15. Трифонов Ю.В., Плеханова Л.Ф., Юрлов Ф.Ф. Выбор эффективных решений в экономике в условиях неопределенности: Монография. – Н. Новгород: Изд-во ННГУ, 1998. – 140 с. 16. Управление проектами. Справочник для профессионалов: Под ред. И.И. Мазура и В.Д. Шапиро. – 2001. 17. Ципес Г., Товб А. Не говори красиво, говори правильно... или Глоссарий управления проектами // Директор ИС. – 2002. – № 4. http://www.osp.ru/cio/2002/04/. 18. ISO/TR 10006: 1997 (E). Quality Management – Guidelines to quality in project management. 19. PMBOK. A Guide to the Project Management Body of Knowledge. PMI Standards Committee. 1996 Edition. 20. Wideman Comparative Glossary of Project Management Terms. PMForum, 2000. http://www.maxwideman.com/.

УДК 681.3

В.А. Висоцька
Національний університет “Львівська політехніка”,
кафедра “Інформаційні системи та мережі”

СИСТЕМА ОПРАЦЮВАННЯ СТРУКТУРИ ЕЛЕКТРОННОГО ПІДРУЧНИКА

The problems of e-teaching systems based on the Intranet technologies development are considereded in this paper.

Розглядаються проблеми створення систем електронного навчання на основі застосування Intranet технологій.

ВСТУП. ЗАГАЛЬНА ПОСТАНОВКА ПРОБЛЕМИ

Світ вступає в епоху інформаційного суспільства. Необхідність сприйняття великої кількості нових знань і умінь, опрацювання великих обсягів інформації, безперервного підвищення кваліфікації сприяє переосмисленню існуючої системи освіти. Формується новий вид освіти, у якому інформаційні технології відіграють системоутворюючу, інтегруючу роль, а відкриття доступу до нових мережевоцентричних технологій навчання і джерел інформації стає визначальним. Усе це викликає підвищений інтерес до дистанційної (мережевоцентричної) освіти (ДО) [1].

За останні десятиліття істотно збільшилися обсяги навчальних матеріалів, які ще значно ускладнилися. При цьому в багатьох навчальних закладах спостерігається нестача висококваліфікованих викладацьких кадрів. Великі ускладнення часто виникають при оперативному опрацюванні, виготовленні і поширенні навчальних посібників різноманітних видів. Зазначені чинники негативно позначаються на якості підготовки тих, кого навчають. У зв’язку з цим велика увага приділяється застосуванню прогресивних методик навчання, зокрема в дистанційному навчанні з використанням обчислювальної техніки [1–3].

Розвиток навчальних систем сьогодні йде в напрямку надання їм властивості адаптації до цілей та умов навчання. У багатьох випадках розроблювачу автоматизованої навчальної системи потрібно наочно уявити її структуру не тільки в загальному вигляді, із точністю в кращому випадку до цілої теми, як це дозволяє зробити більшість систем, але і більш

конкретно, із деталізацією до більш дрібних структур, таких як визначення, теореми, алгоритми й ін. Це дозволить розроблювачу побачити можливі недоробки, неповноту матеріалу, відсутність яких-небудь проміжних елементів, необхідних для логічного зв'язку понять [8–12].

За даною структурою відразу можна буде побачити базові поняття, що є основними для даного підручника, взнати які необхідно ще перед початком процесу навчання. За такою структурою можна легко перевірити послідовність подачі користувачу матеріалу, коректність уведених визначень. Наявність подібної структури може послужити відправною точкою для побудови інтелектуальної системи навчання, що дозволяє залежно від рівня знань користувача вказувати оптимальний шлях навчання і контролювати засвоєні знання, виробити рекомендації щодо зміни плану навчального процесу. Все це загалом дозволить удосконалити цикл навчання і зменшити тимчасові витрати на навчання.

Реалізація перерахованих вище можливостей послужила основою для розробки моделі системи формування й опрацювання структури електронного підручника. За основу був узятий гіпертекстовий підручник, написаний на мові HTML [1–3].

ОГЛЯД ЗАСОБІВ СТВОРЕННЯ НАВЧАЛЬНИХ ПРОГРАМ ТА ФОРМУВАННЯ ВИМОГ ДО ЕЛЕКТРОННОГО ПІДРУЧНИКА

Сьогодні створена досить велика кількість автоматизованих навчальних систем і розроблено багато алгоритмів їх створення. За способом подання навчального матеріалу їх можна поділити на три основні види – у вигляді простих, мультимедійних або гіпертекстових документів [1].

Розглянемо деякі засоби створення гіпертекстових систем.

Довідкова система ОС Windows. Один із підходів полягає у створенні структури даних на основі довідкової системи Windows. Цей підхід має декілька очевидних переваг, головна із яких – вже реалізована навігаційна система, що містить систему пошуку за ключовими словами, автоматичне створення глосарію, можливість виведення документів на друк. Файли довідкової системи можуть містити як форматований текст, так і графіку, і анімацію.

Проте створення таких файлів потребує спеціального програмного забезпечення, за допомогою якого провадиться процес компіляції, самі файли довідника не можуть бути змінені "миттєво" – для цього потрібен компілятор. Файли довідки не можуть містити програмних елементів, довідкова система не містить ніякої внутрішньої мови для їхнього створення. Але замість цього існує засіб, за допомогою якого ми можемо запускати файли, що виконуються, які знаходяться на жорсткому диску локального комп'ютера. Але існує деяка роз'єднаність текстового матеріалу і навчальних (або тестувальних) програм.

Головним недоліком використання довідкової системи Windows є неможливість її модифікації, неможливість зміни інтерфейсу. Вікно перегляду підручника є вмонтованим в операційну систему об'єктом, внести зміни в його навігаційний механізм неможливо.

Пакет ГіперМетод. Система розробки Пакет ГіперМетод – інструмент для створення електронних каталогів, підручників і рекламних видань на CD-дисках, систем допомоги і публікацій у Internet, а також інших мультимедіа-додатків і електронних видань.

ГіперМетод дозволяє створювати гарні і складні мультимедіа-додатки, що відповідають найсучаснішим стандартам, поєднуючи в одне ціле звук, відео, малюнки, анімацію, текст і гіпертекст.

Стандартний варіант пакета містить усього два модулі – так званий монтажний стіл, призначений для загального дизайну і перегляду додатка, і програму перегляду, що являє собою той же монтажний стіл без елементів редагування.

Фаховий варіант пакета доповнений такими модулями:

- асистент за зв'язками – створює гіпертекстові зв'язки автоматично за заданими розроблювачем правилами;
- асистент за текстами – автоматично генерує гіпертексти з великих текстів;
- асистент за структурою – допомагає перевіряти структуру розроблюваного додатка;
- асистент за настановою – автоматично створює дистрибутив мультимедіа CD ROM додатка.

Очевидно, що цей пакет більш орієнтований на розробку мультимедіа-додатків і не є спеціалізованим засобом для створення навчальних систем. Хоча в ньому є деякі можливості, необхідні при розробці навчальних систем, наприклад, можливість аналізу структури, автоматичне генерування гіпертекстів і зв'язків, але відсутність таких можливостей, як можливість вставки тестувальних програм, і аналізу їхніх результатів роблять неможливою розробку якісної навчальної системи за його допомогою.

ФОРМУЛЮВАННЯ ВИМОГ ДО ПІДРУЧНИКА

Отже, можна зазначити відсутність або недостатню розвиненість у всіх розглянутих системах деяких засобів, дуже важливих і корисних для розроблювачів і користувачів автоматизованої навчальної системи. Можна сформулювати список можливостей, які мають бути в автоматизованій навчальній системі.

Для користувачів:

1. Організація навчання різного рівня – від початкового знайомства до докладного засвоєння матеріалу.
2. Можливість надання матеріалу, виходячи з цілі навчання.
3. Компонування матеріалу за результатами тестових перевірок.

Для розроблювачів:

1. Перевірка коректності введених визначень.
2. Формування списку невизначених понять.
3. Побудова для виділених понять (і для всього підручника) графа зв'язку з визначальними поняттями – ієрархічний граф понять або І/АБО-графа залежно від самих виділених понять.

Графи є зручним способом графічного подання знань. Особливий акцент при цьому робиться на зв'язках між різними інформаційними одиницями та між різними фрагментами знань. Важливим є те, що вся інформація про це поняття групується навколо вузла мережі, який відповідає цьому поняттю.

Мережу понять можна неформально уявляти у вигляді графа, вершини якого, як правило, позначають об'єкти предметної області, а дуги відповідають зв'язкам між ними.

Це визначення є неформальним і орієнтоване скоріше на людське розуміння, ніж на проектування і створення систем штучного інтелекту. Остання задача вимагає більш формальних визначень [1]. *Формально ієрархічний граф визначається як набір <V, E>. Тут V – множина інформаційних одиниць, E – зв'язки між інформаційними одиницями.*

На відміну від неформального визначення, останнє вимагає жорсткої фіксації схеми бази знань, тобто набору можливих інформаційних одиниць і типів можливих зв'язків.

Існує значна кількість моделей знань на основі ієрархічних графів (рис. 1).

У статті описано реалізацію другої частини цих можливостей, що є завданням розроблювачів навчальних систем. Але на основі даного проекту з деякими доопрацюваннями можна реалізувати частину можливостей, що висуваються до користувачів.

Рис. 1. Приклад орієнтованого графа

РОЗРОБКА АЛГОРИТМІВ

Модель системи виділення й опрацювання структури електронного підручника (далі “система”) була розроблена автором під керівництвом доцента, канд. техн. наук А.Ю. Берка для опрацювання гіпертекстового підручника, реалізованого на мові HTML [1]. У ній були реалізовані принципи, викладені у попередньому розділі [2, 3].

Система має виконувати такі основні функції:

1. Побудова структури понять електронного гіпертекстового підручника.
2. Відображення отриманої структури в наочному і зручному для користувача вигляді:
- пошук елемента в структурі;
- можливість переходу від перегляду структури до перегляду підручника;
3. Опрацювання отриманої структури:
- перевірка коректності визначень у структурі;
- виділення списку вихідних (невизначених) понять;
- виділення підструктури за заданою множиною понять.

Подання структури електронного підручника у вигляді І/АБО-графів

Подання структури електронного підручника у вигляді *І/АБО-графів* найкраще пристосоване до тематик (задач), які можна розбити на взаємно незалежні підзадачі. Розбиття на підзадачі дає переваги у тому випадку, коли підзадачі є взаємно незалежними, а отже, розглядати їх можна незалежно одну від однієї. Прикладами таких задач можуть бути: символічне інтегрування, ігрові задачі, доведення теорем тощо [4–6].

І/АБО-граф – направлений граф, вершини якого відповідають задачам, а дуги – відношенням між задачами. Між дугами також можуть встановлюватись відношення. Це відношення *І* або *АБО*, залежно від того, чи повинні ми розглядати лише одну із задач-нащадків, чи усі. З вершин можуть виходити дуги, які знаходяться у відношенні І разом з дугами, що між собою перебувають у відношенні *АБО*.

Вершину, з якої виходить лише *І* дуга, називають *І-вершиною*. Вершина, з якої виходить лише *АБО* дуга, називають *АБО-вершиною*.

Цільові вершини у випадку *І/АБО-графа* – тривіальні, або примітивні задачі, які не потребують подальшої деталізації.

Розв’язанням задачі, поданої у вигляді *І/АБО-графа*, є включення щонайменше усіх підзадач І-вершин, тобто шлях стає деревом. Таке дерево розв’язання визначається так (рис. 2):

- Вихідна задача *P* – корінь дерева *T*;

- Якщо *P* є *АБО-вершиною*, то в *T* міститься тільки один з її нащадків (з *І/АБО-графа*) разом зі своїм власним деревом розв'язання;
- Якщо *P* – це *І-вершина*, то усі її нащадки (з *І/АБО-графа*) разом з своїми деревами розв'язання містяться в *T*.

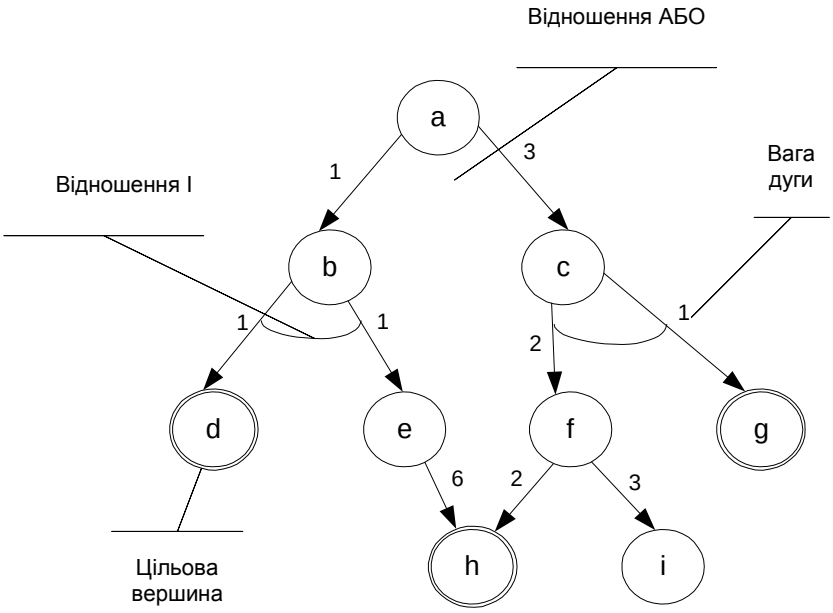

Рис. 2. Приклад І/АБО-графа

Методи пошуку у І/АБО-графі

Для пошуку в *І/АБО-графах* використовують два методи: в глибину та еврістичний пошук.

Процедура пошуку в глибину є незмінною:

1) якщо досліджувана вершина – цільова, то це і є розв'язок;

2) якщо ця вершина має *АБО* нащадків, то потрібно ров'язати одну з відповідних задач нащадків; якщо вершина має *І* нащадків, то потрібно розв'язати всі задачі нащадків.

Для *І/АБО* графів використовується також процедура еврістичного пошуку. Якщо стратегія пошуку в глибину передбачає систематичний перегляд графа і знаходження всіх розв'язків, то стратегія еврістичного пошуку передбачає знаходження лише одного розв'язку. Гарантується, що цей розв'язок є найдешевший при умові, що еврістична функція, що використовується, є нижньою межею реальної вартості дерева.

Вартість дерева розв'язання визначається як сума вартостей усіх дуг. Мета оптимізації – знайти розв'язок з мінімальною вартістю.

Вартістю вершини буде вартість оптимального дерева розв'язання цієї вершини. Вартість вершини, визначена таким чином, характеризує вартість задачі.

Будемо вважати, що вартість вершини *І/АБО-графа* можна визначити і за допомогою відповідної еврістичної функції *h*. Ця оцінка буде використовуватись для керування пошуком.

Позначимо через *H(B)* оцінку складності вершини *B*. Для найвищої вершини поточного дерева пошуку *H(B)* збігається з *h(B)*. З іншого боку, для оцінки внутрішньої вершини дерева пошуку не обов'язково використовувати значення *h*, оскільки існує деяка додаткова інформація про цю вершину – інформація про її нащадків. Отже, для *АБО-вершини*

$$H(B) = \min_i (c(B, B_i) + H(B_i)), \qquad (1)$$

де $c(B,B_i)$ – вартість дуги, яка проходить через B та B_i. Взяття мінімуму означає, що нам потрібно вирішити лише одну задачу, вартість якої є мінімальна.

Складність *І-вершини B* можна наближено оцінити так:

$$H(B) = \sum_i (c(B, B_i) + H(B_i)) \qquad (2)$$

Назвемо *H-оцінку* внутрішньої вершини "поверхневою" оцінкою.

Іншою еврістичною оцінкою є величина *F*, яку у термінах *H* можна визначити так. Нехай B_1 – вершина-предок вершини *B* у дереві пошуку, при чому вартість дуги, яка веде з B_1 у *B*, дорівнює $c(B_1,B)$. Тоді

$$F(B) = c(B_1, B) + H(B) \qquad (3)$$

Нехай B_1 – батьківська вершина вершини *B*, а B_1, B_2, ... – її дочірні вершини, тоді, відповідно до визначення *F* і *H*, маємо

$$F(B) = c(B_1, B) + \min_i H(B_i), \qquad \text{якщо B – АБО-вершина,} \qquad (4)$$

$$F(B) = c(B_1, B) + \sum_i H(B_i), \qquad \text{якщо B – І-вершина.} \qquad (5)$$

Хоча стартова вершина *A* не має предків, будемо вважати, що вартість віртуальної дуги, що входить до неї, дорівнює *0*. Якщо визнати, що *h* дорівнює *0* для всіх вершин *І/АБО-дерева*, то для будь-якого знайденого оптимального дерева розв'язання його вартість, тобто сума вартостей усіх дуг, дорівнює *F(A)*.

Типовою задачею, яка добре розв'язується у термінах *І/АБО-графа*, є задача пошуку маршруту.

Задача пошуку маршруту

Нехай є карта маршруту виділених понять – структура електронного підручника (рис. 3). На ній вузлами позначені головні поняття та задачі. Вказано вартість дороги з одного вузла в інший (проміжні теми, які за змістом доповнюють дане поняття):

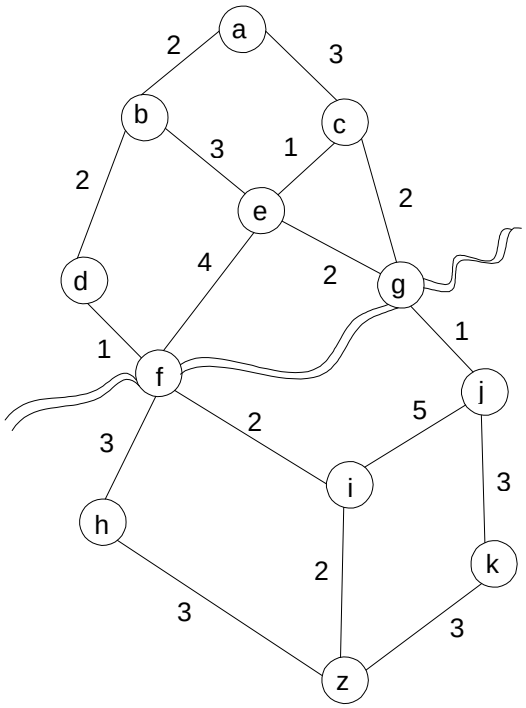

Рис. 3. Карта маршруту

І/АБО-граф подається у вигляді

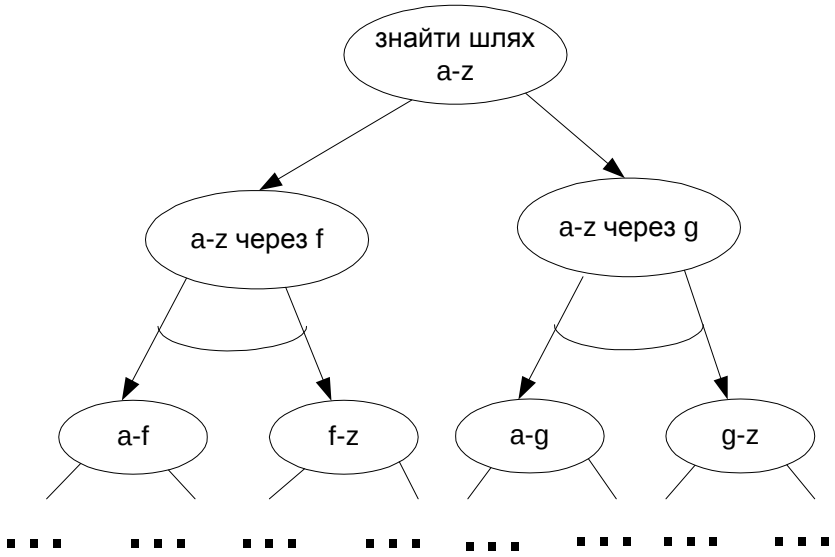

Рис. 4. Частина І/АБО-графа

ФОРМУВАННЯ СТРУКТУРИ ПІДРУЧНИКА

Електронний підручник подається у вигляді сукупності параграфів визначених типів. Ці типи – визначення, теореми, пояснення, приклади, доведення, алгоритми та ін. У цій статті описана система, яка працює з двома основними типами параграфів – із визначеннями і теоремами.

Щоби можна було формувати структуру підручника, він має бути побудований за визначеними правилами. Було розроблено такі правила побудови параграфів підручника.

Для оцінки початку і кінця параграфів було вирішено використовувати такі конструкції:

<A NAME="*мітка_початку_параграфа*"></A>

текст визначення

<A NAME="*мітка_закінчення_параграфа*"></A>

Вибір подібних конструкцій заснований на таких положеннях. По-перше, вставка даних конструкцій ніяк не відбивається на зовнішньому вигляді HTML-документа. По-друге, дані конструкції одночасно є мітками параграфів із погляду HTML, тобто, не вводячи ніяких додаткових міток, ми можемо побудувати посилання на будь-який подібно описаний параграф. По-третє, використання саме таких конструкцій полегшує побудову гіпертекстового документа, тому що багато засобів розробки гіпертекстів, наприклад, Microsoft Word, дозволяють робити в тексті закладання, що перетворяться саме в подібні теги.

Якщо в тексті визначення зустрічаються посилання на інші параграфи, вони мають бути оформлені у такому вигляді:

<A HREF=" мітка_початку_параграфа ">

Подібне оформлення параграфів дозволяє побудувати структуру понять з урахуванням усіх наявних зв’язків між ними.

Структура електронного підручника формується так. Весь процес розбитий на два етапи [1].

Перший етап – перегляд підручника й упорядкування списку всіх понять, побудованих за описаною вище схемою. Для кожного параграфа складається список усіх посилань, виявлених усередині нього, у вигляді імені сторінки разом з іменем посилання.

Другий етап – аналіз даного списку понять для побудови зв'язків між ними. Аналізуються внутрішні параграфні посилання і на їхній основі будуються зв'язки між поняттями.

Файл, що містить структуру електронного підручника, є звичайним текстовим файлом. Перший рядок файла завжди містить повний шлях до папки, у якому знаходиться оброблений електронний підручник.

Такі рядки являють собою описи вершин графа понять і мають такий формат:

ідентифікатор, ім'я, адреса, тип,

де **ідентифікатор** – унікальний номер вершини, **ім'я** – назва вершини, що збігається з обумовленим поняттям, **адреса –** відносний шлях та ім'я файла, що містить дане поняття, **тип** – числове значення, що визначає тип вершини (0 – визначення, 1 - теорема). Всі ці параметри перераховані через кому.

Закінчується частина описів вершин графа рядком, що містить слово «Edges». Після цього рядка до кінця файла йдуть списки суміжностей кожної вершини. Один рядок містить один список суміжностей, що починається з номера вершини, до якого цей список належить, і далі через кому перераховані номера суміжних із нею вершин.

Успішне функціонування системи можливе за умови, що той, якого навчають, і той, хто навчає, готові до взаємодії у новому середовищі.

Отже, для реалізації процесу дистанційного навчання на сучасному етапі важливо готувати не тільки кадровий склад, але і склад потенційних споживачів, якщо мова йде про навчальний процес у вищому навчальному закладі, тобто студентів.

Як показує практика, автоматична ефективна діяльність як викладача, так і студента у цьому середовищі неможлива. Обидві сторони учасників навчального процесу потрібно спеціально готувати, оскільки середовище має свої специфічні особливості, обумовлені природою свого виникнення на основі телекомунікацій.

Готовність/неготовність студента до умов навчання в новому середовищі виявляється в основному в його уміннях/невміннях, наявністю або відсутністю навичок роботи з комп'ютерною технікою, а також психологічної складової, необхідної для ефективної взаємодії з однолітками і викладачем не віч-на-віч, як в традиційних формах спілкування, а на відстані, тобто віртуально.

Готовність викладача до активного координування та керування у віртуальному навчальному процесі проявиться в його кваліфікаційних особливостях: досвід роботи з комп'ютерною технікою, володіння спеціальними знаннями про типи психологічного спілкування у віртуальному середовищі, а також власної психологічної готовності до нових форм взаємодії і діяльності тощо.

Отже, першими та основними умовами для створення ресурсів системи ДО є такі:

1. Забезпечення спеціальними заходами підготовки студентів до здійснення навчальної діяльності в специфічному освітньому середовищі.

2. Підготовка кадрів, здатних створювати ресурси ДО і кваліфіковано супроводжувати процес навчання.

3. На основі системного підходу і відповідно до особливостей процесу ДО вироблення вимог і принципів щодо засобів, форм, методів навчання і діяльності учасників навчального процесу.

Визначення вважається коректним, якщо воно не визначається саме через себе або через поняття, що визначаються через нього. Іншими словами, якщо застосувати це до графа понять, то можна сказати, що в графі повинні бути відсутні контури. Для знаходження контурів у графі понять використовується метод пошуку в глибину зі зберіганням шляху [4–6].

Суть алгоритму пошуку в глибину є такою (рис. 5–6). Нехай у нас є граф G = <V,E>, де V – скінченна непорожня множина вершин, E – скінченний набір ребер графа.

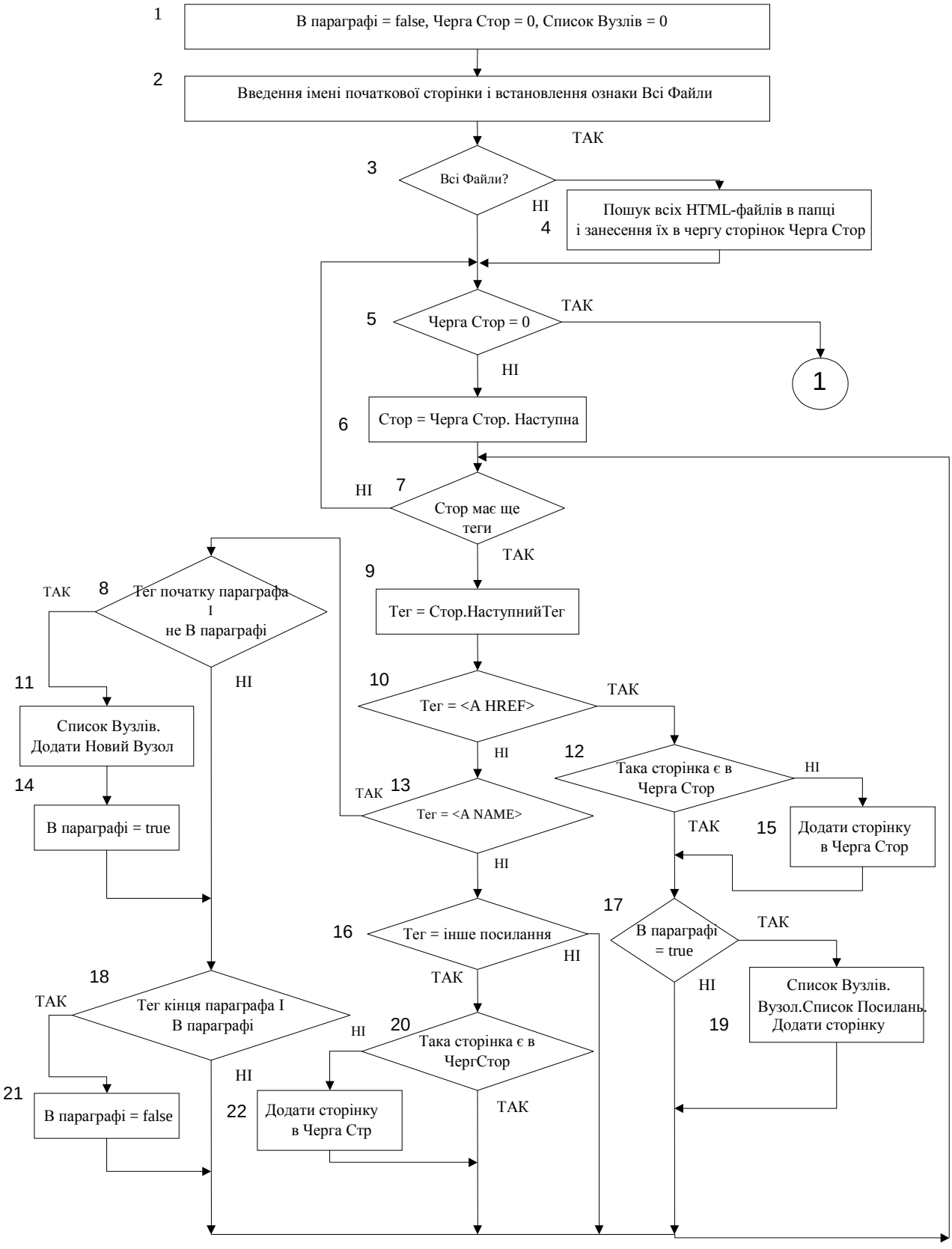

Рис. 5. Блок-схема першого етапу алгоритму формування структури

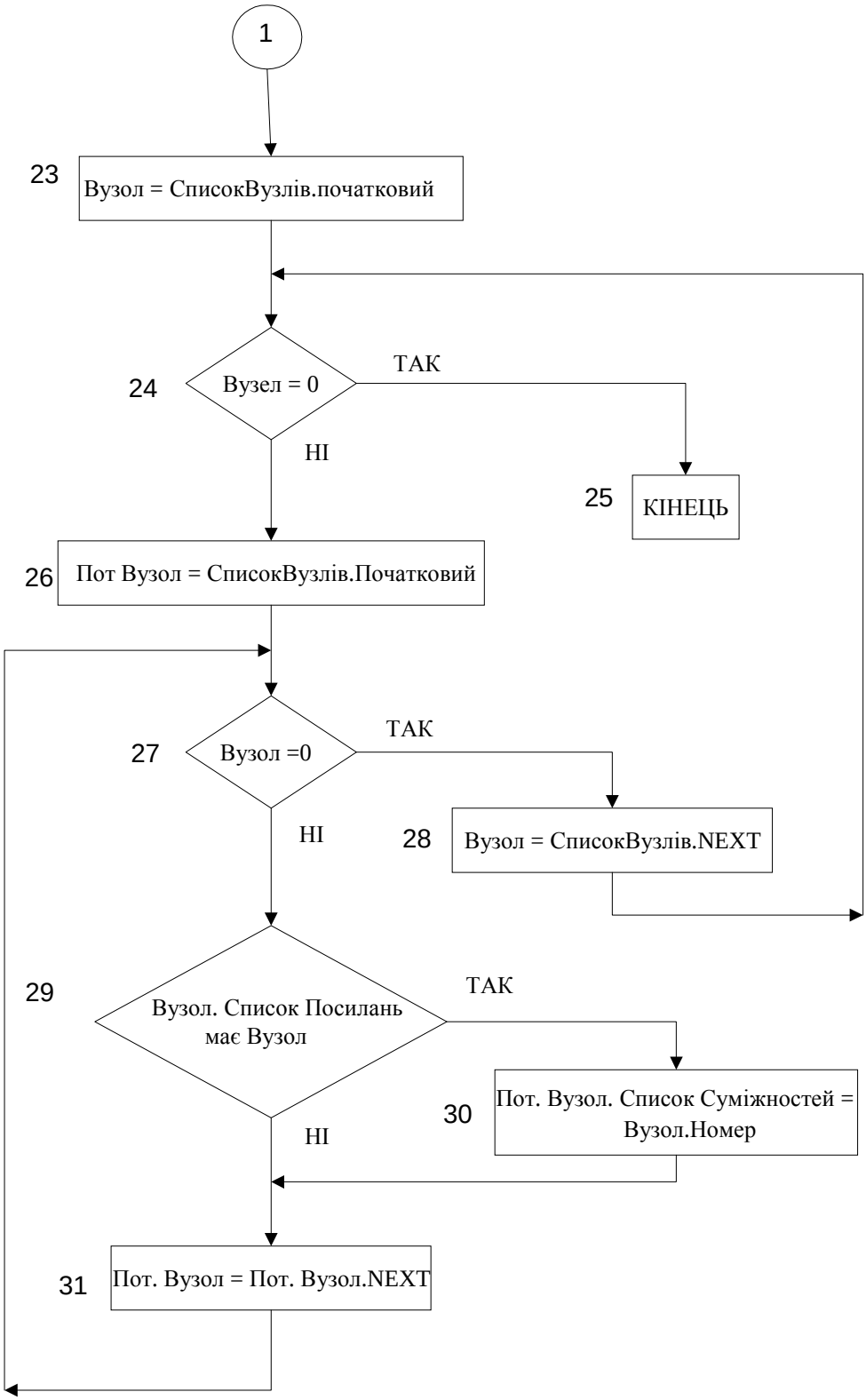

Рис. 6. Блок-схема другого етапу алгоритму формування структури

Починаємо пошук із довільної вершини $v_{0.}$

Вибираємо довільну вершину *u*, суміжну з v_0 і повторюємо пошук від вершини *u*.

У загальному випадку припустимо, що ми знаходимося в деякій вершині *v*. Якщо існує ще не переглянута вершина *u*, суміжна для *v*, то повторюємо пошук, починаючи з вершини *u*. Якщо непереглянутих вершин, суміжних із *v*, немає, то вважаємо вершину *v* використаною і повертаємося у вершину, із якої потрапили в *v*. Якщо при поверненні одержуємо $v = v_0$, то вважаємо алгоритм завершеним, а усі вершини – використаними.

Ідея пошуку контурів за допомогою методу пошуку в глибину полягає в тому, що при пошуку зберігається шлях, який складається з переглянутих вершин. Якщо ми зустрічаємо вже переглянуту вершину – це означає, що виявлений контур. При цьому за збереженим у пам'яті шляхом ми можемо відновити вершини цього контуру.

Вихідними поняттями є ті поняття, при визначенні яких не використовуються інші поняття, що містяться у підручнику. У графі понять такими є початкові вершини графа, які не мають вхідних дуг.

Алгоритм виявлення таких понять тривіальний. Проглядаються усі вершини графа, і ті вершини, що не мають вхідних дуг, включаються у список.

Підструктури за заданою множиною понять можуть виділятися двома засобами. Перший засіб – виділення підграфа від вихідних понять, тобто від початкових вершин, до обраних вершин. Другий – виділення підграфа від обраних вершин до кінцевих вершин графа понять. Для обох варіантів використовується в принципі однаковий підхід.

У першому варіанті для кожної початкової вершини графа відшукується шлях до кожної з обраних вершин. Отримані в результаті цього шляхи об'єднуються, й утворюється шукана підструктура.

У другому варіанті відмінність полягає в тому, що для кожної обраної вершини графа шукається шлях до кожної кінцевої вершини. Потім також отримані шляхи об'єднуються.

Шлях між двома вершинами знаходять з використанням методу пошуку в глибину. Щоб знайти шлях між вершинами *v* і *u* методом пошуку в глибину, потрібно почати пошук у вершині *v* і продовжувати його до відвідування вершини *u*. У момент відвідування вершини *u* послідовність переглянутих (але невикористаних) вершин буде визначати шлях із *v* у *u*.

РЕАЛІЗАЦІЯ МОДУЛЯ ФОРМУВАННЯ СТРУКТУРИ

Сформулюємо основні критерії вибору середовища програмування для створення даного модуля.

1) максимально можлива зручність у роботі, тобто зручний і сучасний інтерфейс користувача;

2) модуль має працювати з максимальною швидкістю. Небажані ситуації, коли користувачу доведеться довго очікувати закінчення роботи модуля;

3) підтримка довгих імен файлів;

4) максимальна простота в установці і використанні модуля;

5) мінімальні витрати на розробку модуля;

Модуль забезпечує виконання таких функцій:

– вибір файла сторінки електронного підручника, із якого почнеться опрацювання і який є початковим файлом підручника або вибір папки, у якій знаходиться підручник;

– опрацювання підручника, починаючи з заданого файла або усіх файлів підручника в заданій папці з метою побудови структури підручника;
– запис отриманої структури у вихідний файл у визначеному форматі, необхідний для роботи модуля опрацювання і відображення структури.

Реалізація модуля опрацювання і відображення структури

Сформулюємо основні критерії вибору середовища програмування для створення даного модуля.

- Створення максимально можливої зручності в роботі. Для цього модуль повинний мати зручний і сучасний інтерфейс користувача.
- Модуль має інтегруватися з електронним підручником, щоби при роботі зі структурою можна було б відразу ж переглянути той елемент у підручнику, до якого виникло зацікавлення.

Електронний підручник має бути написаний на мові HTML і для його перегляду може використовуватися будь-який стандартний браузер Internet. Відповідно інтеграція модуля з підручником означає, що цей модуль також має працювати під керуванням браузера, щоб мати можливість доступу до підручника.

Існує декілька варіантів для реалізації подібного роду взаємодії між програмою і браузером.

Реалізація модуля як програми з інтерфейсом CGI. CGI – Common Gateway Interface є стандартом інтерфейсу (зв'язку) зовнішньої прикладної програми з інформаційним сервером типу HTTP, Web. Зазвичай гіпертекстові документи, що витягаються з WWW серверів, містять статичні дані. За допомогою CGI можна створювати CGI-програми, або шлюзи, що у взаємодії з такими прикладними системами, як система керування базою даних, електронна таблиця, ділова графіка та ін. зможуть видати на екран користувача динамічну інформацію. Програма-шлюз запускается WWW сервером у реальному масштабі часу. WWW сервер забезпечує передачу запиту користувача шлюзу, а вона, у свою чергу, використовуючи засоби прикладної системи, повертає результат опрацювання запиту на екран користувача. Програма-шлюз може бути написана практично на будь-якій мові програмування.

Недоліки цього варіанта очевидні. Цей варіант потребує обов'язкової установки WWW-сервера, дуже важко організувати нормальний інтерфейс із користувачем, все виведення у вікно браузера здійснюється тільки командами HTML, що ускладнює реалізацію й обмежує можливості програми. Перевагою даного варіанта є те, що CGI-програма за винятком інтерфейсу є повноцінною програмою, яка не має будь-яких обмежень з питань безпеки.

Використання мов сценарію JavaScript або VB Script. JavaScript – мова для упорядкування сценаріїв (скриптів), розроблена фірмою Netscape. Ця мова по суті є підмножиною мови Java, але має свою специфіку. Мова VB Script розроблена компанією Microsoft також для створення скриптів. Як випливає з назви, в основу цієї мови покладено Visual Basic. Ці мови застосовуються для створення інтерактивних Web-сторінок. Текст скрипта записується в HTML сторінку і виконується браузером. Мова JavaScript підтримується в браузерах Netscape Navigator, починаючи з версії 2.0 і Internet Explorer, починаючи з версії 3.0. VB Script підтримується тільки в Internet Explorer, починаючи з версії 3.0.

Дані мови функціонально дуже обмежені й орієнтовані в основному на роботу з документами HTML, створення користувацького інтерфейсу і виконання елементарних

функцій. Цілком відсутні можливості роботи з файлами, використовувані структури даних і можливі операції над ними занадто примітивні, щоб дозволити побудувати достатньо складну програму.

*Створення Java-апплета.*Мова Java була задумана як машинно-незалежна й об'єктно-орієнтована мова програмування для Internet. Оскільки мова HTML, використовувана як стандарт в Internet, не вирішувала багатьох проблем, пов'язаних із наданням користувачу якісно нових можливостей для роботи в глобальній мережі (перегляд сторінок, що містять відеозображення і звук, керування переглядом і зручні графічні засоби для роботи), то виникла необхідність у деякій уніфікованій мові, що інтерпретується на різноманітних апаратних платформах. Спочатку як таку мову хотіли використовувати мову C++ шляхом розширення її можливостей і адаптації до вимог роботи в мережі. Але в процесі роботи з'ясувалося, що деякі риси мови C++ не задовольняють вимоги мови для глобальної мережі. Основною хибою був явний розподіл пам'яті і відповідно робота з вказівниками, що ускладнювало б використання програм на різноманітних платформах. У результаті була створена об'єктно-орієнтована машинно-незалежна, синтаксично схожа на C++ мова програмування в глобальній мережі – мова Java.

Мову Java можна використовувати для розробки програм двох типів: самостійних додатків і аплетів. Самостійний додаток створюється як окрема програма, виконувана інтерпретатором мови Java. Аплет – це програмний код, виконуваний інтерпретатором мови Java, вмонтованим у Web-браузер.

Сьогодні інтерпретатори мови Java вмонтовані у найбільше популярні Web-браузери такі, як Netscape Navigator і Microsoft Internet Explorer.

Компілятор мови Java перекладає програми у так званий байт-код. Байт-код є машинно-незалежним кодом, сформованим відповідно до специфікацій JVM (Java Virtual Machine). Інтерпретатор мови Java виконує скомпільований байт-код, використовуючи необхідні класи тієї апаратної платформи, на якій він виконується.

Цей варіант найбільше підходить для реалізації поставленої задачі. Аплет є практично повнофункціональним додатком, за винятком деяких обмежень, що накладаються вимогами безпеки. Він забезпечує можливість створення графічного користувацького інтерфейсу, подібного до інтерфейсу системи MS Windows. Є можливість керувати переглядом документів у браузері, будувати на екрані складні графічні зображення.

Мова Java надає потужні засоби для створення й опрацювання складних структур даних. Нові браузери використовують зараз поряд з інтерпретаторами Java також динамічні компілятори, що значно збільшує швидкість виконання аплетів. Отже, останній варіант, тобто реалізація модуля у вигляді аплета, найбільше задовольняє поставлені у цій роботі вимоги.

Модуль забезпечує виконання таких функцій:

1. Зчитування файла зі структурою електронного підручника, сформованого модулем формування структури.
2. Перевірка даної структури на коректність і у випадку виявлення некоректності відображення некоректної ділянки структури.
3. Відображення ліченої структури на екрані у вигляді графа.
4. Відображення списку вихідних (невизначених) понять.
5. Пошук заданого елемента в графі за його назвою;

6. Можливість перегляду обраного елемента безпосередньо в електронному підручнику;
7. Виділення підструктури за заданою множиною вихідних понять.

Вимоги до апаратного і програмного забезпечення. Як уже зазначалося вище, цей модуль являє собою аплет, і, отже, призначений для перегляду Web-браузером. Сьогодні найбільше поширені два Web-браузери – це Microsoft Internet Explorer і Netscape Navigator. Проте, незважаючи на те, що обидва ці браузери підтримують аплети Java, існують деякі розходження в їхній інтепретації, пов'язані з відсутністю єдиного стандарту та конкуренцією між фірмами Microsoft і Netscape. Виходячи з того, що браузер Internet Explorer сьогодні найбільше поширений, цей модуль орієнтований саме на роботу з ним.

Модуль встановлюють копіюванням каталога STRUCT на жорсткий диск комп'ютера, який містить клас аплета та інші класи, необхідні для роботи модуля. Рекомендується розміщати даний каталог у каталозі, що містить електронний підручник, оскільки тоді файл структури, створений модулем формування структури в цьому ж каталозі, буде автоматично завантажений модулем опрацювання структури (у випадку використання параметрів за замовчуванням); упровадження модуля в одну зі сторінок підручника або створення окремої HTML сторінки, використовуваної для запуску модуля. У обох випадках необхідно вставити в документ такий код:

<applet code=struct. Struct. class width=200 height=100></applet>

Цей код може бути доповнений або можуть бути змінені такі параметри, як ширина і висота, але ім'я класу повинно залишитися незмінним, причому регістр букв також має значення.

Модуль запускають у браузері через сторінку, що містить посилання на аплет модуля. Його можна вбудувати як в окрему сторінку HTML, так і в одну зі сторінок електронного підручника.

Під час запуску головного вікна модуля автоматично перевіряється наявність у каталозі, з якого був відкритий аплет, що містить файл HTML, файла з ім'ям «STRUCT.DAT». Якщо цей файл виявлено у каталозі, то структура завантажується з нього. Якщо ж файл із таким ім'ям відсутній, то на екран видається повідомлення про помилку, і робота модуля завершується.

Відразу після завантаження структура аналізується на коректність. У випадку виявлення в ній некоректних зв'язків видається відповідне повідомлення, й у вікні перегляду відображається некоректний ланцюжок понять. У цьому випадку необхідно виправити у підручнику зазначені некоректності, сформувати наново його структуру і потім уже продовжити її аналізувати.

При успішному відкритті і читанні файла у вікні перегляду відобразиться шуканий граф понять, і модуль буде готовий для роботи з ним.

ВИСНОВОК ТА ПЕРСПЕКТИВИ

Нині розроблено широкий спектр педагогічних, програмних та технічних засобів для побудови систем дистанційного мережевоцентричного навчання (СДМН). Проте подальший розвиток дистанційних технологій навчання стає неможливим без побудови ґрунтовної формально-математичної основи, у першу чергу – без створення формальної моделі мережевоцентричного навчання. Без такої основи неможливо розробити ефективні методи проектування та вдосконалення СДМН. Спостерігається дещо однобічний

гуманітарно-педагогічний підхід до ДО, фахівці якого зосереджуються на дидактично-педагогічних особливостях ДН. Як наслідок, нині відсутні розширення для мережево-центричних технологій усталених методів і засобів аналізу та проектування інформаційних систем (зокрема структурних методологій). Відсутність математичної моделі системи мережевоцентричного навчання унеможливлює розроблення алгоритмів оптимізації СДМН та прогнозування поведінки таких систем у часі. Як наслідок, недостатньо розвинуті методи повноцінного розроблення СДМН та їх адміністрування.

Специфічні особливості дистанційної освіти (ДО) обумовлені використанням нових засобів у процесі навчання – телекомунікаційних і інформаційних технологій. Дистанційна освіта – це, насамперед, система, у якій взаємодіє низка необхідних елементів: людина, що навчається, з її освітніми запитами, змістовний компонент, який складається з електронного підручника, системи завдань, системи контролю знань як з боку учня (самоперевірка), так і з боку викладачів, системи моніторингу і керування навчальним процесом і ін., організаційний компонент, під яким можна розуміти кілька колективів людей, що забезпечують реалізацію процесу навчання – це автори, методисти, координатори (викладачі ДО), психологи, а також програмно-телекомунікаційна група.

У ході роботи над проектом було проведено аналіз сьогоднішнього стану автоматизованих навчальних систем і засобів їхньої розробки, виявлено їхні переваги та недоліки. На основі цього були сформульовані вимоги до навчальної системи, частина з яких лягла в основу даної роботи.

Стаття має на меті підвищення ефективності роботи викладачів при створенні електронних навчальних систем, дозволяючи поліпшити якість та зменшити терміни розробки подібних систем.

1. Берко А.Ю., Висоцька В.А. Intranet архітектура інтелектуальних систем електронного навчання // Вісн. Нац. ун-ту "Львівська політехніка". – 2001. – № 438. 2. Андриенко Г.Л., Андриенко Н.В. Интеллектуальная гипертекстовая система для исследования проблем и обучения // Сб. трудов: Конф. по искусственному интеллекту КИИ-94. – Тверь, 1994. – С. 58–62. 3. Норенков Ю.И., Михайловский О.В. Адаптивная автоматизированная обучающая система // Сб. трудов: Конф. по искусственному интеллекту КИИ. – Тверь, 1994. 4. Нечепуренко М.И., Попков В.К., Майнагашев С.М. и др. // Алгоритмы и программы решения задач на графах и сетях. – Новосибирск: Наука. Сиб. Отд-ние, 1990. – 515 с. 5. Кристофидес Н. Теория графов. – М.: Мир, 1978. 6. Емеличев В.А., Мельников О.И., Сарванов В.И., Тышкевич Р.И. Лекции по теории графов. – М.: Наука, 1990. 7. Баженова И.Ю. Язык программирования Java. – М.: Диалог-МИФИ, 1997 – 288 с. 8. Гелеверя Т.Е. ДОСТУП-2 – Инструментарий разработки курсов дистанционного обучения // Телематика 2001: Тр. Междунар. конф. ГОСНИИ ИНФОРМАТИКА науч.-метод. – Москва - С.-Петербурн, 2001. 9. Грицай В.П. Информационная технология "Динамический гипертекст ТЕТ-А-ТЕТ" // Вестн. Междунар. Соломонова ун-та. – 2000. – Вып.4. – С. 124–133. 10. Брукс Ф. Как создаются программные системы. – СПб.: Символ, 2001. 11. Информационные технологии в учебном процессе – Львов: Малти М, 2001. 12. Пасека Г., Стецюк А. Инструментальные средства для создания систем обучения с испоьзование Web // Раб. сем. Национального центра инновационных технологый в обучении. – К., 2000. 13. Новиков Ф.А. Дискретная математика для программистов. – К., 2003. 14. Капітонова Ю.В., Кривий С.Л., Летичкевський О.А., Луцький Г.М., Печурін М.К. Основи дискретної математики. – К, 2002.