRC = "homepage.htm" NAME = "Frame1">
<FRAMESET cols = "45%, 55%">
<FRAME SRC = "menu.htm" NAME = "Frame2">
<FRAME SRC = "menu2.htm" NAME = "Frame3">
</ FRAMESET>
</ FRAMESET>
```




```
<FRAMESET cols = "*, 55%">
<FRAME SRC = "homepage.htm" NAME = "Frame1">
<FRAMESET rows = "15%, 15%, 70%">
<FRAME SRC = "menu.htm" NAME = "Frame2">
<FRAME SRC = "menu2.htm" NAME = "Frame3">
<FRAME SRC = "menu3.htm" NAME = "Frame4">
</ FRAMESET>
</ FRAMESET>
```




```
<FRAMESET cols = "50%, 50%">
<FRAMESET rows = "50%, 50%">
<FRAME SRC = "homepage.htm" NAME = "Frame1">
<FRAME SRC = "homepage2.htm" NAME = "Frame2">
</ FRAMESET>
<FRAMESET rows = "50%, 50%">
<FRAME SRC = "menu.htm" NAME = "Frame3">
<FRAME SRC = "menu2.htm" NAME = "Frame4">
</ FRAMESET >
</ FRAMESET >
```

3. ПОРЯДОК ВИКОНАННЯ РОБОТИ

- Ознайомитися з основними теоретичними положеннями;
- Створіть HTML-документ, в якому будуть реалізовані посилання на всі попередні приклади сторінок і будь-які 5 сайтів (вузлів) в Інтернет. Оформіть сторінку за своїм смаком;
- Створіть HTML-документ, який буде містити якийсь текст. На початку тексту має бути розміщений його заголовок і зміст. Кожен пункт змісту повинен служити посиланням на певний розділ в документі. Оформити сторінку за своїм смаком;

- Створити HTML-документ, який буде містити наступну таблицю 1 з заголовком.

Таблиця 1. Сутнісні ознаки дистанційної освіти.

Автор, джерело	визначення дефініції	Сутнісні признаки
1	2	3
Концепція створення та розвитку системи дистанційного навчання	**Дистанційна освіта** - це комплекс освітніх послуг, що надаються широким верствам населення в країні і за рубежем за допомогою спеціалізованого інформаційно-освітнього середовища, яке базується на засобах обміну навчальною інформацією на відстані (супутникове телебачення, радіо, комп'ютерний зв'язок і т.п.).	Комплекс просвітницьких послуг
Волов В.Т.	**Дистанційна освіта** - форма освіти, що забезпечує використання сучасних технічних засобів і інформаційних технологій для доставки навчальних матеріалів та інформації безпосередньо до споживачів, незалежно від їх місцезнаходження.	Форма освіти
Агроновіч Б.Л., Чудінов В.Н.	**Дистанційна освіта** - це дидактична технологічна система особистісно-орієнтованої безперервної освіти, що реалізується за допомогою віртуального інформаційно-освітнього середовища, оперативного спілкування з якою забезпечується незалежно від місця проживання слухача і без порушення його звичайного укладу життя.	Дидактична технологічна система особистісно-орієнтованої безперервної освіти
Андрєєв А.А., Солдаткін В.І.	**Дистанційна освіта** - це система, в якій реалізується процес дистанційного навчання, створюються індивідуумом досягнення і підтвердження освітнього цензу.	система
Матеріали Сьомої Міжнардної конференції з дистанційної освіти	**Дистанційна освіта** - це така педагогічна система, в якій	Педагогічна система

1	2	3
	реалізується процес дистанційного навчання з підтвердженням просвітницького цензу.	
Бондар В.А., Воронін В.А., Жуков О.І. та ін.	**Дистанційна освіта** - це сучасна технологія навчання, нова для освітніх установ нашої країни, яка зобов'язана своїм виникненням розвитку інформаційних технологій і компютерної техніки.	Сучасні технології навчання
Основні поняття і терміни в сфері освіти: Словник / Упорядники. В.П. Бугрин, Н.В. Борісова	**Дистанційна освіта** - це цілеспрямоване і методично організоване керівництво навчально-пізнавальною діяльністю і розвитком осіб, які перебувають на віддалі від освітнього закладу і при цьому не вступають в постійний контакт з його педагогічним персоналом.	Освіта на відстані
А.Ю. Уваров	**Дистанційна освіта** - засоби, за допомогою яких навчальні матеріали і методичне забезпечення доставляються слухачеві. Його використовують, коли доставка або навчальна взаємодія забезпечуються за допомогою сучасних технічних засобів.	засіб

Вирівняти таблицю по центру сторінки. Для шапки таблиці задати блакитний колір фону, а для решти осередків - світло-зелений. Текст в першому стовпці поставити темно-синім кольором, колір в другому стовпці поставити темно-синім, колір тексту в третьому стовпці залишити без змін.

• Створити HTML-документ, який буде містити таблицю №2 з заголовком. Задати ширину для таблиці на весь екран.

Таблиця 2. Обгрунтування актуальності дослідження.

Обгрунтування актуальності напрямку	**Обгрунтування практичної актуальності теми дослідження**		**Обгрунтування наукової актуальності теми**
	Оцінка результатів педагогічного	Оцінка якості педагогічного процессу	

	процесу		
Показати значимість виділеної проблеми і необхідність її вирішення	Показати недоліки навченості і вихованості слухача, які слід усунути	Показати недоліки в педагогічному процесі, які ведуть до низької якості навчання і вихованості слухачів	Показати ступінь розробки виокремлиної проблеми в науці, вказати на недосить вивчені аспекти

- Розробити HTML-документи з різною фреймовою структурою (горизонтальне, вертикальне і змішане розбиття сторінки на фрейми) і використовуючи барвисто оформлені HTML-документи створені на основі завдань попередніх практичних робіт № 1-5, 7 «Назва теми розробленого матеріалу»:

- створити сторінку з двома фреймами, в одному з яких будуть відображатися модулі і блоки розробленого матеріалу, а в іншому фреймі буде відкриватися відповідний модулям і блокам матеріал у вигляді HTML-документів оформлені за вашим бажанням;

- створити сторінку з трьома фреймами, в одному з яких буде відображатися структурний матеріал вище виконаного завдання, в другому - питання для самоконтролю за розробленим матеріалом, а в третьому фреймі буде відображатися вже виконаний матеріал HTML-сторінки.

- Відповісти на контрольні питання.

4. Контрольні питання

- Що таке гіпертекст і гіперпосилання? За допомогою якого тега задається гіпер посилання?

- У чому відмінність між абсолютними і відносними посиланнями?

- Яким чином описуються гіперпосилання на графічні файли?

- Що таке анкер і для чого він потрібен? За допомогою, яких тегів та елементів можна здійснювати перехід між розділами всередині документа?

- Як влаштована таблиця? - Для чого служать таблиці в HTML-документах?

- Які атрибути можуть використовуватися в тезі <TABLE>?

- За допомогою яких тегів описуються рядки і осередки таблиці?

- За допомогою яких атрибутів можна об'єднувати і розбивати осередки в таблиці?

- За допомогою яких атрибутів задається колір меж таблиці і колір заливки (фону) всередині таблиці?

- Що таке фрейми? Коли використовуються фрейми?

- Який основний синтаксис опису сторінки з фреймами?

- Які атрибути може мати тег <FRAMESET>? Для чого використовується кожен атрибут?
- Навести приклади фреймів.
- Які існують атрибути зв'язку між фреймами?

ПРАКТИЧНА РОБОТА № 10

Тема: **Функціональні розділи. Кольори елементів.**

1. ЗАГАЛЬНІ ВІДОМОСТІ

Мета роботи: Вивчити правила створення HTML документів, ознайомитися зі структурою HTML документа його функціональними розділами, отримати практичні навички і досвід їх створення.

Оснащення: індивідуальні методичні вказівки для самостійної роботи, науково-методична література, періодична преса, матеріал, підготовлений в процесі виконання попередніх практичних робіт, електронний підручник «Основи проектування педагогічних програмних засобів», електронний підручник віртуального факультативу.

2. ОСНОВНІ ТЕОРЕТИЧНІ ПОЛОЖЕННЯ

Термін HTML (HyperText Markup Language) означає «мова маркування гіпертекстів». Першу версію HTML розробив співробітник Європейської лабораторії фізики елементарних частинок Тім Бернерс-Лі.

HTML-документ є текстовим файлом з розширенням htm (Unix-системи можуть містити файли з розширенням html). HTML документи можуть бути створені і відредаговані в будь-якому текстовому редакторі.

Мова HTML представляє з себе безліч команд, укладених між знаками «<» і «>», наприклад, <html> - це називається міткою (англійською - tag, читається «тег»). Більшість HTML-міток - парні, тобто на кожну мітку що відкривається виду <tag> є закриває мітка вигляду </ tag> з тим же ім'ям, але з додаванням знака слеш - «/». Багато міток, крім імені, можуть містити атрибути - елементи, що дають додаткову інформацію про те, як browsers повинен обробити поточну мітку. У найпростіших HTML документах таких атрибутів немає.

Таким чином, мітка або тег HTML складається з наступних один за одним в певному порядку елементів:

- лівої кутової дужки "<" (такого ж, як символ «менш ніж»);

- слеша «/», який означає, що тег є кінцевим, що закриває деяку структуру, тобто в цьому контексті ви можете читати символ «/», як кінець.

- імені тега, наприклад TITLE або PRE

- необов'язкових, якщо навіть тег може мати їх, атрибути.

Тег може бути без атрибутів або супроводжуватися одним або декількома атрибутами, наприклад: ALIGN = CENTER

- правої кутової дужки «>» (такий же, як символ «більше ніж»), які відкривають <...>, і закривають </ ...> дії які відповідають даним командам. Це можуть бути команди створення будь-яких елементів сторінки, як, наприклад вбудована таблиця або зображення. Текст, укладений між тегами, при виведенні на екран за допомогою browsers підпорядковується правилам притаманним для даних тегів (як, наприклад розмір або колір). Деякі з тегів не вимагають закриття, і припиняють свою дію з появою аналогічних команд які їх відкривають. (Наприклад, рядок в списку переривається там, де дана команда створення наступного рядка списку.) Однак, «зайвий» тег який закриває не зашкодить. Browsers всі незрозумілі для нього команди пропускає повз, що вельми важливо для недосвідчених творців HTML документів. Тому, краще закривати теги - це полегшить читання документів.

Теги можуть бути вкладені один в одного, для надання тексту відразу декількох властивостей. В цьому випадку, *порядок закриття тегів повинен бути строго протилежний порядку їх відкривання.* Наприклад, якщо ви спочатку виділили текст жирним шрифтом, а потім ще й курсивом, то закривати слід спочатку внутрішній тег (курсив) і тільки потім жирний шрифт.

Наприклад:

<B> Вищеописаний <I> текст </ I> </ B> <I> буде мати такий вигляд. </ I>

При перегляді в browsers матимемо:

Вищеописаний *текст* *буде мати такий вигляд*.

HTML не є чутливою до клавіатури, тобто якими буквами набрані команди мови: <html> сприймається аналогічно <HTML> або <hTmL>. Виняток становлять спеціальні символи.

Обов'язкові структурні елементи HTML-документа
<Html> ... </ html>

Тег <html> повинен відкривати HTML-документ. Аналогічно , тег </ html> повинен завершувати HTML-документ.

«Голова» документа: <head> ... </ head>

Ця пара тегів вказує на початок і кінець заголовку документа. Крім найменування документа (див. Опис тега <title> нижче), в цей розділ може включатися безліч службової інформації.

<Title> ... </ title>

Все, що знаходиться між мітками <title> і </ title> , тлумачиться browsers як назва документа. *Netscape Navigator* , наприклад, показує назву поточного документа в заголовку вікна і друкує його в лівому верхньому куті кожної сторінки при виведенні на принтер.

Назва - необов'язкова частина в документі, але крім відображення в заголовку browsers імені сторінки, може включати необмежену кількість дуже корисних META-інструкцій. Зазвичай вони розташовуються між двома першими мітками <head> і <title>. META-інструкція це стандартний опис теми документа (для пошукових систем) або ж пряма вказівка для browsers.

Приклад:

<META HTTP-EQUIV = "Content-Type" CONTENT = "text / html"> - інструкція дає вказівку браузеру інтерпретувати документ як HTML-текст.

<META HTTP-EQUI V = "Refresh" CONTENT = "17; URL = http: //www.abcd.ru"> (або URL = music.mid ">) - Така інструкція через 17 секунд почне завантаження вузла з указаним URL (або почне відтворення файлу music.mid в звуковому форматі, якщо той підтримується browsers.)

Простір між мітками що закриваються </ title> і </ head> часто використовується для зберігання операторів JavaScript і VBScript використовують глобальні змінні і функції, а також при впровадженні Каскадних таблиць стилів (команда, вставлена між тегами </ title> і < / head>. У ній описані елементи сторінки, які мають однаковий вигляд, тобто до яких будуть застосовані певні стильові рішення) або для оголошення, наприклад, нестандартного розміру шрифту сторінки за допомогою тега <basefont size = "5" >

Примітка: Рекомендується назва не довше 64 символів.

«Тіло» документа: <body> ... </ body>

Ця пара тегів вказує на початок і кінець **тіла** HTML-документа, яке власне, і визначає зміст документа.

Основний текст сторінки знаходиться після необов'язкового заголовка, між так же необов'язковими мітками: **<BODY>** ... тіло сторінки ... **</ BODY>** . (Сучасні браузери самі розпізнають, де кінчається назва і починається тіло документа ...). Якщо елемент BODY не містить атрибути, які розміщені всередині мітки, використання його не дає явного ефекту в безпосередньому

відображенні документа. Даний тег можна використовувати для завдання кольору або фонового малюнка для сторінки.

Атрибути використовуються в тегах <body>

BGCOLOR - визначає колір фону документа

TEXT - застосовується безпосередньо для визначення кольору тексту документа

LINK -використовується для кольору невідвіданого гіпертекстового зв'язку, тобто визначає колір виділеного елементу тексту, при натисканні на який відбувається перехід по гіпертекстовому посиланню

VLINK - застосовується для кольору відвіданого гіпертекстового зв'язку, тобто визначає колір посилання на документ, який вже був переглянутий раніше

ALINK - даний атрибут використовується для кольору активного гіпертекстового звязку, тобто використовується для виділення тексту при натисканні

BACKGROUND - використовується для URL фонового образу

BOTTOMMARGIN - встановлює кордон нижнього поля документа в пікселях

TOPMARGIN - встановлюється кордон верхнього поля в пікселях

BGPROPERTIES - якщо встановлено значення FIXED, фонове зображення не прокручується

LEFTMARGIN - встановлює кордон лівого поля документа в пікселях

RIGHTMARGIN - встановлює кордон правого поля документа в пікселях

SCROLL - Yes / No встановлює наявність або відсутність полос прокрутки вікна браузера

Наприклад:

<BODY BGCOLOR = GREEN> - зелений колір сторінки.

Кольори елементів

Фоновий колір і колір тексту для всієї сторінки призначаються параметрами тега BODY:

<body bgcolor = "колір" text = "колір"> *... тіло сторінки ...* **</ body>**

Колір шрифту для окремих ділянок тексту (тег також може керувати розмірами шрифту):

<font color = "колір"> *... фрагмент тексту ...* **</ font>**

Колір фону таблиці: (аналогічно і для окремих осередків, тільки table змінюється на tr, td або th)

<table bgcolor = "колір"> *... тіло таблиці (осередки) ...* **</ table>**

Колір базового шрифту для всієї сторінки також можна визначити тегом:

<basefont color = "колір"> - розташованим в «голові» сторінки (частіше застосовується для установки розмірів шрифту)

Сучасні браузери (від IE 4.0.) Розпізнають 140 визначених кольорів.

Задання кольору можливо як в іменному, так і в цифровому вираженні, тобто у вигляді: <BODY BGCOLOR = tomato »або« font color = FF34C6>

Таблиця з назвами іменних кольорів представлена на рисунку 1.

antiquewhite	aqua	aquamarine
black	chartreuse	darkblue
blue	blueviolet	brown
coral	cornflowerblue	cornsilk
darkgoldenrod	darkgray	cyan
beige	darkgreen	darkkhaki
darkred	darksalmon	darkseagreen
darkturquoise	darkviolet	deeppink
floralwhite	forestgreen	fuchsia
goldenrod	indigo	lightblue
green	greenyellow	honeydew
limegreen	deepskyblue	dimgray
khaki	lavender	lavenderblush
lightcyan	lightgoldenrodyellow	lightgreen
lightskyblue	lightslategray	lightsteelblue
magenta	maroon	mediumaquamarine
mediumseagreen	mediumslateblue	mediumspringgreen
mediumorchid	lime	lightpink
maccasin	novajowhite	navy
mintcream	livedrab	palevioletred
orangered	orchid	palegoldenrod
peachpuff	peru	pink
rosybrown	royalblue	saddlebrown
powderblue	paleturquoise	snow
sienna	silver	skyblue
steelblue	tan	teal
turquoise	slategray	tomato
wheat	white	whitesmoke

Малюнок 1. Таблиця кольорів.

У цифровому вираженні кольоровість шифрується інтенсивністю трьох основних кольорів: червоного, зеленого і синього (RGB) в шістнадцятирічній системі насиченості заданого кольору одним із трьох основних кольорів в діапазоні від нуля (00) до 255 (FF). 00- нульова інтенсивність, FF- максимальна насиченість. Відповідно: 000000- чорний колір, FF0000- червоний, 00FF00 -

зелений, 0000FF - синій, FFFFFF- білий максимальної інтенсивності.Виходячи з цього правила, кольори з одинаковими значеннями всіх трьох параметрів (Red = Green = Blue) будуть сірим, але різної інтенсивності. Розберемо декілька прикладів.

bgcolor = # FFFFFF

Колір фону . Насиченість червоним, зеленим і синім однакова - FF (це шістнадцятирічне уявлення числа 255). Результат - білий колір.

t ext = # 000000

Колір тексту. Насиченість червоним, зеленим і синім однакова - 00 (нуль). Результат - чорний колір.

link = # FF0000

Колір гіпертекстового посилання. Насиченість червоним - FF (255), зеленим і синім - 00 (нуль). Результат - червоний колір.

Приклад кольорів в RGB представлений на малюнку 2 Верхня частина, колонками по три, була виставлена як приклад правильного підбору кольорів, в керівництві Артемія Лебедева.

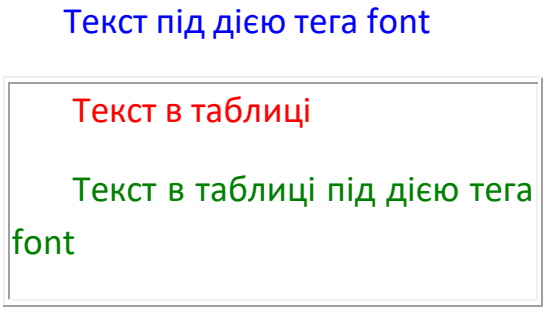
0	ffc200	003562	0069a3
2	ff0000	006d93	000000
d	000000	e8aa00	ffffff
Приклади кольорів в RGB			
d	9595c6	9999ff	009933
9	968187	cc0033	007299
c	ae88b8	960018	e96b9e
f	c699bd	999966	a9 834f
5	ff99cc	33cc66	6699cc

Малюнок 2. Приклад кольорів в RGB.

Розглянемо ієрархію колірних тегів:

<Html> <head> <basefont color = yellow> </ head>
<Body bgcolor = white text = red>
< Font color = blue >
Текст під дією тега font
<t able border> <tr> <td>
Текст в таблиці < br >
< Font color = green > Текст в таблиці під дією тега font </ font >
</ Table >
</ Font >
</ Body > </ html >

Наочно даний код буде виглядати, так як показано на малюнку 3



Малюнок 3. Вид ієрархії колірних тегів в HTML –документі.

Жовтий колір, заданий тегом basefont перепризначувався параметром text тега body на червоний (тег <basefont> проте залишається актуальним через можливості керувати розміром шрифта, що неможливо для тега BODY.). Будь-який текст, введений зараз без додаткових колірних тегів, буде відображений червоним. Але в запропонованому прикладі такий тег є: - <font color = blue> . Він забарвлює верхній рядок прикладу в блакитний колір. В межі тега потрапляє і таблиця, але на текст який знаходиться всередині таблиці дія цього тегу не розповсюджується! Таким чином, текст таблиці буде виведений кольором, зазначеним в параметрі text (червоним). Якщо ж ми застосуємо для тексту всередині таблиці свій власний тег <font color = green> то він буде справно працювати.

Примітка: текст всередині таблиці слід наповнювати власними тегами <FONT> (це стосується і розмірів).

Крім того, тег <BODY> може містити атрибут background = "[ім'я файлу]" , який задає зображення, що служить фоном для тексту та інших зображень. Як і будь-яке інше зображення, фон повинен бути представлений в форматі GIF (файл з розширенням * .gif) або JPEG (файл з розширенням * .jpg або * .jpeg). Браузери заповнюють множинними копіями зображення-фону весь простір вікна, в якому відкрито документ, подібно до того, як при будівництві великий простір стін покривають маленькими (і однаковими) плитками. Важливо відзначити, що колір фону і зображення-фон ніяк не відображаються на папері, якщо розпечатати HTML-документ. З цього витікає важливий практичний наслідок: **намагайтеся не використовувати текст білого кольору** .

Отже, **<body background = Image.gif>** - команда для використання фонового малюнка, припустимо з імя’м Image.gif

Якщо сторінка займає місця більше ніж вікно браузера, то, здвигаючи смугу прокрутки, буде здвигатися так само і фоновий малюнок. Для того щоб він не «відповзав» слід використовувати наступний синтаксис:

<body bgcolor = "red" background = "fon.jpg" bgprope r ties = "fixed">

В даному прикладі ще до завантаження фонового зображення (fon.jpg) сторінка прийме червоний колір (підбирається в тон фонової картинки, адже зображення може мати великий час завантаження, а може і зовсім загубитися при поганому зв'язку. Досить важко буде прочитати блідо жовтий текст на білому тлі.), при пропусканню фоновий малюнок буде стояти на місці (bgproperties = fixed) а текст з картинками буде повзти по нім, як титри в кінофільмі.

Примітка : подвійні лапки, так само необов'язкові для сучасних браузерів (а в деяких випадках навіть шкідливі у зв'язку з марним збільшенням розміру сторінки). Їх використання необхідно, в тому випадку, коли значення параметрів складається з декількох слів розділених пробілами або є певним виразом як **style.**

Приклад текстового і числового значення кольору представлений в таблиці 1.

Таблиця 1. Співвідношення текстового і числового значень кольорів.

колір	**текстове значення**	**числове значення**
1	2	3
чорний	Black	# 000000
Темно синій	Navy	# 000080
синій	Blue	# 0000FF
зелений	Green	# 008000
Синьо-зелений	Teal	# 008080
лимонний	Lime	# 00FF00
морської хвилі	Aqua	# 00FFFF
Темно-бордовий	Maroon	# 800000
фіолетовий	Purple	# 800080
оливковий	Olive	# 808000
сірий	Gray	# 808080
Срібний	Silver	# C0C0C0
червоний	Red	# FF0000
фуксин	Fuchsia	# FF00FF
жовтий	Yellow	# FFFF00
білий	White	#FFFFFF

Приклад WEB-сторінки

Виходячи з вищесказаного, приклад мінімально допустимого коду для сторінки яка правильно працює в мережі виглядає наступним чином.

<Html>
<Head>
<Meta http-equiv = Content-Type content = text / html>
< Title > Приклад </ title >
</ Head >
< Body > Місце для основного коду сторінки </ body >
</ Html >

Заголовки

Для виділення заголовків в тексті існує тег <розмір> *... заголовок ...* </ розмір> (від head- голова). Де розмір заголовка змінюється від 1 до 6 в порядку убування. Заголовок першого рівня - найбільший, шостого рівня, природньо - найдрібніший. У заголовках закриває тег обов'язковий.

< H 1> ... </ H 1> - < H 6> ... </ H6>

Наведений тег виділяє свій заголовок відступом рядків. До нього можна застосовувати атрибут align - вирівнювання:

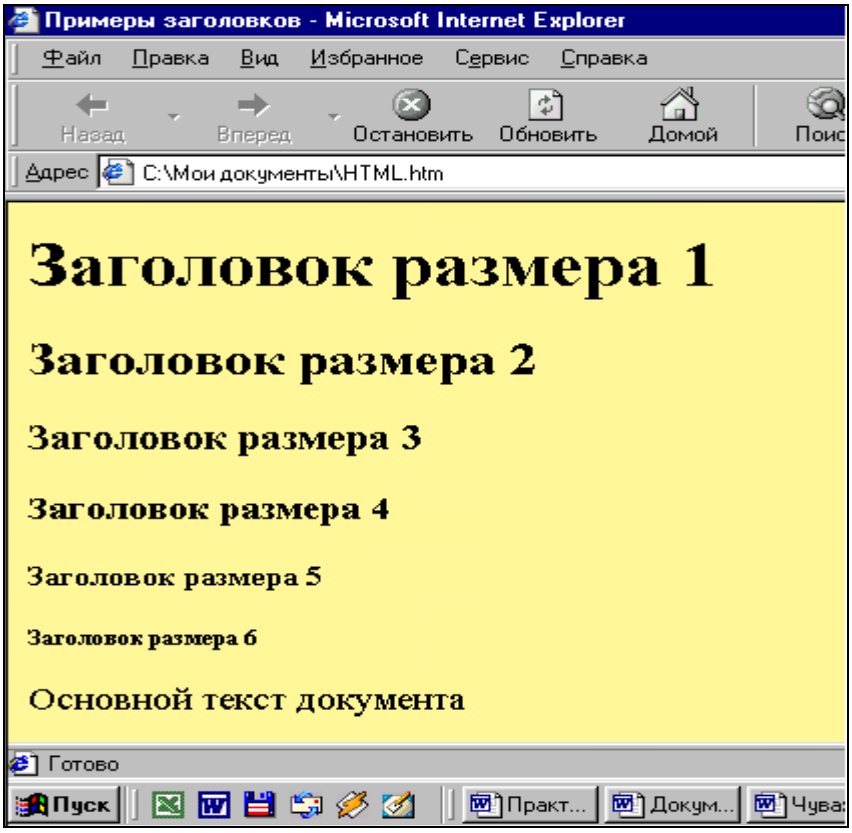
<h5 align = "right"> Заголовок п'ятого рівня **</ h5>**

При цьому заголовок документа буде вирівняний по правому краю

Приклад HTML-команди:

```
< HTML >
< HEAD >
<TITLE> Приклади заголовків </ TITLE>
</ HEAD>
<BODY BGCOLOR = fff99d>
<H1> Тема розміру 1 </ H1>
<H2> Тема розміру 2 </ H2>
<H3> Заголовок розміру 3 </ H3>
<H4> Заголовок розміру 4 </ H4>
<H5> Заголовок розміру 5 </ H5>
<H6> Тема розміру 6 </ H6>
Основний текст документа
</ BODY>
</ HTML>
```

Наочний вид прикладу представлений на малюнку 4.



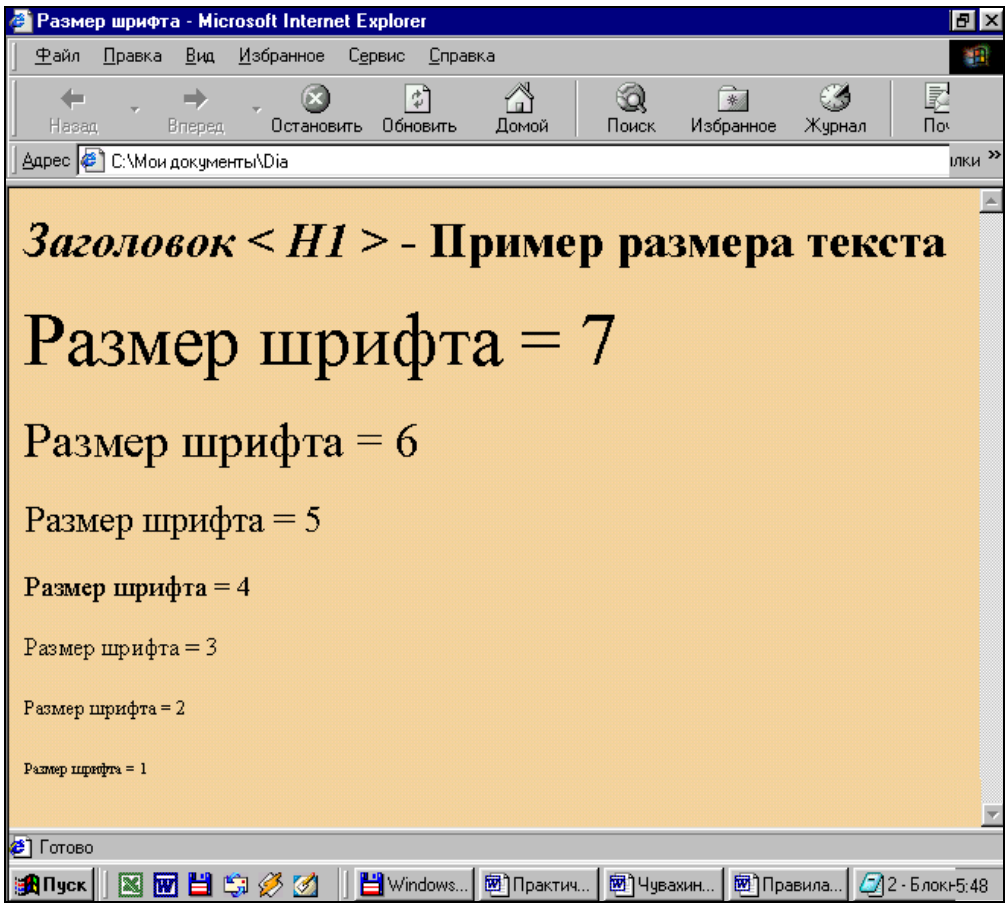
Малюнок 4. Наочний вид заголовків HTML – сторінки.

Розмір шрифту

Розмір шрифту в іншому тексті задається за допомогою команд <font size = "n"> ... </ font> Теги можуть бути вкладені одна в одну в незліченній кількості. Однак при цьому необхідно їх правильно закривати! Значення для ***n*** можуть лежати в межах від 1 до 7 по зростанню розміру шрифту (протилежно розмірам заголовків). Наприклад, наведений нижче код дає на екрані наступне (малюнок 5).

```
< Html >
< Head >
<title> Розмір шрифту </ title>
</ Head>
< Body Bgcolor = fad 69 f >
<h1> <i> Тема <H1> </ i> - Приклад розміру тексту </ h1>
<p> <font size = "7"> Розмір шрифту = 7 </ font>
  <p> <font size = "6"> Розмір шрифту = 6 </ font>
  <p> <font size = "5"> Розмір шрифту = 5 </ font>
  <p> <font size = "4"> Розмір шрифту = 4 </ font>
  <p> <font size = "3"> Розмір шрифту = 3 </ font>
  <p> <font size = "2"> Розмір шрифту = 2 </ font>
  <p> <font size = "1"> Розмір шрифту = 1 </ font>
```

</ Body> </ html>



Малюнок 5. Наочний приклад розміру шрифту тексту.

Також можливе завдання розміру шрифту щодо шрифту «за замовчуванням» у вигляді: **size = + 2** або **size = -1**. Звичайний експлоереровский розмір це Залеу кожного користувача можуть бути і свої налаштування. Розмір шрифту для конкретної сторінки можна задати тегом **<basefont size = 5>** - Розташування в «голові» сторінки, для установки даного розміру відповідно рівному 5.

Для тексту що знаходиться в кожній з комірок таблиці слід застосовувати власний тег <FONT> (це стосується і кольору шрифта)

Розміщення тексту буває і зовсім без міток, в цьому випадку стає неможливим його вирівнювання (що в принципі буває і непотрібно), а шрифт приймає розмір відповідний <font size = "3">.

Колір шрифту задається параметром color = "колір" наприклад: **<font color = green size = + 2>** Розмір шрифту = 5 **</ font>** видасть той же рядок що і в попередньому прикладі тільки зелений. При цьому черговість в завданні параметрів будь-якого тега неважлива.

Розмір, тип і колір основного шрифту можна встановити за замовчуванням тегом < **BASEFONT** > - базовий шрифт

Basefont встановлює основний розмір шрифту, який застосовується до звичайного і попередньо відформатованого тексту, але не до заголовків, за винятком тих, які модифікуються з використанням елемента FONT із зазначеним відносним розміром шрифту (наприклад, FONT SIZE = + 1), а також тип і колір шрифту. Дія цього елемента розповсюджується не на все. В Netscape, наприклад, BASEFONT не впливає на розмір шрифту в межах таблиці. *Таким чином, що би встановити розмір шрифту в межах таблиці, необхідно вставити елемент зміни шрифту в кожну клітинку!*

Даний тег використовує такі атрибути:

SIZE - розмір шрифту (від 1 до 7)

COLOR - колір вмісту елемента FONT

FACE - тип шрифту, яким програма перегляду буде виводити текст

Таким чином, тег виглядає так:

< BASEFONT SIZE = n >

або

<BASEFONT COLOR = колірна специфікація>

або

<BASEFONT FACE = тип шрифту >

Абзац

На відміну від текстових документів переривання рядків в HTML-файлах не суттево. При перегляді HTML-джерела в текстовому редакторі обрив рядка може відбуватися в будь-якому місці, але при перегляді в браузері це переривання буде проігноровано. Замість 68 прогалин браузер покаже в тексті тільки один, якщо тільки ви не використовуєте Авторський стиль.

Розбиття документа на складові задається за допомогою таблиць, міток форматування: <p> і </ p> (від **p** aragraf) а також < br > (від **br** eak). (Крім цього можлива вставка примусових прогалин за допомогою команди: & nbsp; (**n**o**b** reak **sp** ace) для кожного пробілу). Звичайний текст рекомендується розташовувати в абзацах: <p> текст </ p>.

У цьому випадку він буде виділений новим рядком. (Закриваючий тег необов'язковий. У разі якщо його пропущено, наступний абзац почнеться з наступної мітки яка відкриває <P>, проте його застосування іноді доцільно, наприклад для переривання вирівнювання).

Редагування тексту усередині абзацу може проводиться із застосуванням Логічних і Фізичних стилів форматування. При логічному форматуванні, Ви повідомляєте browsers що бажаєте виділити фрагмент тексту. При цьому браузери різних виробників будуть виділяти цей текст на свій розсуд. При физичному форматуванні Ви самі ставите стильове рішення шрифту - жирний,

під нахилом, закреслений і т.д. в цьому випадку всі browsers будуть підкорятися Вашим наказам). Таким чином,

<P> ... </ P>

ця пара тегів описує абзац. Все, що укладено між <P> і </ P> , сприймається як один абзац.

Відступ від краю

Абзаци можуть бути виконані з відступом від краю за допомогою тегів: <dl> і </ dl>. Ці теги добре підходять для виділеня важливих абзаців з одноманітного тексту. Структура взаємодії тегів досить складна, і має кілька внутрішніх команд. В межах цих тегів наприклад, не повинно бути тегів <p>.

Примітка: якщо виділеного абзацу не виходить, то треба вставити після тега який відкривається команду «Червоного рядка»: <dd> який потребує закриття.

Вирівнювання

Велике значення в HTML має можливість вирівнювання. Наприклад, наступний рядок: <P ALIGN = "CENTER"> Вирівнювання по центру </ P> видасть на екрані:

Для align (читається «Елайн», від англійського «вирівнювати») допустимі значення: «center» (вирівнювання по центру), «right» (вирівнювання по правому краю) і «justify» (рівномірний розподіл по всій довжині рядка **тільки для тексту**). За замовчуванням, браузер сам рівняє всі елементи по лівому краю. Вирівнювання може застосовуватися до тегів: <p>, <table>, до картинок і заголовків.

Так само в HTML існує спеціальний тег вирівнювання: ... </ center>. Все, що розташоване між цими мітками буде вирівняно, так само як і по команді <p align = "center">

Тобто теги <Hi> і <P> можуть містити додатковий атрибут ALIGN , наприклад:

<H1 ALIGN = CENTER> Вирівнювання заголовка по центру </ H1>

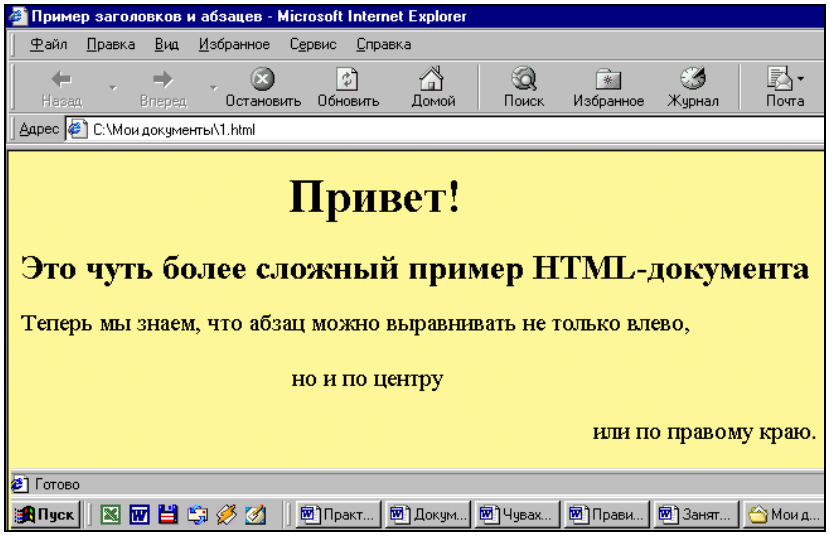
або

<P ALIGN = RIGHT> Зразок абзацу з вирівнюванням по правому краю </ P>

Все вищеописане можна побачити на наступному прикладі:

```
< Html >
< Head >
< Title > Приклад заголовків і абзаців </ title >
</ Head>
<body BGCOLOR = fff99d>
<H1 ALIGN = CENTER> Привіт ! </ H1>
<H2> Це трохи складніший приклад HTML-документа </ H2>
<P> Тепер ми знаємо, що абзац можна вирівнювати не тільки вліво, </ P>
```

<P ALIGN = CENTER> але і по центру </ P> <P ALIGN = RIGHT> або по правому краю. </ P>
</ Body>
</ Html>
Наочний вид прикладу представлений на малюнку 6.



Малюнок 6. Приклад HTML -сторінки з заголовками, абзацами і вирівнюванням.

3. ПОРЯДОК ВИКОНАННЯ РОБОТИ

• Ознайомитися з основними теоретичними положеннями;

• Виконати завдання

- Створіть HTML-документ з наступним заголовком: «Це моя перша Web-сторінка». Документ повинен спиратися тільки на основні елементи і виводити наступний текст: « **Сленг** - це слова, які часто розглядаються як порушення норм стандартної мови. Це дуже виразні, іронічні слова, які послуговують для позначення предметів, про які говорять в повсякденному житті ».

Примітка : для створення Web -сторінки необхідно створити документ з розширенням файлу * .txt (наприклад, Student .txt) з мінімальним кодом (див. Приклад 1). Потім поміняйте розширення файлу * .txt на * .htm.Закрийте його.

- Клацніть мишею по значку створеної вищеописаним способом WEB-сторінки. Після її відкриття поставте курсор миші в будь-яку точку сторінки яка відкрилася в Explorere і клацніть правою кнопкою. У меню виберете «перегляд вHTML ».

Відредагуйте текст, у вікні текстового редактора, додавши в нього наступний абзац: « Сленг складається зі слів і фразеологізмів, які виникли і спочатку вживалися в окремих соціальних групах і відображав цілісну орієнтацію цих груп ». Текст абзацу повинен бути темно-синім або темно-

зеленим. Для фонового малюнка використовуйте будь який графичний файл у форматі JPG або GIF .

Закрийте редактор.

На питання про збереження дайте відповідь позитивно.

У Explorere клацніть «оновити».

• Створити WEB-сторінку використовуючи матеріал інформаційного блоку практичної роботи № 4. Найважливішу інформацію виділіть іншим кольором. Текст документа зробіть темно-синім або темно-зеленим кольором. Фоновий малюнок повинен бути нейтральним.

• Відповісти на контрольні питання.

4. Контрольні питання

- Що таке HTML-документ? Що являє собою мова HTML?
- З яких частин складаються теги?
- Які обов'язкові теги повинен включати в себе будь-який HTML-документ?
- Для чого застосовуються теги <HEAD> і <TITLE>?
- З чого може складатися тіло HTML-документа?
- Які атрибути використовуються в тезі <body>?
- Якими тегами задається колір елемента? Які атрибути він може включати?
- Що таке метаінформація? Для чого вона використовується?
- Для чого застосовуються заголовки в HTML-документах? Якими тегами вони описуються?
- За допомогою, яких команд задається розмір шрифту в тексті?
- Яким параметром задається колір шрифту в документі?
- Яким тегом задається розмір, тип і колір основного шрифта? Які атрибути при цьому використовуються?
- Яким чином в HTML-документах встановлюються абзаци в тексті?
- Яким чином в HTML-документах задається вирівнювання текста?

Література

Основна література

1. Мазурок Т.Л. Системи управління навчанням: навчальний посібник. – Одеса: ПНПУ ім. К.Д. Ушинського, 2013. 160 с. (Протокол №8 28.03.2013 Вченої ради ун-ту).

2. Плотніков В.М., Мазурок Т.Л. Некласичні логіки нечітких даних: навч. посібн. Одеса: ОДАХ, 2011. 182 с. (Гриф МОН №1/11-4518 від 03.06.2011).

3. Лозович О.М., Мазурок Т.Л. Системи штучного інтелекту: посібник до виконання лабораторних робіт. Одеса: ОДАХ, 2011. 100 с.

4. Мазурок Т.Л., Селіванова А.В. Системи штучного інтелекту: посібник до виконання контрольних робіт. Одеса: ОДАХ, 2008. 50 с.

5. Гаврилова Т.А., Хорошевский В.Ф. Базы знаний интеллектуальных систем. СПб.: Питер, 200. 384 с.

6. Гаврилова Т.А., Червинская К.Р. Извлечение и структурирование знаний для экспертных систем. М.: Радио и связь, 1992. 200 с.

7. Пасічник В.В. Організація баз даних та знань: підручник для студентів ВНЗ. – Київ.: BHV, 2006. 384 с.

8. Руденко О.Г. Штучні нейронні мережі: навчальний посібник. Харків.: Компанія СМІТ, 2006. 404 с.

Допоміжна література

1. Люггер Дж.Ф. Искусственный интеллект. Стратегии и методы решения сложных проблем. Москва.: Вильямс, 2003. 864 с.

2. Башмаков А.И., Башмаков И.А. Разработка компьютерных учебников и обучающих систем. Москва.: Филин, 2003. 616 с.

3. Когаловский М.Р. Перспективные технологии информационных систем. М.: ДМК Пресс, 2003. 288 с.

4. Башмаков А.И., Башмаков И.А. Интеллектуальные информационные технологии: Учебное пособие. Москва.: Изд-во МГТУ им. Н.Э. Баумана, 2005. 304 с.

5. Беспалько В.П. Образование и обучение с участием компьютеров (педагогика третьего тысячелетия). Москва.: МПСИ, 2002. 325 с.

6. Информатизация образования: направления, средства, технологии / под общ. ред. С.И. Маслова. Москва.: Издательство МЭИ, 2004. 868 с.

7. Атанов Г.А. Обучение и искусственный интеллект, или основы современной дидактики высшей школы. Донецк: Изд-во ДОУ, 2004. 504 с.

8. Анфилатов В.С., Емельянов А.А., Кукушкин А.А. Системный анализ в управлении. М.: Финансы и статистика, 2003. 368 с.

9. Згуровський М.З., Панкратова Н.Д. Основи системного аналізу. Київ.: BHV, 2007. 544 с.

10. Гладун В.П. Партнёрство с комп’ютером. Київ.: Port-Royal, 2000. 128 с.

11. Когнитивное управление в интеллектуальных обучающих системах / А.Ф. Верлань, М.Ф. Ус, А.В. Пискун, В.А. Федорчук; под ред. А.Ф. Верлань. – Черкассы: ЧИУ, 2002. 104 с.

12. Рассел С. Искусственный интеллект. Современный поход М.: Изд.дом «Вильямс», 2007. 1408 с.

13. Пупков К.А. Интеллектуальные системы. Москва.: МГТУ им. Н.Э. Баумана, 2003. 348 с.

14. Хайкин С. Нейронные сети: полный курс. Мінськ.: Изд.дом «Вильямс», 2006. 1104 с.

15. Курейчик В.М. Теория и практика эволюционного моделирования. Таганрог, 2003. 432 с.

16. Рутковская Д. Нейронные сети, генетические алгоритмы и нечёткие системы. Мінськ.: Горячая линия. Телеком, 2006. 452 с.

17. Чалий О.В. Синергетичні принципи освіти та науки / О.В. Чалий. – К.: АТ «Випол», 2000. – 253 с.

18. Белова Л.А. Логико-математические основы управления учебными процессами высших учебных заведений: Монография. Харьков: Восточно-региональный центр гуманитарно-образовательных инициатив, 2001. 272 с.

19. Проектування інформаційних систем: Посібник / За ред. В.С. Пономаренка. К.: Видавничий центр «Академія», 2002. 488 с.

20. Лысенко Ю.Г. нейронные сети и генетические алгоритмы: учебное пособие / Ю.Г. Лысенко, Иванов Н.Н., Минц А.Ю. Донецк: ООО «Юго-Восток, Лтд», 2003. 265 с.

21. Гнатієнко Г.М. Експертні технології прийняття рішень: Монографія / Г.М. Гнатієнко, В.Є. Снитюк. К.: ТОВ «Маклаут», 2008. 444 с.

Інформаційні ресурси в Інтернеті

1. Електронний каталог бібліотеки ПНПУ ім. К.Д. Ушинського. URL: https://unilib.library.pdpu.edu.ua/search.php (дата звернення 27.08.2020).

2. URL: http://aied.inf.ed.ac.uk/ - International Journal of Artificial Intelligence in Education (IJAIED) (дата звернення 27.08.2020).

3. URL: http://aied.inf.ed.ac.uk/aiedsoc.html - International Artificial Intelligence in Education Society (дата звернення 27.08.2020).

4. URL: http://ifets.ieee.org / - образовательные технологии и общество (дата звернення 27.08.2020).

5. URL: http://www.aaai.org/ - International Artificial Intelligence in Education Society (дата звернення 27.08.2020).