Міністерство освіти і науки, молоді та спорту України
Чернігівський національний педагогічний університет
імені Т.Г.Шевченка

Горошко Юрій Васильович

Інформаційне моделювання у підготовці учителів математики та інформатики

Монографія

Чернігів
2012

ББК
Г
УДК 37.091.12:51:004
Рекомендовано до друку
Рецензенти

ЗМІСТ

ВСТУП

Глобалізація, зміна технологій, перехід до постіндустріального, інформаційного суспільства, утвердження пріоритетів сталого розвитку, інші властиві сучасній цивілізації риси зумовлюють розвиток людини як головну мету, ключовий показник і основний важіль сучасного прогресу, потребу в радикальній модернізації галузі освіти, ставлять перед державою, суспільством завдання забезпечити пріоритетність розвитку освіти і науки, першочерговість розв'язання їх нагальних проблем [205]. В цьому ж документі відмічено, що “Пріоритетом розвитку освіти є впровадження сучасних інформаційно-комунікаційних технологій, що забезпечує дальше удосконалення навчально-виховного процесу, доступність та ефективність освіти, підготовку молодого покоління до життєдіяльності в інформаційному суспільстві” [205].

Оскільки вміння моделювати надзвичайно важливе для дослідника і є невід’ємною рисою будь-якої творчої особистості, якою повинен бути і майбутній педагог, дуже важливо сформувати у нього компетентності щодо моделювання, а особливо інформаційного моделювання, у всіх сферах його майбутньої професійної діяльності. Оволодіння ефективними прийомами опрацювання даних на основі їх формалізації і структуризації за допомогою інформаційного моделювання полегшує сприйняття їх постійно зростаючих потоків.

Важливість проблем формування навичок інформаційного моделювання підтверджують вимоги до кваліфікаційних рівнів та систем компетентностей більшості фахівців з різних напрямів діяльності, в тому числі і до педагогів та дослідників. До цих вимог обов’язково включаються вміння формулювати і розв’язувати проблеми, застосовувати основоположні закономірності системного аналізу, абстрагування, формалізації, оскільки інформаційне моделювання є важливим компонентом пізнавальної діяльності. Тому навчання інформаційного моделювання є важливою складовою всього процесу навчання у

педагогічному університеті. Особливо актуальною проблема навчання інформаційного моделювання постає в зв'язку з широким розповсюдженням сучасних інформаційних технологій і комп'ютерного моделювання, яке буде успішним тільки у випадку, якщо студенти оволодіють вміннями інформаційного моделювання, такими як постановка цілі, системний аналіз, формалізація.

Як відомо, у змісті навчання інформатики як одна з основних виділяється лінія моделювання [172-175, 207]. У працях О.О. Ракітіної та інших [11, 12] зміст навчання інформатики поділено на такі змістові лінії: "Інформаційні процеси і інформаційні системи", "Моделювання і формалізація", "Управління і інформаційні технології".

За [13] лінію "Моделювання і формалізація" називають методологічною, оскільки при її вивченні формуються основні знання, вміння і навички, необхідні для ефективного використання сучасних інформаційних технологій в професійній діяльності. Поняття інформаційної моделі розкривається через поняття формалізації. В загальному випадку формалізація – це відкидання зайвого і виділення головного. Якщо в процесі формалізації не врахувати важливих моментів, а приділити увагу другорядним, то інформаційна модель може стати неадекватною початковому інформаційному процесу.

В процесі як навчання, так і практичної діяльності люди мають справу з різноманітними інформаційними моделями. Значна частина об'єктів, з якими людині доводиться мати справу, фактично є моделями. Опанування цього факту є важливим фактором соціалізації студентів у сучасному інформаційному суспільстві. Тому розвиток навичок формалізації є одним з важливих завдань навчання у вузі. Причому інформатиці тут відводиться особлива роль, оскільки формалізація є основою алгоритмізації і автоматизації, а отже і сучасних інформаційних технологій, оскільки автоматизація в підсумку призводить до поняття алгоритму, а алгоритм –

основа програмних засобів, які є головним інструментом сучасних інформаційних технологій.

Традиційно вважають, що інформаційна модель об’єкта – це його певним чином формалізований і структурований опис, поданий за допомогою тих чи інших засобів (знакових систем, різних мовних засобів і т.п.), зокрема це можуть бути тексти, графіка, формули, звукозаписи, мисленні образи і т.п.

Стійке оволодіння навичками інформаційного моделювання дозволяє більш глибоко засвоїти предмети, що вивчаються, а також сформувати риси активної особистості, задатки вченого-дослідника.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ ФОРМУВАННЯ СИСТЕМИ КОМПЕТЕНТНОСТЕЙ В ГАЛУЗІ ІНФОРМАЦІЙНОГО МОДЕЛЮВАННЯ МАЙБУТНІХ ВЧИТЕЛІВ МАТЕМАТИКИ І ІНФОРМАТИКИ

1.1. Поняття про модель і моделювання

1.1.1. Моделювання як метод пізнання. Щодо поняття моделі у сучасних вчених до цього часу немає єдності у поглядах, оскільки це поняття застосовується у найрізноманітніших сферах людської діяльності. Але при розкриванні і тлумаченні поняття моделі використовуються наступні ознаки [304, с.231]:

- відображення і (або) відтворення (імітація) в моделі досліджуваного об’єкту або процесу;
- придатність до заміщення об’єкту, процесу, що пізнається;
- придатність для отримання нового знання про об’єкт;
- наявність точних умов і правил побудови моделі і переходу від відомостей про модель до відомостей про об’єкт.

А.А. Леонова [153] вказує на те, що сучасні уявлення про моделювання невідривно пов’язані з конкретними теоретичними поняттями, моделювання при цьому розглядається як процес руху від абстрактного до конкретного, а сама модель інтерпретується як об’ємне синтетичне поняття, що відображає єдність теоретичного, емпіричного і системно-структурного підходів, або іншими словами модель – це конкретна структура об’єкта в єдності його функціонування і розвитку. Таким чином системно-структурний метод включає в себе моделювання не тільки як умову, але і як важливий елемент системного погляду на речі.

В.О. Штофф [313] стверджує, що модель є гносеологічним образом в силу того, що завжди при уявній, а також і матеріальній побудові моделі її слід розглядати, як систему, гносеологічно вторинну в порівнянні з

об’єктом пізнання. При цьому поняття гносеологічного образу охоплює не тільки індивідуальну свідомість, але і суспільну, і не тільки свідомість, але і предметну діяльність, тому передбачається також врахування засобів реалізації, втілення і зберігання відомостей, що накопичені людством.

Я.Г. Неуймін [183] робить висновок про універсальність поняття моделі. Він вважає, що незалежно від рівня, цілей, методів і засобів пізнання в основі гносеологічного процесу знаходиться модель, і модель же є його результатом, але вже така, що отримала розвиток, збагачена новими відомостями, подана певним ідеальним образом, знаковою системою або реалізована матеріально.

Е.П. Семенюк [228] розглядає процес відображення дійсності в уяві людини як її моделювання, а модель як універсальну форму гносеологічного відображення, що відображає певний загальний аспект пізнавального процесу, методологічний підхід в науковому дослідженні. На його думку моделлю є довільний гносеологічний образ, що виступає на довільному рівні наукового пізнання – евристичному, теоретичному, метатеоретичному.

Процес моделювання людиною включає такі мислительні операції, як абстрагування, ідеалізація та формалізація.

Абстрагування дозволяє позбавитися від несуттєвих властивостей об’єктів чи процесів, що моделюються, сконцентрувавшись на суттєвих для даної задачі.

Сутність ідеалізації полягає в тому, що опираючись на абстрагування відношень, ці відношення перетворюють в об’єкти, і за рахунок цього з’являється можливість їх вивчати.

Формалізація полягає у використанні строгої наукової мови для опису моделі (математичні позначення, мови програмування тощо).

І.Б.Новік дає наступне тлумачення моделі: “Модель – це штучний об’єкт (що представляє собою речовинний агрегат або знакову систему), що знаходиться в об’єктивній відповідності з об’єктом, який

досліджується, придатний для заміщування досліджуваного об’єкта на певних етапах пізнання, такий, за допомогою якого в процесі дослідження можна отримати певні відомості, які можна перевірити дослідом, і які за встановленими правилами можна перевести у відомості про об’єкт, що досліджується” [184].

Наведемо також тлумачення поняття моделі, подане в “Українській радянській енциклопедії” [293]:

“Модель (франц. modele від лат. modulus – міра, мірило, зразок) – у широкому розумінні – предмет, явище, система (опис, схема, знак, графік, план, макет, форма тощо), що за певних умов виступає у значенні замінника чи представника якогось іншого предмета, явища чи системи. Також це наукове поняття, пов’язане з методом моделювання. В цьому значенні модель – речовознакова або уявна система, за допомогою якої відтворюються, імітуються чи відображуються принципи внутрішньої організації або функціонування, певні властивості, ознаки чи характеристики об’єкта дослідження (оригіналу), безпосереднє вивчення якого неможливе, ускладнене або недоцільне, і якою можна замінити цей об’єкт у пізнавальному процесі з метою одержання нових знань про нього”. Таким чином відношення “модель – оригінал” не природне, а зумовлене процесом пізнання, і питання про їх співвідношення, ступінь подібності, адекватності є одним з найважливіших і найскладніших у процесі застосування моделі у науковому пізнанні. Спрощене, однобічне розуміння моделі як тотожної оригіналу або як такої, що має мало спільного з ним, не тільки спричинює хибні наукові результати, а й є джерелом серйозних помилок методологічного характеру, призводить до філософських заблуджень, особливо при моделюванні процесів мислення, інтелектуальної діяльності людини. Лише у значенні певної аналогії з оригіналом при чіткому визначенні їх подібності і відмінності модель виступає ефективним засобом наукового дослідження [293; С.66].

У Великому тлумачному словникові сучасної української мови наводиться наступне тлумачення терміну модель [23]:

1) Зразок якого-небудь нового виробу, взірцевий примірник чогось.
2) Тип, марка конструкції. Нова модель машини.
3) Зразок, що відтворює, імітує будову і дію якого-небудь об'єкта, використовується для одержання нових знань про об'єкт. Модель простору. || Предмет, відтворений у зменшеному, іноді у збільшеному або натуральному вигляді.
4) Те, що є матеріалом, натурою для художнього зображення, відтворення. || Особа, яка позує перед живописцем або скульптором; натурщик, натурщиця.
5) Зразок, з якого знімається форма для відливання або відтворення в іншому матеріалі.
6) Уявний чи умовний (зображення, опис, схема і т. ін.) образ якого-небудь об'єкта, процесу або явища, що використовується як його "представник".
7) Конструкція, структура, зразок, за яким побудована певна одиниця мови з одиниць нижчого рівня.
8) Система математичних залежностей або програма, що відображає суттєві властивості об'єкта, процесу чи явища, які вивчаються.

Таким чином різні тлумачення моделі досить тісно корелюють між собою.

Розглянемо тепер означення поняття “моделювання” за [293; с.64]:

“Моделювання – непрямий, опосередкований метод наукового дослідження об’єктів пізнання (безпосереднє вивчення яких з певних причин неможливе, ускладнене чи недоцільне) шляхом дослідження їх моделей. Як специфічний пізнавальний прийом, своєрідна форма відображення об’єктивної дійсності моделювання виникло в античному світі одночасно з появою наукового пізнання. Наукові основи

моделювання почали закладатися з виникненням точного природознавства – в епоху Відродження. Перший етап його наукового розвитку пов'язаний з генезисом і встановленням теорії подібності (Г.Галілей, І.Ньютон та інші). В епоху науково-технічного прогресу моделювання фактично перетворилося на загальнонауковий метод пізнання, стало ефективним теоретичним і експериментальним засобом дослідження складних процесів і явищ дійсності. Розвитку моделювання особливо сприяла поява кібернетики і комп'ютерів. Моделюванню, модельному, зокрема модельно-кібернетичному експериментові, належить важлива роль у перевірці наукових гіпотез, побудові і розвитку різних теорій. Моделювання на основі широкого застосування абстрагування та ідеалізації дає змогу виділяти (а потім відтворювати у моделях і досліджувати) саме ті параметри, характеристики чи властивості модельованих об'єктів, які безпосередньо підлягають пізнанню. Моделювання надзвичайно розширює можливості наукового пізнання, оскільки дозволяє наочніше уявляти досліджувані явища, "наближувати" їх, змінювати реальний режим їх протікання, усувати шкідливий вплив супровідних сторонніх факторів тощо. Процес моделювання поділяється на такі основні етапи: постановка проблеми, побудова (вибір) моделі, її дослідження, екстраполяція одержаних результатів на оригінал. Теоретичною основою моделювання є теорія фізичної подібності, що служить основою фізичного моделювання, при якому модель і оригінал мають однакову фізичну природу; теорія фізичної аналогії, що становить основу предметно-математичного моделювання, коли модель і оригінал відрізняються матеріально, але мають еквівалентний математичний опис; теорія ізоморфізму та гомоморфізму систем, що лежить в основі знакового моделювання, яке зводиться до оперування знаковими моделями (формулами, знаками тощо) за певними фіксованими системами правил. Особливо важливу роль у науковому пізнанні відіграє такий вид знакового моделювання, як логіко-математичне, зокрема інформаційне моделювання, що здійснюється

засобами математики, математичної логіки і інформатики. В практиці наукового пізнання зустрічається також мисленне моделювання, яке полягає в мисленному оперуванні чуттєво-наочними образами, умоглядними конструкціями, схемами, системами суджень тощо".

Об'єкт, для якого створюється модель, називають оригіналом або прототипом. Будь-яка модель не є абсолютною копією свого оригіналу, в ній лише відображаються деякі якості й властивості оригіналу, найбільш істотні для обраної мети дослідження. При створенні моделі завжди робляться певні допущення й гіпотези [118].

На основі системного підходу можна створювати повноцінні моделі. Особливості системного підходу полягають у наступному. Об'єкт, що досліджується, розглядається як система, опис і дослідження елементів якої не виступає як сама мета, а виконується з урахуванням їх місця (наявність підзадач). У цілому об'єкт не відокремлюється від умов його існування й функціонування. Об'єкт розглядається як складова частина чогось цілого (сам є підзадачею). При системному підході на перше місце виступають не тільки причинні пояснення функціонування об'єкта, але й доцільність включення його до складу інших об'єктів. Допускається можливість наявності у об'єкта множини індивідуальних характеристик і ступенів свободи. Створення універсальних моделей – це наслідок використання системного підходу [118].

Моделювання в багатьох випадках є незамінним способом пізнання. Наприклад, дослідження міцності греблі на річці неможливо провести шляхом прямого експерименту – як з економічної, так і з екологічної точок зору, тому параметри такої греблі розраховують на основі її математичної моделі. В інформатиці даний спосіб називається обчислювальним експериментом і ґрунтується на трьох основних поняттях: модель – алгоритм – програма.

1.1.2. Види моделювання. Існує кілька ознак, за якими класифікуються моделі [118]. Наприклад стосовно фактору часу моделі

можуть бути статичними та динамічними. За допомогою статичної моделі описується об’єкт на певний момент часу, а в динамічній – відображається процес змінювання об’єкта з часом.

За цілями використання можна виділити навчальні моделі, дослідні моделі, науково-технічні моделі, імітаційні моделі, ігрові моделі.

Деякі важливі положення щодо ігрового моделювання розглянемо згідно [194], де відмічено, що ігри тісно пов’язані з моделями, оскільки будь-яка гра – це модель життя. В зв’язку з цим в грі моделюються реальні життєві події, ситуації. За допомогою ігрових моделей реалізують діяльність для імітації реальної практики. Ігрові моделі лежать в основі військових маневрів, ділових, дитячих та спортивних ігор.

Однак не всі моделі відносяться до класу ігрових. В ігрових моделях люди грають для себе, і якщо в процесі такої гри виникає нове знання, воно буде сприйняте учасниками ігрового процесу. Практично в будь-якій діловій грі моделюється реальна професійна ситуація, разом з тим ділові ігри містять такі складові, які відрізняють їх принципово від всіх інших технологій перш за все тим, що вони проводяться (здійснюються) за операціональним сценарієм або блок-структурою, в які закладено більш-менш жорсткий алгоритм “правильності” і “неправильності” рішення, що приймається, тобто учасник бачить той вплив, що здійснили його рішення на майбутні події [194].

Пізнавальний ефект ігрових технологій обумовлено комбінованим використання трьох методів: аналітичного, експертного і експериментального [194].

За допомогою аналітичного методу конструюється гра. В процесі такого конструювання аналізується великий фактичний матеріал, виявляються важливі елементи і зв’язки, формулюються гіпотези і теоретичні положення щодо явищ, які вивчаються. Участь професіоналів у грі підвищує їх експертний потенціал [194].

Експертний метод виявляється в тому, що спостерігаючи систему, яка вивчається, зсередини, гравці і експерти аналізують і переоцінюють свій попередній досвід і знання [194].

Експериментальний метод дозволяє кожну гру розглядати як лабораторний експеримент над системою, що вивчається.

Імітаційні і ділові ігри можуть застосовуватися для розв'язування різноманітних задач: навчання, прийняття господарських рішень, організаційного проектування, досліджень. Використання ігор як засобу навчання надає можливість підвищення мотивації навчально-пізнавальної діяльності учнів, застосовувати отримані знання для розв'язування практичних задач [194].

Застосування ігор для господарських рішень припускає включення їх у вигляді блоку у використовувану систему управління. Регулярне проведення таких ігор дозволяє робити прогнози, проводити заходи стосовно розвитку виробництва, відпрацьовувати організаційні структури або механізми господарювання [194].

Ще одним базовим поняттям ігрового моделювання є дослідницькі цілі. Вони служать для отримання нових даних, для перевірки гіпотез і теоретичних положень.

Для кожної гри складається програма, в якій визначаються задачі, які потрібно розв'язати. В ході гри отримують дані з варіантами розв'язання цих задач (джерелами даних є спостереження за поведінкою гравців, результати анкетного опитування, пропозиції, що висловлюються гравцями у підсумковій дискусії). Ці дані опрацьовуються і аналізуються організаторами гри. Результати аналізу фіксуються у звіті [194].

За галузями знань виділяються моделі біологічні, економічні, історичні, соціологічні і т.д.

Наприклад, *економічне моделювання*, або моделювання економічних процесів – метод дослідження економічних об'єктів за відповідними моделями. Здійснюється за допомогою знакових та матеріальних систем,

що відображають властивості, зв’язки, відношення об’єктів та економічних явищ. Економічне моделювання дозволяє уникнути небажаного експериментування з реальними економічними об’єктами. В економіці широко використовується математичне моделювання, коли економічні процеси описують математичними залежностями. Для аналізу таких залежностей широко використовують комп’ютери. Результати аналізу таких моделей дозволяють будувати прогнози, давати реальні оцінки економічних процесів [293].

Моделі в економіці почали використовувати у XVIII-му столітті. Досить широко в економічному моделюванні використовують оптимізаційні методи, наприклад група методів, що об’єднуються терміном “лінійне програмування”, зокрема симплекс-метод. Поряд з лінійними моделями економічних процесів широко відомі нелінійні та стохастичні моделі.

Процес моделювання в економіці має ряд послідовних стадій. На першій стадії в загальному вигляді ставиться мета моделювання. На другій стадії виокремлюють об’єкт (процес). Для третьої стадії характерне виявлення внутрішніх властивостей та внутрішніх і зовнішніх зв’язків характеристик процесу, що моделюється. Це дає можливість сформулювати найважливіші обмеження для математичної моделі і конкретизувати мету її розробки, а також визначити в разі потреби критерій оптимальності розв’язку. На четвертому етапі проводиться математична постановка економічної задачі, для розв’язування якої застосовують комп’ютер [293].

При економічному моделюванні, як і при багатьох інших видах моделювання, слід пам’ятати, що термін “оптимальність” відноситься до моделі, а не до реального життя. Те, що оптимально в моделі, не завжди оптимально в реальному житті. Справа в тому що в моделі в тій чи іншій мірі спрощується реальне життя. Тому при побудові моделі слід враховувати, що модель повинна бути настільки детальною, щоб результат

задовольняв потреби дослідника, ступінь детальності відповідала доступним даним, а побудову і аналіз моделі можна було провести за прийнятний час.

Моделі можуть бути *матеріальні* та *абстрактні*. Матеріальна модель завжди має реальне втілення і використовується для проведення експерименту, в якому вивчається певна її властивість (властивості), наприклад модель автомобіля для дослідження аеродинамічних характеристик у аеродинамічній трубі. Абстрактні ж моделі орієнтовані на теоретичні дослідження, наприклад чисельні експерименти. До абстрактних моделей зокрема відносять інформаційні.

Інформаційна модель – це сукупність даних про об'єкт, за допомогою яких описують властивості і стан об'єкта, процесу або явища, а також зв'язки його з навколишнім світом.

В інформаційних моделях об'єкти подаються у вигляді словесних описів, текстів, малюнків, таблиць, схем, креслень, формул і т.д.

Форма подання інформаційної моделі залежить від способу кодування (алфавіту) і матеріального носія.

Уявлюване моделювання – відображення об’єкта в уяві. Такі моделі формуються в уяві людини, що сприяє її свідомій діяльності. Їх створення завжди передує створенню матеріального об'єкта, матеріальної і інформаційної моделі, будучи одним з етапів творчого процесу. Наприклад, програма в уяві програміста – уявлювана модель програмного продукту.

Вербальне (словесне) моделювання – це подання інформаційної моделі засобами природної розмовної мови (фонемами). Уявлювана модель, виражена в розмовній формі, називається вербальною (від латинського слова verbalize – усний). Форма подання такої моделі – усне або письмове повідомлення. Прикладами є літературні твори, дані в навчальних посібниках і словниках, інструкції користування пристроєм, правила дорожнього руху.

Наочне моделювання – це вираження властивостей оригіналу за допомогою образів. Наприклад, малюнки, художні полотна, фотографії, кінофільми. При наочному моделюванні поняття часто кодуються рисунками – іконічне моделювання. Сюди ж відносяться геометричні моделі – інформаційні моделі, подані засобами графіки.

Геометрична модель – певний об’єкт, геометрично подібний своєму оригіналу. Призначені геометричні моделі для навчальних або демонстраційних цілей. В якості прикладів можна навести моделі геометричних фігур (куба, конуса тощо), моделі автомобілів, літаків і т.п. Найважливішими характеристиками геометричних моделей є розміри та пропорції.

В *образно-знаковому моделюванні* використовуються знакові образи якого-небудь виду: схеми, графи, креслення, графіки, плани, карти. Наприклад, географічна карта, план квартири, родовідне дерево, графічна схема алгоритму. До цієї групи відносяться *структурні інформаційні моделі*, створювані для наочного зображення складових частин і зв'язків об'єктів. Найбільш прості й розповсюджені інформаційні структури – це таблиці, схеми, графи, блок-схеми, дерева.

В *знаковому моделюванні* використовуються алфавіти формальних мов: умовні знаки, спеціальні символи, букви, цифри й передбачається сукупність правил оперування цими знаками. Приклади: спеціальні мовні системи, фізичні або хімічні формули, математичні вирази і формули, нотний запис тощо. Програма, записана за правилами мови програмування, є знаковою моделлю процесу розв’язування задачі [118].

Математичне моделювання – це метод дослідження процесів чи явищ шляхом побудови системи математичних співвідношень (математичних моделей), що є описами досліджуваних об’єктів, та їх подальшого вивчення (аналізу). При математичному моделюванні використовують загальні основи природознавства, спеціальні закони конкретних наук, результати спостережень та експериментів, імітаційне моделювання за

допомогою комп’ютерів тощо. Математичне моделювання здійснюється засобами математики та логіки з використанням, як правило, комп’ютерів. Дослідження математичних моделей дає змогу передбачити розвиток процесу, розрахувати його характеристики, здійснювати управління цим процесом, проектувати системи з бажаними характеристиками тощо.

Математична модель – спосіб подання інформаційної моделі, що відображає зв'язок різних параметрів об'єкта через математичні формули й поняття. В [77] наведено класифікації математичних моделей.

За принципами побудови *математичні моделі* поділяють на:

1. Аналітичні.

2. Імітаційні.

В аналітичних моделях процеси функціонування реальних об'єктів, процесів або систем описуються у вигляді явних функціональних залежностей.

Аналітичні моделі поділяються на типи в залежності від математичної проблеми:

1) рівняння (алгебраїчні, трансцендентні, диференціальні, інтегральні);
2) апроксимаційне задання (інтерполяція, екстраполяція, чисельне інтегрування та диференціювання);
3) задачі оптимізації;
4) стохастичні проблеми.

Однак в міру ускладнення об'єкта моделювання побудова аналітичної моделі перетворюється в проблему, що важко розв’язується. Тоді дослідник змушений використовувати *імітаційне моделювання*.

У імітаційному моделюванні функціонування об'єктів, процесів або систем описується набором алгоритмів. За допомогою алгоритмів імітують реальні явища, що пов’язані з процесом або системою із збереженням їх логічної структури і послідовності змін в часі (перебігу процесу). Імітаційне моделювання дозволяє за вихідними даними отримати відомості про стани процесу або системи в певні моменти часу. Можна сказати, що

імітаційні моделі – це проведені на комп'ютері обчислювальні експерименти з математичними моделями, за допомогою яких імітують поведінку реальних об'єктів, процесів або систем.

В залежності від характеру досліджуваних реальних процесів і систем математичні моделі можуть бути:

1. Детерміновані.

2. Стохастичні.

В детермінованих моделях передбачається відсутність будь-яких випадкових впливів, елементи моделі (змінні, математичні зв'язки) досить точно встановлені, поведінку системи можна точно визначити. При побудові детермінованих моделей найчастіше використовуються алгебраїчні рівняння, інтегральні рівняння, алгебра матриць.

В стохастичних моделях враховується випадковий характер процесів в досліджуваних об'єктах і системах, що описуються за методами теорії ймовірності та математичної статистики.

За типом вхідних даних моделі поділяються на:

1. Неперервні.

2. Дискретні.

Якщо дані і параметри є неперервними, а математичні зв'язки стійкими, то модель – неперервна. І навпаки, якщо дані та параметри – дискретні, а зв'язки нестійкі, то і математична модель дискретна.

У відповідності до описуваних явищ і процесів моделі поділяються на:

1. Статичні.

2. Динамічні.

В статичній моделі описується стан об'єкта, процесу або системи в певний фіксований момент часу. В динамічних моделях відображається змінювання об'єкта, процесу або системи в часі.

За ступенем відповідності між математичної моделлю і реальним об'єктом, процесом або системою математичні моделі поділяють на:

1. Ізоморфні (однакові за формою).

2. Гомоморфні (різні за формою).

Модель називається ізоморфною, якщо між нею і реальним об'єктом, процесом або системою існує повна поелементна відповідність. Модель називається гомоморфною – якщо існує відповідність лише між найбільш значними складовими частинами об'єкта та моделі.

Математичні моделі можна класифікувати також згідно з особливостями застосованого в них математичного апарату:

1. Математичні моделі з зосередженими параметрами.

2. Математичні моделі з розподіленими параметрами.

Як правило за допомогою математичних моделей з зосередженими параметрами описують динаміку систем, що складаються з дискретних елементів. В математичному тлумаченні такі моделі – це системи звичайних лінійних або нелінійних рівнянь. Такі моделі широко використовуються для опису систем, що складаються з дискретних об’єктів або сукупності однакових об’єктів. Як приклад можна навести динамічну модель напівпровідникового лазера. В цій моделі фігурують дві динамічних змінних: концентрації неосновних носіїв заряду і фотонів в активній зоні лазера.

У випадку складних систем кількість динамічних змінних, і відповідно, диференціальних рівнянь може бути великою (сотні і тисячі). В цих випадках корисними є різні методи редукції системи, що базуються на ієрархії системи в часі, оцінці впливу різних факторів і відкиданні несуттєвих серед них тощо.

Математичними моделями з розподіленими параметрами описуються процеси дифузії, теплопровідності, розповсюдження хвиль різної природи тощо. Ці процеси можуть бути не тільки фізичної природи. Такі моделі широко розповсюджені в біології, фізіології і інших науках. Частіше за все в якості основи такої математичної моделі застосовують рівняння математичної фізики, в тому числі і нелінійні [143].

У тих випадках, коли моделювання орієнтоване на дослідження моделей за допомогою комп'ютера, одним з його етапів є розробка *комп'ютерної моделі*.

Комп'ютерна модель – це створений за рахунок використання ресурсів комп'ютера віртуальний образ, за допомогою якого якісно й кількісно відображаються внутрішні властивості й зв'язки об'єкта, що моделюється. За допомогою такої моделі можна передавати і зовнішні характеристики об’єкта.

Статистичне моделювання – це дослідження об’єктів пізнання на їх статистичних моделях. Воно зводиться до чисельних методів розв’язування математичних задач, при яких шукані величини описують за допомогою ймовірнісних характеристик певного випадкового явища і визначають їх шляхом статистичного аналізу результатів спостережень за функціонуванням моделі. Як приклад використання статистичного моделювання можна навести метод Монте-Карло. Метод Монте-Карло – це загальна назва групи чисельних методів, що базуються на отриманні великої кількості реалізацій стохастичного процесу, який формується так, щоб його ймовірнісні характеристики були досить близькими до з аналогічних величин, якими характеризується об'єкт, що досліджується. Використовується статистичне моделювання для розв’язування задач в галузях фізики, математики, економіки, оптимізації, теорії управління і інших.

1.2. Поняття компетентності. Компетентності в освіті

За [299] компетентність у перекладі з латинської competentia означає коло питань, у яких людина добре обізнана, має знання та досвід. Компетентна в певній сфері людина має відповідні знання та практичний досвід, що дозволяють їй обґрунтовано висувати судження щодо цієї сфери й ефективно діяти в ній.

Там же відмічено, що у методиках навчання окремих предметів поняття компетентності використовується давно, наприклад, поняття лінгводидактичних компетентностей застосовується при вивченні мов, комунікативних – при вивченні інформатики. В останні роки поняття «компетентність» вийшло на загальнодидактичний і методологічний рівень. Це пов'язано з його системно-практичними функціями й інтеграційною метапредметною роллю в загальній освіті. Посилення уваги до цього поняття обумовлене також рекомендаціями Ради Європи, що стосуються відновлення освіти, її наближення до замовлення соціуму. Необхідність формування в школі ключових компетентностей учнів відзначена в концепції модернізації вітчизняної освіти.

В [299] вказано, що не існує єдиного узгодженого визначення та переліку ключових компетентностей. Оскільки система компетентностей – це насамперед замовлення суспільства на підготовку його громадян, такий перелік багато в чому визначається узгодженою позицією соціуму в певній країні або регіоні. Досягти такого узгодження вдається не завжди. Наприклад, у ході реалізації міжнародного проекту «Визначення та добір ключових компетентностей», який здійснюється Організацією економічного співробітництва й розвитку та національних інститутів освітньої статистики Швейцарії та США, строгого визначення ключових компетентностей вироблено не було.

Під час симпозіуму Ради Європи на тему «Ключові компетентності для Європи» було визначено такий орієнтовний перелік ключових компетентностей.

Вивчати:

- уміти здобувати користь із досвіду;
- з'ясовувати взаємозв'язки своїх знань і впорядковувати їх;
- організовувати свої власні прийоми вивчення;
- уміти вирішувати проблеми;
- самостійно займатися своїм навчанням.

Шукати:

- запитувати різні бази даних;
- опитувати оточуючих;
- консультуватись в експертів;
- отримувати дані;
- уміти працювати з документами та класифікувати їх.

Думати:

- з'ясовувати взаємозв'язки минулих і поточних подій;
- критично ставитись до того чи іншого аспекту розвитку суспільства;
- уміти протистояти непевності та труднощам;
- займати позицію в дискусіях і обгрунтовувати свої власні думки;
- аналізувати політичне й економічне оточення, в якому проходять навчання та робота;
- оцінювати соціальні звички, пов'язані зі здоров'ям, споживанням, а також із навколишнім середовищем;
- уміти оцінювати твори мистецтва й літератури.

Співробітничати:

- уміти співробітничати та працювати у групі;
- приймати рішення - улагоджувати розбіжності та конфлікти;
- уміти домовлятись;
- уміти розробляти та виконувати контракти.

Прийматися за справу:

- включатись у проект;
- нести відповідальність;
- входити до групи або колективу та робити свій внесок;
- доводити солідарність;
- уміти організовувати свою роботу;
- уміти користуватись обчислювальними та моделюючими приладами.

Адаптуватись:

- уміти використовувати сучасні інформаційно-комунікацйні технології;
- доводити гнучкість перед викликами швидких змін;
- показувати стійкість перед труднощами;
- уміти знаходити нові рішення.

За словами С.А.Ракова [213, с.7], поняття компетентності в освіті виникло як розуміння того факту, що компетентність випускника загальноосвітнього навчального закладу слід поставити як завдання і як місію освіти, і ця компетентність має забезпечити йому, з одного боку, можливість самореалізації в суспільстві, а з другого — сприяти прогресу суспільства на позиціях науки, гуманізму, демократії, становленню та поглибленню громадянського суспільства. Мірою компетентності випускника загальноосвітнього закладу є її відповідність прийнятим у суспільстві еталонам компетентності члена суспільства.

Там же С.А.Раков вказує, що навчальні компетентності – це інтелектуальний розвиток особистості та здатність учитися протягом усього життя, і формулює напрями набуття навчальних компетентностей:

- застосовувати фундаментальні навички лічби, письма і читання;
- застосовувати навички використання інформаційно-комунікаційних технологій;
- організовувати і рефлексувати власний навчальний процес (аналізувати і оцінювати хід своїх думок і дій), обирати і застосовувати ефективні стратегії навчання та нові комп'ютерно-орієнтовані технології;
- застосовувати технології пошуку, аналізу та систематизації відомостей з різних джерел, навички критичного мислення, стратегії набуття нових знань та умінь, включаючи запам'ятовування, різні способи письмового фіксування, нагромадження відповідного асоціативного досвіду;

- аналізувати об'єкти, ситуації та взаємозв'язки, використовувати та оцінювати власні стратегії розв'язування пізнавальних проблем, висловлювати свою думку, застосовувати різноманітні прийоми аргументування в різних соціокультурних контекстах;
- досліджувати у системі різні технічні, наукові, соціальні та інші питання.

За словами О.М. Спіріна [254] під інформаційно-комунікаційними компетентностями розуміють підтверджену здатність особистості застосовувати на практиці інформаційно-комунікаційні технології для задоволення власних потреб і розв'язування суспільно-значущих, зокрема, професійних, задач у певній предметній галузі або виді діяльності.

С.Г.Литвинова виділяє 6 рівнів ІКТ-компетентностей вчителів загальноосвітніх навчальних закладів [158]:

1) початковий – це рівень розуміння необхідності ІКТ для розвитку освіти;
2) мінімальний базовий – це рівень знань і вмінь стосовно використання готових програмних продуктів;
3) базовий – рівень володіння знаннями і вміннями стосовно використання основних понять ІКТ;
4) поглиблений – це оперування знаннями з ІКТ у професійній діяльності;
5) дослідницький – рівень вчителя, який вільно оперує знаннями з ІКТ, інтернет-ресурсами і використання їх у дослідницькій, проектній діяльності;
6) рівень експерта – рівень знань, вмінь, досвіду, практичної діяльності що дозволяють вільно оперувати знаннями з ІКТ, інтернет-ресурсами, оцінювати інноваційний розвиток ІКТ і виступати в якості експерта з питань впровадження ІКТ у навчально-виховний процес.

За рекомендаціями ЮНЕСКО 2011 р. [258] сформульовано перелік компетентностей, що необхідні вчителям у всіх аспектах їх роботи:

1. Розуміння ролі ІКТ в освіті.
2. Навчальна програма і її оцінювання.
3. Педагогічні практики.
4. Технічні і програмні засоби ІКТ.
5. Організація і управління освітнім процесом.
6. Професійний розвиток.

Поєднання переліку цих компетентностей з трьома підходами до навчання, що базуються на розвитку потенціалу людини ("Застосування ІКТ", "Оволодіння знаннями", "Здобування знань") задає структуру системи ІКТ-компетентностей вчителів, яка таким чином включає 18 модулів (Табл. 1.1).

Таблиця 1.1.

Структура ІКТ-компетентностей вчителів (ЮНЕСКО)

	Застосування ІКТ	Оволодіння знаннями	Здобування знань
Розуміння ролі ІКТ в освіті	TL.1	KD.1	KC.1
Навчальна програма і її оцінювання	TL.2	KD.2	KC.2
Педагогічні практики	TL.3	KD.3	KC.3
Технічні і програмні засоби ІКТ	TL.4	KD.4	KC.4
Організація і управління освітнім процесом	TL.5	KD.5	KC.5
Професійний розвиток	TL.6	KD.6	KC.6

За цим документом ЮНЕСКО, стратегічна задача, що розв'язується в рамках підходу "Застосування ІКТ" – підготувати учнів, громадян та

працівників, які можуть використовувати ІКТ для соціального розвитку і економічного зростання своєї країни. Вчитель повинен знати, де і як використовувати (або не використовувати) ІКТ при роботі в класі, при поданні матеріалу, при розв'язуванні задач управління навчальним процесом, а також в ході професійного розвитку – поглиблення своїх знань в предметній галузі і методиці. На ранніх етапах реалізації даного підходу вчителі повинні вміти добирати і використовувати в своїй роботі існуючі навчальні і ігрові програми, різноманітні веб-ресурси, а також тренажери для формування навичок. Вчителі повинні вміти застосовувати ІКТ для досягнення освітніх результатів, які передбачені освітніми стандартами, для оцінювання знань. Крім цього, вчителі повинні використовувати ІКТ для ведення поточної звітності і свого професійного розвитку.

У рамках підходу "Засвоєння знань" вчителі використовують інструментальні програмні засоби, які відносяться до відповідної предметної галузі: візуалізацію при вивченні природничих наук, інструменти для аналізу даних при вивченні математики, моделювання і рольові ігри при вивченні суспільних наук. Компетентності, пов'язані з цим підходом, включають в себе вміння маніпулювати даними, структурувати проблеми і ставити задачі, поєднувати застосування інструментальних програмних засобів (в межах свого предмету) з методами особистісно-орієнтованої навчальної діяльності, з виконанням учнями спільних навчальних проектів. Для виконання групових навчальних проектів педагоги повинні використовувати мережеві ресурси, що дозволяює учням працювати сумісно, отримувати доступ до даних і спілкуватись із зовнішніми експертами в ході аналізу і розв'язування обраних ними проблем. Вчителі також повинні вміти використовувати ІКТ для розробки планів і оцінки їх виконання при проведенні індивідуальних і групових навчальних проектів, контактувати з експертами і співробітничати з іншими педагогами, а також використовувати мережу

для отримання відомостей, зв'язку з колегами і іншими експертами з метою підвищення свого професійного рівня.

Разом з тим, як зазначає М.І. Жалдак, "Використання комп'ютера в навчальному процесі має бути педагогічно виваженим".

Задача, що розв'язується в межах підходу "Здобування знань" – виховання учнів, громадян і працівників, які здатні здобувати нові практично важливі знання, брати участь в інноваційному процесі і навчатися протягом всього життя. Педагоги, що використовують цей підхід, повинні вміти готувати і проводити навчальні заняття, направлені на досягнення цих стратегічних цілей, а також брати активну участь у розробці відповідних програм розвитку своїх шкіл. Педагоги повинні формувати в учнів навички громадянина суспільства знань, які необхідні для здобування нових знань, а саме розв'язувати проблеми, налагоджувати спілкування, співробітничати, експериментувати, критично мислити, займатись творчістю. Завдання вчителя – в явній формі моделювати таку поведінку учнів. Учителі повинні виступати в якості зразкових учнів, майстрів навчання і здобувачів знань, бути постійно учасниками освітніх експериментів і інновацій. Для роботи в рамках підходу "Здобування знань" педагоги повинні вміти:

- розробляти цифрові освітні ресурси і будувати навчальне середовище;
- використовувати ІКТ в якості інструмента для формування у учнів здатності здобувати знання і розвивати своє критичне мислення;
- підтримувати рефлексію як необхідну складову частину навчальної роботи;
- створювати в середовищі учнів і своїх колег навчальні спільноти і "спільноти знань".

Педагоги повинні володіти достатніми компетентностями, щоб бути провідниками в процесі формування і впровадження в життя бачення

(стратегії розвитку) своєї школи, як суспільства, що базується на інноваціях і постійному засвоєнні нового в умовах інформатизації освіти.

У дослідженні [316] С.М.Яшанов визначає загальну структуру системи інформатичних компетентностей вчителя.

За словами С.М.Яшанова володіння інформатичними компетентностями надає можливість учителю звільнити час від виконання рутинних операцій для пошуку творчих підходів у навчанні учнів з використанням засобів інформаційних технологій, що забезпечує вдосконалення та підвищення ефективності процесу навчання і готує учнів до життєдіяльності в інформаційному суспільстві.

Таким чином, інформатичні компетентності можуть бути охарактеризовані через ефективність діяльності з використанням комп'ютера, що означає ефективне застосування знань, умінь і навичок для розв'язування існуючих або поставлених перед людиною завдань. Грамотна людина знає про щось абстрактно, а компетентна – може на основі знання конкретно й ефективно вирішувати яке-небудь інформатичне завдання чи проблему. У той же час, компетентність означає відмову від прямого копіювання чужого досвіду, норм, традицій, зразків, звільнення від стереотипів, чиїхось вказівок, розпоряджень, установок [316, с.65].

На основі загальних інформатичних компетентностей та системи компетентностей у галузі інформаційного, математичного та комп'ютерного моделювання економічних процесів, сформульованої Н.В.Кузьміною та О.В.Струтинською в [147], пропонується наступна система компетентностей майбутніх вчителів математики і інформатики у галузі інформаційного моделювання:

- володіння методами аналізу об'єктів, побудови та дослідження інформаційних моделей задачі чи проблеми з предметної галузі;
- уміння добирати інформатичний та математичний апарат для створення інформаційних моделей задачі чи проблеми з предметної галузі;

- уміння досліджувати інформаційні моделі за допомогою сучасних прикладних програм, що використовуються в предметній галузі, до якої відносяться розглядувані задачі чи проблеми;
- уміння добирати сучасні програмні засоби для виконання комп'ютерного моделювання об'єктів, процесів, явищ даної предметної галузі;
- уміння з'ясовувати адекватність інформаційних моделей досліджуваним об'єктам, процесам, явищам;
- уміння визначати та оцінювати похибки комп'ютерного аналізу інформаційних та математичних моделей;
- уміння добирати мову програмування для реалізації сформованої інформаційної моделі задачі чи проблеми з даної предметної галузі;
- уміння реалізовувати інформаційні моделі у вигляді програми відповідною мовою програмування.

1.3. Аналіз сучасного стану психолого-педагогічних досліджень проблем формування у студентів системи компетентностей в галузі інформаційного моделювання

Питанням навчання учнів та студентів моделювання взагалі і інформаційного моделювання зокрема приділяли достатню увагу багато дослідників.

Моделювання і алгоритмізація відіграє важливу роль у формуванні як загальної, так і професійної культури студента. В доповіді на Міжнародному конгресі стосовно математичної освіти ще у 1989 р. академік Єршов А.П. сказав, що через програмування і побудову інформаційних моделей в змістову частину математики входять абстракції людської діяльності, властивості штучних і живих систем [88; с.28]. У статті «Про предмет інформатики» А.П. Єршов пише: «Як самостійна наука інформатика вступає в права тоді, коли в рамках відповідної конкретної науки будується інформаційна модель того або іншого

фрагмента дійсності – в інформатиці розглядаються методологічні принципи побудови таких моделей і маніпулювання ними» [89].

Рада Міністрів Республіки Бєларусь назвала в якості одного з пріоритетних напрямів фундаментальних наукових досліджень “математичні моделі і їх застосування до аналізу систем і процесів у природі та суспільстві” [187]. Це обумовлено тим, що математичне моделювання є потужним засобом наукового дослідження в природничо-математичних галузях знань, а в останній час інтенсивно проникає і в гуманітарні науки, наприклад, в соціологію та лінгвістику. Крім того математичні моделі і методи широко використовуються при розв’язуванні виробничих задач [140].

В моделюванні поєднуються переваги як теорії, так і експерименту. Академік А.А. Самарський відмічає, що робота не з самим об’єктом (явищем, процесом), а з його моделлю дає можливість безболісно, відносно швидко і без суттєвих затрат дослідити його поводження в довільних уявлюваних ситуаціях (переваги теорії). В той же час обчислювальні експерименти з моделями дозволяють, спираючись на міць сучасних обчислювальних методів і технічних інструментів інформатики, вивчати об’єкти достатньо повно, що недоступно при чисто теоретичних підходах (переваги експерименту) [221; с.4].

Спеціаліст повинен сприймати систему як комплекс взаємопов’язаних елементів, тобто модель, він знає, як влаштовані зв’язки між її компонентами і правила їх функціонування. Це дозволяє йому прогнозувати поведінку системи, знаючи, як функціонують її частини і як вони пов’язані між собою, і навпаки, за спостереженими ефектами поведінки системи визначити особливості функціонування її складових. Це суттєво відрізняє спеціаліста від псевдоспеціаліста, що тільки завчив послідовності дій для різних ситуацій і користується ними, не розуміючи того, як влаштована система. Таким чином спеціаліст оперує моделлю, а псевдоспеціаліст – набором шаблонів. Відповідно спеціаліст може

самостійно і досить легко проаналізувати систему, що споріднена з даною, а псевдоспеціаліст потребує, щоб йому надали нові шаблони для його діяльності, щоб він "зазубрив" їх і зміг користуватися ними в своїй діяльності. Тому суттєво важлива задача при підготовці спеціаліста навчити його будувати моделі процесів і явищ із предметної галузі його майбутньої діяльності і відповідно навчити оперувати такими моделями.

Ефективність моделювання залежить від теорій і гіпотез, які покладено в основу моделі і від яких залежать межі спрощень моделі відносно оригінала.

Ю.А.Шрейдер відмічає, що інформаційне моделювання є компонентом пізнавальної діяльності, тому треба навчати молоде покоління використовувати інформаційне моделювання в ході навчання і подальшої діяльності.

Розповсюджене сьогодні комп’ютерне моделювання буде більш успішним, якщо учні оволодіють пізнавальними (стрижневими) вміннями інформаційного моделювання: орієнтувальною основою діяльності, цілепокладанням, системним аналізом, основами формалізації, тобто способами побудови теоретичної інформаційної моделі (В.Б.Гісін, С.О.Бєшенков).

Інформаційне моделювання дозволяє встановити на більш високому рівні цілісне знання про оточуючий світ, перейти до фундаменталізації змісту освіти.

Мозолін В.В. вбачає 2 основних напрямки фундаменталізації курсів інформатики:

1) *математизація змісту* навчання й розвиток формального компонента діяльності (центральними поняттями інформатики стають комп’ютер і алгоритм);
2) побудова курсів інформатики від феномена інформації та інформаційних процесів до методів їх вивчення за допомогою

інформаційних моделей шляхом використання комп'ютера як засобу управління інформаційними процесами [169].

С.О.Семеріков вважає, що найбільш перспективним є курс, в якому об'єднуються ці підходи на основі широкого застосування комп'ютерного моделювання [200]. Він зауважує, що саме моделювання є тим самим "клеєм", який пов'язує всі інформатичні дисципліни. З метою їх інтеграції ним розроблена та впроваджена система наскрізного навчання моделювання на всіх спеціальностях педагогічних ВНЗ, що мають спеціалізацію "Інформатика". Так початкове ознайомлення з моделюванням відбувається при вивченні електронних таблиць на першому курсі. При вивченні теми "бази даних та інформаційні системи" відбувається ознайомлення з інформаційним моделюванням, яке є основою наступного курсу – "Інформаційні системи та технології". Процес алгоритмізації розглядається через побудову *алгоритмічних моделей*, природним продовженням яких є *програмні моделі.* Він пропонує почати вивчення основ об'єктно-орієнтованого програмування з огляду *об'єктно-орієнтованого моделювання*. Для студентів, які навчаються за спеціальністю "Математика та основи інформатики" ним запропоновано побудувати моделі числових об'єктів різної природи як інтерпретацію відповідних алгебраїчних структур – класів натуральних чисел, цілих, раціональних та комплексних чисел, реалізуючи при цьому класичні алгоритми "довгої арифметики". Вивчення класів поповнюваних одно та двовимірних масивів пропонується конкретизувати задачами на побудову векторних, поліноміальних та матричних об'єктів [200]. Далі в цьому курсі розглядаються візуальні моделі. Інший спосіб введення такого типу моделей пропонується при розгляді систем комп'ютерної математики, в яких можлива повна реалізація процесу моделювання (від постановки задачі до аналізу результатів обчислювального експерименту). *Геометричне моделювання* розглядається на основі вивчення бібліотеки

тривимірної графіки OpenGL в об’єктній реалізації на прикладі розв’язування задач кінематики, молекулярної динаміки та інших.

Імітаційне моделювання вивчається на основі алгебраїчних методів операційного числення. Такий кібернетичний підхід пропонується далі розвивати в курсі “Системи управління базами даних”, де основну увагу приділено реалізації розглянутих в шкільному курсі інформатики інформаційних моделей, зокрема моделей реляційної алгебри.

Також пропонується розглядати моделі інтелекту у відповідному курсі.

Проте в дисертаційній роботі С.О.Семерікова [239] все це наведено досить схематично. Не розглянуто такі важливі застосування моделювання, як технологія створення педагогічних програмних засобів, дискретні та неперервні моделі даних у стохастиці, що є важливими складовими частинами даного дослідження.

За С.О.Бєшенковим і його колегами [9] в інформатичних дисциплінах повинна наслідуватись логіка розвитку природничих наук, а саме реалізовуватись наступна послідовність: феномен (інформаційні процеси) – метод пізнання (інформаційне моделювання) – галузь застосування (інформаційні основи управління, інформаційні технології, інформаційні системи в соціальній сфері і т.п.). В інформаційному суспільстві починають спрацьовувати інформаційні принципи, тобто принципи, які так чи інакше пов’язані з фундаментальними поняттями “інформація”, “інформаційний процес”, “інформаційна система”.

До загальних інформаційних принципів можна, наприклад, віднести:

- основний тезис формалізації;
- принцип інформаційного моделювання;
- принцип інформаційного управління;
- принцип нелокальності інформаційних взаємодій;
- принцип універсальності цифрового кодування.

Суть основного тезису формалізації полягає в принциповій можливості розділення об’єкта і його позначення. Наслідками основного тезису формалізації є:

- факт автономності знаків і знакових систем (можливість оперування знаками без звертання до об’єкту);
- можливість різних інтерпретацій знаків і знакових систем [10].

Принцип інформаційного моделювання формулюється С.О.Бєшенковим та його колегами так: «наукове пізнання здійснюється шляхом моделювання, основні моделі – це описи об’єктів або процесів, тобто інформаційні моделі. Крім того інформаційні процеси і моделі відіграють важливу роль в спілкуванні і практичній діяльності» [10].

Причини, за якими поняття, наприклад математичної моделі, отримує характер інформаційного принципу, полягає в універсальному характері інформаційної моделі – описі об’єкта або процесу певною мовою. Наслідками цього принципу є:

- визнання наявності трьох класів моделей – матеріальних моделей, інформаційних моделей, уявних моделей;
- надання інформаційним моделям статусу самостійних об’єктів, за допомогою яких можна впливати на світосприймання та вчинки людей [9].

Принцип інформаційного управління пропонується розглядати як розширення і поглиблення кібернетичного аспекту інформатики, який в інформаційному суспільстві отримує нові риси, а саме коли вплив на об’єкт, яким управляють, здійснюються тільки через відповідні сигнали (повідомлення) [9]. Як приклад автори наводять “управління через нестабільність”, суть якого полягає в застосуванні малих дій в точках нестабільності (точках біфуркацій). Дослідження цих питань здійснюється в рамках так званої теорії катастроф (Р. Том, В.І. Арнольд, Ю. Мозер і ін.), що має застосування в різноманітних галузях людської діяльності. Зокрема, наявність точок біфуркації в даній системі говорить про те, що в

ній принципово неможливо передбачити розвиток процесу, а отже, ефективно реалізувати технологічний ланцюжок в традиційному розумінні – при формальному здійсненні кожного кроку.

Принцип нелокальності інформаційної взаємодії полягає в тому, що для такої взаємодії не діють закони збереження (обмін матеріальними предметами відрізняється від обміну даними). Загальноосвітнє значення принципу нелокальності полягає в глибшому осмисленні умов і наслідків людської діяльності. Зокрема, при розв'язуванні якої-небудь задачі або ухваленні якого-небудь рішення необхідно враховувати не лише безпосередні умови або результати, але і стан середовища, яке «оточує» цю проблему, її контекст [9].

І.О. Теплицький наголошує на наявності у процесі моделювання серйозного протиріччя: з одного боку моделювання неможливе без спрощення, без нехтування другорядними чинниками, а з іншого – завжди існує ризик "переспростити" модель, відкинувши якісь важливі риси об'єкту разом з другорядними [274]. На його думку, для подолання цього протиріччя не існує якихось конкретних рекомендацій. Це сфера особистого досвіду дослідника, заснованої на ньому професійної інтуїції, рівня та якості освіти, інтелекту та творчості дослідника. Тому важливим стає аналіз чужих і своїх вдалих і невдалих дій, постійні вправи, розв'язування дослідницьких задач. У [274] І.О. Теплицький пропонує набір задач дослідницького характеру із різних предметних галузей, демонструє побудову їх математичних моделей та пропонує табличний процесор для комп'ютерної реалізації цих моделей та їх дослідження. В [274] розглянуто моделі процесу поширення чуток і епідемій, моделювання в галузі екології (різні моделі популяції), моделювання у фізиці (механічні коливання, рух тіла в полі сили тяжіння, задача про політ паперового літачка, задача про м'яку посадку на місяць), стохастичне моделювання за допомогою випадкових чисел (метод Монте-Карло, моделювання броунівського руху).

Т.В. Мінькович пропонує скористатися засобами комп'ютерного моделювання для інтеграції теоретичної інформатики та інформаційних технологій, поєднуючи традиційну модель комп'ютерної системи як сукупність апаратного і системного програмного забезпечення з моделлю інформаційних процесів через розв'язування таких задач інформатики, як подання повідомлень та описів інформаційних процесів, вивчення та організації кібернетичних систем, інформаційне моделювання реального світу [165].

У процесі моделювання досить широко використовується аналогія. В навчанні аналогія – це педагогічний прийом, який полягає у встановленні подібності в певному відношенні між об'єктами, що вивчаються. Цінність аналогії у процесі навчання полягає в тому, на її основі полегшується засвоєння навчального матеріалу, активізуються думки студентів, їхні пізнавальні пошуки, виникають здогадки, правильність яких потребує подальших досліджень.

За допомогою аналогії відбувається перенесення відомостей з оригіналу на модель. Посилання відносяться до оригіналу, а висновки – до моделі.

Висновки за аналогією можна поділити на 3 види щодо їх вірогідності:

- строга аналогія (висновки вірогідні).
- нестрога аналогія (істинність висновків ймовірнісна).
- хибна аналогія (висновки хибні).

В роботі Н.В.Іванової та Лю Минь [117] строгу аналогію пропонують використовувати при вивченні нової мови програмування на основі знань про вже вивчену мову програмування. Так мову програмування С вивчають, використовуючи аналогії з вже вивченою мовою Паскаль. Нестрогу аналогію там же пропонують використовувати при розв'язуванні задач, однотипних з вже розв'язаними. Також аналогію використовують тоді, коли, намагаючись розв'язати більш складну задачу, починають з іншої, більш простої.

Введення моделювання у навчання математики дозволяє студентам опанувати той факт, що математичні конструкції – це моделі реальних відношень, що в свою чергу дозволяє суттєво змінити відношення студентів до математики, зробити їх навчання більш обдуманим і продуктивним, а саме підсилити:

1. “Синтаксичну” сторону алгоритмів, що вивчаються з метою координації і зближення підходів до вивчення алгоритмів в курсах математики і інформатики.
2. Модельний аспект з метою створення у студентів уявлення про весь технологічний ланцюг розв’язування задачі за допомогою комп’ютера [146].

В традиційному навчанні математики спостерігається в основному розв’язування задач, вже сформульованих в математичних термінах. Внаслідок цього студенти отримують навички в розв’язуванні достатньо складних математичних задач, проте виявляються зовсім безсилими перед простою практичною задачею, оскільки не вміють перевести її в математичну. Такі труднощі можна пояснити недостатньою сформованістю вміння здійснювати математичну формалізацію, яка в свою чергу є одним з етапів моделювання. Під формалізацією треба розуміти приведення суттєвих властивостей і ознак об’єкта моделювання до певної форми. У відповідності до сказаного в якості підсилення модельного аспекту в курсі математики Кузьменко М.В. визначено введення в процес навчання математики задач, в яких передбачається зведення суттєвих властивостей і ознак об’єкта моделювання до математичної форми, або іншими словами, математична формалізація [146].

Інформаційне моделювання широко використовується в психолого-педагогічних дослідженнях.

А.П.Панфілова [194] вбачає необхідність внесення в зміст навчання інформатики понять “модель” і “моделювання”. За її словами ця необхідність обумовлена задачею формування в учня науково-

теоретичного типу мислення, що означає мислення про дійсність через особливі специфічні об’єкти, сконструйовані в історичному процесі розвитку науки, – моделі реальних процесів і явищ. Вона вважає, що моделювання є вищою формою наочності і позиціонується для з’ясування і фіксації у зручному вигляді суттєвих особливостей і взаємозв’язків явищ, які вивчаються, що дозволяє використовувати моделювання для побудови і фіксації загальних схем дій і операцій, які учні повинні проробити в процесі вивчення складних абстрактних понять.

В [194] А.П.Панфіловою відмічено, що розробка інноваційних методів навчання, як правило, пов’язується з кількома видами діяльності:

1) пошуки на лінії репродуктивного навчання (“індивідуально призначене навчання”, “персоналізована система навчання”, “бригадно-індивідуальне навчання”), конкретно-дидактична основа якого пов’язана з програмованим навчанням;
2) пошуки на лінії дослідницького навчання, в рамках якого навчальний процес будується як пошук пізнавально-прикладних, практичних відомостей (нових інструментальних знань про способи професійної діяльності);
3) модель навчальної дискусії, характерними рисами якої є насамперед організація діяльності з ознайомлення кожного учасника з тими відомостями, які є в інших, сприяння різним підходам до одного і того самого предмета обговорення; рівноправне співіснування різних неспівпадаючих точок зору стосовно питань, що обговорюються; можливість критикувати і відхиляти будь-яку з точок зору, залучення учасників до пошуку групової, як правило, компромісної згоди у вигляді загального рішення;
4) організація навчальної діяльності на основі ігрової моделі, що передбачає включення в навчальний процес імітаційного і рольового моделювання, ігор, тренінгів і вправ.

На думку А.П.Панфілової в навчальному процесі сучасних вузів найбільш актуальними інноваціями є технології третього і четвертого видів. Обидві ці моделі навчання тісно пов'язані, оскільки будуються на організації діяльності, спрямованої на пошук і прийняття рішень і потім для внутрігрупової і міжгрупової дискусій стосовно проблем, що стосуються майбутньої професії і змодельовані в рамках ігрових технологій навчання: імітаційних і ділових ігор, сценаріїв інсценувань, мозкових атак і т.п.

Н.В.Сичкова [261] відмічає, що формування у майбутніх вчителів вмінь дослідницької діяльності забезпечується шляхом реалізації функціонально-змістової моделі, розробка якої здійснена на основі ідей:

а) функціонально-змістова модель процесу, що розглядається, будується у відповідності з принципами: єдності навчально-виховної і науково-дослідної діяльності; системності; моделювання наукових досліджень в дидактичній підготовці студентів; рефлексії;

б) змістове поле професійних знань визначається в логіці характеру і направленості дослідницької діяльності педагога.

Однією з характерних особливостей сучасної педагогічної науки є розповсюдження модельного підходу, що проявляється в активному використанні моделей на всіх етапах педагогічних досліджень.

При визначенні моделі фіксуються наступні ознаки [304; с.231]:

- відображення і (або) відтворення (імітація) об'єкту чи процесу, що вивчається, в моделі;
- придатність до заміщення досліджуваного об'єкта, процесу;
- придатність для отримання нових відомостей (нове знання про об'єкт);
- наявність точних умов і правил побудови моделі і переходу від відомостей про модель до відомостей про об'єкт.

Процес відображення дійсності в свідомості людини Е.П. Семенюк також розглядає як її моделювання, а модель – як універсальну форму

гносеологічного відображення, що виражає деякий загальний аспект пізнавального процесу, методологічний підхід в науковому дослідженні. В широкому сенсі, за Е.П.Семенюк, «моделлю є будь-який гносеологічний образ, що виступає на будь-якому рівні наукового пізнання – евристичному, теоретичному, метатеоретичному» [228, с.178].

На думку Н.В.Сичкової моделювання не є тотожно рівним відображенню, яке взагалі є однією з гранично загальних категорій теорії пізнання, але моделювання є активною реалізацією суб'єктом, який пізнає, пізнавального потенціалу, закладеного в принципі відображення. Модель використовується як засіб апріорного передбачення результатів, що формально-логічно виводяться, а з іншого боку – в ній закріплюється у вигляді ідеально організованого і впорядкованого об'єкта образ, який у своєму найбільш розвиненому вигляді набуває форми теорії, утворюючи тим самим опору для нового етапу мислення [261].

Будь-яка процедура моделювання неминуче ґрунтується на тому чи іншому виді абстракції і поза нею не може бути здійснена [313, с.7]. Також з моделюванням глибоко пов'язані такі мислительні процедури, як ідеалізація та формалізація.

Важливою задачею моделювання і найбільш високим його пізнавальним рівнем є побудова таких моделей, використання яких дозволяє виявити суть явищ і процесів, що досліджуються, в цілому, тобто розглянути їх як певні системи.

Фундаменталізація навчання будь-якої навчальної дисципліни розглядається як перетворення навчально-пізнавальної діяльності у таку впорядковану діяльність, яка може бути універсально застосовною і має творчий характер. В результаті утворюється єдина система понять, термінів, означень, позначень. Це дозволяє проводити колективні дослідження, є їх необхідною умовою. Індивідуалізовані дослідження синтезуються в колективне дослідження дисципліни, що вивчається, за програмою, базисом якої є стандартна програма. Фундаменталізація змісту

навчання дозволяє реалізувати прагнення студентів і викладачів до встановлення, розвитку і розширення міжпредметних зв’язків, що збагачує знання студентів [261].

Дидактичне моделювання як частковий випадок загальнонаукового моделювання має місце в педагогічній науці і практиці. Дидактичне моделювання здійснюється в три етапи [157]:

- етап формування моделі;
- етап перетворення або випробування моделі;
- етап перенесення результатів випробування на об’єкт моделювання.

Формування вмінь інформаційного моделювання відбувається поетапно: розвиток рефлексії, структурно-модельного мислення, здібностей працювати з даними у відповідності з розв’язуванням навчальних задач різного типу (репродуктивних, творчих, евристичних, проблемних; соціологічних, психологічних, педагогічних, математичних і ін.) [46].

Відповідний педагогічний метод для покращення розуміння наукових пояснень учням базується на *концептуальних моделях*, тобто наборах слів і/або діаграм, призначених для того, щоб допомогти учням створити розумові конструкції систем, що вивчаються [297]. В концептуальній моделі виділяються основні об’єкти і дії в системі, взаємозв'язки між ними. Використовуючи концептуальну модель, викладач може сприяти запам’ятовуванню вже існуючої, але неповної схеми, і не тільки проілюструвати, але і пояснити принципи, яких навчає.

За [297], використання моделей сприяє розпізнаванню і інтеграції нових відомостей з уже існуючими в пам'яті студента, зменшується на 14% в дослівному збереженні кількість тих слів, що використовувались при поясненні навчального матеріалу викладачем. Тобто студенти, які отримали модель, розуміють і можуть застосувати принципи, які їм пояснюють, і в меншій мірі схильні покладатися на механічне запам’ятовування слів викладача, що свідчить про осмислене навчання.

Дослідження показали, що концептуальні моделі найбільш ефективні, коли подаються студентам, які не мають достатніх попередніх знань або мають недостатні здібності, щоб засвоїти матеріал заняття. Більш ймовірно, що студенти з більш високими здібностями приходять на заняття з уже готовими моделями процесу або здатні їх швидко створити. З іншого боку, для тих студентів які мало знають або мало здібні, модель, надана викладачем, забезпечує контекст для засвоєння нових принципів.

1.4. Лінія моделювання в шкільній інформатиці

О.А.Кузнецов відмічає, що генеральний шлях розвитку інформатики в школі полягає в постійному підвищенні її загальноосвітнього потенціалу, який в значній мірі пов’язаний з формуванням умінь будувати за допомогою певних засобів інформаційні моделі, працювати з ними і аналізувати їх. Значно більш важливими, ніж частинні і дуже вузькі вміння працювати з тими чи іншими засобами інформаційних технологій, є вміння формалізувати, подавати інформаційні моделі за допомогою тих засобів, які зараз є в інформатиці [121].

І.В.Роберт зауважує, що використання методу інформаційного моделювання переводить процес навчання з рівня “повідомлення суми знань – засвоєння суми знань” на рівень “дослідницький підхід і прогнозування результатів експериментально-дослідницької діяльності”, дозволяє навчити учнів самостійно “відкривати” закономірності, що вивчаються, формувати узагальнене уявлення про оточуючий світ [220].

Інформаційне моделювання відображає тісні міжпредметні зв’язки інформатики з іншими навчальними предметами, а понятійний апарат і метод моделювання має широкі використання при вивченні практично всіх предметів [312].

В [144] відмічається, що навчальні задачі і ситуації в курсі інформатики будуються на базі змістових постановок задач і навчальних інформаційних моделей, які знайомі учням з інших курсів. При вивченні

інформатики вони розглядаються з “інформаційних” або “алгоритмічних” позицій, що досить часто приводить до поглиблення і систематизації знань учнів, появи нових асоціативних зв’язків.

Н.В.Морзе в [176] відмічає, що змістова лінія моделювання відноситься до теоретичних основ курсу інформатики, вважає, що програмні засоби інформаційних технологій – СУБД, табличні редактори та інші – слід розглядати як засоби для опрацювання інформаційних моделей. Разом з тим не слід вважати, що тема моделювання носить лише теоретичний характер і відокремлена від всіх інших тем. Алгоритмізація і програмування також мають пряме відношення до моделювання.

На думку Н.В.Морзе, інформаційне моделювання пов’язане з питаннями системології, системного аналізу. Ступінь глибини вивчення цих питань суттєво залежить від рівня підготовленості учнів. Учні, особливо середніх класів (базова школа), ще важко сприймають абстрактні, узагальнені поняття. Тому розкриття таких питань повинно спиратися на прості, доступні учням приклади.

За Н.В.Морзе вивчення моделювання у школі повинно починатися з розгляду такого важливого поняття, як об’єкт, а потім переходити до розгляду моделі, як замінника об’єкта, який має такі самі або близькі характеристики в плані предмету дослідження. Далі пропонується розділити моделі на фізичні (матеріальні) і інформаційні (знакові), виявити спільне і розбіжності в таких моделях, дається роз'яснення понять моделі та інформаційної моделі.

Автор виділяє три типи задач в галузі інформаційного моделювання, які за зростанням ступеня складності для сприймання учнями розташовуються у такому порядку:

1) задано інформаційну модель об’єкта; потрібно навчитися її аналізувати, робити висновки, використовувати для розв’язування задач;

2) дано набір несистематизованих даних про реальний об’єкт (процес, систему); потрібно систематизувати їх і таким чином створити інформаційну модель;
3) дано реальний об’єкт (процес, систему); потрібно розробити його інформаційну модель.

У підручнику з інформатики для 10-х класів [125] на вивчення об’єктів і моделей відведено 2 параграфи. Розглядається поняття об’єкта, класифікація об’єктів, поняття моделі, класифікація моделей:

- за способом подання;
- за формою подання;
- за галузями використання;
- з урахуванням фактору часу.

Далі пропонується розглянути побудову математичної моделі у наступному порядку:

1. Проаналізувати умову задачі та визначити мету побудови математичної моделі.
2. Визначити положення, на основі яких будуватиметься математична модель.
3. Визначити початкові дані й кінцеві результати.
4. Виокремити об’єкти, які розглядаються в умові задачі.
5. Виокремити властивості об’єктів, суттєві для розв’язування даної задачі.
6. Описати властивості об’єктів за допомогою математичних співвідношень, в яких пов’язуються кінцеві результати з початковими даними (ввести змінні та скласти рівняння, нерівності або системи рівнянь чи нерівностей, або задати функції, через які пов’язуються ці змінні відповідно до властивостей об’єктів).

Наведений впорядкований скінчений набір дій ілюструється нескладним прикладом.

Таким чином можна побачити, що на даний момент вивчення моделювання у шкільному курсі інформатики дуже фрагментарне, незначне за обсягом.

В російській шкільній інформатиці питанням моделювання, в тому числі і інформаційного, приділяється значно більше уваги, виділена окрема, значна за обсягом лінія “Формалізація і моделювання”. Зміст цієї лінії включає поняття:

- моделювання як метод пізнання;
- формалізація;
- матеріальні та інформаційні моделі;
- інформаційне моделювання;
- основні типи інформаційних моделей.

Лінія моделювання разом з лінією інформології та інформаційних процесів є теоретичною основою базового курсу інформатики.

Підручник Н.В. Макарової “Информатика 7-9” [120] базується на системно-інформаційній концепції шкільного курсу інформатики. Н.В.Макарова вважає, що інформатика, навчання якої дозволяє акумулювати знання з різних предметних галузей, є саме тією дисципліною, де реально можна втілити ідею розвитку системного мислення у кожного учня. Одним із сучасних інструментів системного аналізу та синтезу систем є інформаційне (абстрактне) моделювання, проведене за допомогою комп'ютерів. Інформаційні моделі повинні відображати суттєві з точки зору дослідника властивості об'єктів-оригіналів.

З огляду на сказане, в якості основних цілей навчання інформатики Н.В.Макарова виділяє наступні:

- формування інформаційної культури школяра, під якою розуміється вміння цілеспрямовано працювати з інформаційними ресурсами та використовувати для цього комп'ютер;

- навчання системного підходу до осмислення всього, що відбувається навколо, в процесі аналізу та дослідження структури інформаційних об'єктів та їх взаємозв'язків, які є моделями реальних об'єктів і процесів;
- розвиток логічного мислення, творчого та пізнавального потенціалу будь-якої дитини, її комунікативних здібностей, використовуючи для цього багатий комп'ютерний інструментарій.

Відповідно до цілей навчання у підручнику досить детально розглянуто поняття об’єкта і моделі, класифікацію моделей, поняття інформаційної моделі, подання інформаційної моделі у вигляді таблиці, поняття системи, інформаційних моделей елементів системи, основні етапи моделювання і відповідно реалізацію цих етапів у середовищі графічного редактора, текстового процесора, табличного процесора, в СУБД.

У темі “Моделювання в середовищі графічного редактора” розглянуто:

1. Поняття про моделювання в середовищі графічного редактора.
2. Моделювання геометричних об'єктів. Моделювання геометричних операцій. Моделювання об'єктів з заданими властивостями.
3. Конструювання – різновид моделювання. Моделювання паркету. Комп'ютерне конструювання з мозаїки. Створення меню музичних форм. Створення геометричних композицій з готових музичних форм. Створення набору цеглинок для конструювання. Конструювання із цеглинок за загальним виглядом. Створення розміщення меблів. Моделювання об'ємних конструкцій із цеглинок за трьома проекціями.
4. Різноманітність геометричних моделей. Моделювання різьблення на дереві. Моделювання віконних наличників. Моделювання топографічної карти або плану місцевості. Графічне подання опису процесу.

У темі “Моделювання в середовищі текстового процесора”:

Словесна модель. Моделювання складових документів. Структурні моделі: таблиця, схема, блок-схема, структура ділових документів. Алгоритмічні моделі.

У темі “Моделювання в електронних таблицях”:

Етапи моделювання в електронних таблицях. Розрахунок геометричних параметрів об'єкта. Моделювання ситуацій. Опрацювання масивів даних. Моделювання біологічних процесів. Моделювання руху тіла під дією сили тяжіння. Моделювання екологічних систем. Моделювання випадкових процесів.

У темі “Інформаційні моделі в базах даних”:

Етапи створення інформаційних моделей в базах даних. Стандартні та індивідуальні інформаційні моделі. Інформаційна модель "учні".

На питання, безпосередньо присвячені моделюванню, відводиться близько 50 годин.

Поряд з підручником з інформатики Н.В.Макаровою запропоновано практикум-задачник з моделювання [119]. Підручник складається з 4-х розділів.

В 1-му розділі містяться завдання щодо моделювання у графічному редакторі за допомогою його специфічних інструментів – лінійки, циркуля, транспортира. Також тут наведено завдання на такого виду моделювання, як конструювання, тобто розробку деталей та більш складних об’єктів з цих деталей, наприклад конструювання паркетів, мозаїк.

В 2-му розділі пропонуються завдання на моделювання в середовищі текстового процесора. Тут розглядаються вербальні (словесні) моделі, які в тестовому процесорі подаються у вигляді текстових документів. Таким чином текстовий документ з одного боку є поданням уявної моделі в знаковій формі, а з іншого – це об’єкт середовища текстового процесора. Наведено задачі на створення простих текстових документів (наприклад описів літературних героїв), складених документів (наприклад вітальних

листівок), наукових текстів (наприклад фрагментів підручників з математики, фізики і т.п.), структурних моделей – документів, що мають чітко визначену структуру (наприклад протоколів засідань, схем класифікацій), алгоритмічних моделей (наприклад графічних схем алгоритмів).

В 3-му розділі містяться задачі на моделювання в електронних таблицях, які розв'язуються відповідно до загальної схеми: постановка задачі, розробка моделі, комп'ютерний експеримент і аналіз результатів. Розглянуто задачі на розрахунок геометричних параметрів об'єкта (наприклад отримання коробки максимального об'єму з прямокутного листа картону), моделювання ситуацій (наприклад визначення кількості рулонів шпалер для обклеювання довільної кімнати, час обслуговування черги тощо), опрацювання масивів даних (наприклад статистичне опрацювання масиву денних та нічних температур), моделювання біологічних процесів (наприклад розрахунок і аналіз біоритмів), моделювання руху тіла під дією сили тяжіння (наприклад дослідження руху тіла, кинутого під кутом до горизонту, рух парашутиста), моделювання екологічних систем (розрахунок чисельності популяції), моделювання випадкових процесів (наприклад гра в кидання монети, гра в рулетку).

4-й розділ присвячений розгляду інформаційних моделей в базах даних. Розглянуто типові задачі на формування даних в таблиці, пошук серед таких даних потрібних, подання даних у зручній для користувача формі (наприклад інформаційна модель "Історичні події", інформаційна модель "Учні").

1.5. Вплив еволюції суспільства і освіти на розвиток методичної системи навчання інформатики

Інформаційно-комунікаційні технології відіграють все більшу роль у сучасному постіндустріальному суспільстві. Відповідно виникає проблема

вдосконалення системи освіти, приведення її у відповідність до потреб постіндустріального суспільства. Таку інноваційну систему освіти деякі дослідники почали називати “Освіта 2.0”.

Як відомо, останнім часом все більшої популярності набувають технології під загальною назвою Web 2.0. Термін було запроваджено у 2004 році видавництвом О'Рейлі (англ. O'Reilly Media) та комерційним організатором серії конференцій під назвою "Web 2.0", – МедіаЛайв (англ. MediaLive, сьогодні англ. CMP Technology). Web 2.0 – методика проектування систем, які стають тим кращими, чим більше людей ними користуються, тобто користувачі самі займаються наповненням, перевіркою та вдосконаленням змісту таких систем. Це суттєво відрізняє сайти, що відносяться до Web 2.0, від традиційних сайтів, зміст яких визначали розробники, а не користувачі. Наприклад, це online-енциклопедії, такі як Вікіпедія, зміст яких наповнюють і перевіряють самі користувачі.

Іншою важливою відмінністю Web 2.0 є mash-up – використання на сайті у якості джерел відомостей інших сервісів Інтернету, за рахунок чого значно змінюється функціональність сайту. Наприклад, інтеграція фотосервісів для подання в Інтернеті фотографій, де містяться геотеги з картографічним сервісом, дозволяє прив’язати такі фотографії до карт.

Наступною важливою характеристикою Web 2.0 є соціалізація, тобто використання розробок, навколо яких створюються співтовариства. Також соціалізація сайту дозволяє користувачеві робити індивідуальні налаштування сайту, створення особистої зони (особисті файли, фото, відео, блоги), щоб користувач відчув власну індивідуальність. Прикладами таких Web 2.0-сайтів є численні *соціальні мережі*, які стали дуже популярними на даний час, наприклад FaceBook, MySpace, ВКонтакте, Одноклассники тощо.

До Web 2.0 відносять також сайти, що містять досить складні програми для зменшення трафіку і покращення інтерфейсу користувача.

Як приклад можна навести поштовий сервіс Gmail. У більш широкому розумінні це є веб-служби, тобто програми, доступ до яких здійснюється через Веб, тобто за допомогою браузера. Наприклад веб-служби Google Docs містять набір потужних вільно поширюваних офісних програм, якими можна користуватися безкоштовно, і певний обсяг пам'яті для користувацьких файлів. Такий підхід дозволяє з будь-якого комп'ютера, з якого є вихід в Інтернет, незалежно від операційної системи і наявності офісного пакету, мати доступ до цих програм і файлів користувача.

Одним з піонерів Web 2.0 є відома фірма Google. Нею і був запропонований термін "Освіта 2.0". У березні 2008 р. фірмою Google була проведена у Москві конференція під назвою "Освіта 2.0". На цій конференції була запропонована формула "Освіта 2.0=Освіта+ Web 2.0", тобто "Освіта 2.0" базується на використанні технологій Web 2.0 в освіті. Розглянемо більш детально аргументацію і міркування прихильників такого трактування терміну "Освіта 2.0" за результатами круглого столу "Освіта 2.0", організованого фірмою Google [180].

На думку А.Наумова термін "Освіта 2.0" припускає існування так би мовити попередньої версії освіти, тобто "Освіти 1.0"? Це освіта, що базується на моделі Яна Амоса Каменського. Бурхливий розвиток суспільства і інформаційних технологій, таких як кіно, радіо, телебачення, мало вплинув на освітні технології. Освітні радіопередачі та телепередачі, кінофільми не стали переворотом у освіті. А вчитель, дошка та крейда залишилися. Освіта була і залишається едукоцентричною, тобто вчитель є основним, а у багатьох випадках і єдиним джерелом знань для учнів.

Останнім часом все більше говорять про кризу сучасної освіти. І ця криза пов'язана, насамперед, з розвитком постіндустріального суспільства, основними ресурсами якого стають інформаційні ресурси. Засадою успіху сучасної людини і самого суспільства стає система неперервної освіти, здатність ефективно опрацьовувати різноманітні повідомлення, працювати у колективі. Для сучасного випускника важливим є вже не лише обсяг знань,

отриманих у вищому навчальному закладі, а і здатність відшуковувати потрібні повідомлення в їх бурхливому потоці. Інформаційні потоки змінилися і основою опрацювання повідомлень є вже не комп'ютери, а телекомунікаційні технології.

В зв'язку з цим перспективи використання інформаційних технологій в освіті пов'язуються з комунікаціями і насамперед з Інтернетом як глобальним середовищем інтерактивного навчання. Але сам Інтернет нікого не навчить, більше того, існує небезпека "захлинутися" у морі всеможливих повідомлень, неможливості відшукати потрібні повідомлення і дані серед інформаційного "сміття". І тут постає проблема формування нового вчителя. На перше місце виходить готовність викладача вчитися разом зі своїми учнями, не відставати від них у освоєнні інформаційно-комунікаційних технологій. Щоб мати авторитет у учнів чи студентів, викладач повинен постійно доводити, що не відстає від технологій, які постійно змінюються, вдосконалюючись. І це призводить до перегляду професійної підготовки педагогів.

Разом з цим не слід забувати і слова Н.Вінера "Важко розраховувати на хорошу ідею, маючи слабкі знання предмету, і ще важче розраховувати на таку ідею, не маючи ніяких знань", а також слова славетного Блеза Паскаля "Випадкові відкриття роблять лише добре підготовлені уми".

Елементи "Освіти 2.0" можна віднайти у освітніх технологіях вже зараз. Яскравим прикладом є дистанційна освіта. Зараз дуже широкого розповсюдження набула вільно поширювана оболонка освітнього середовища Moodle, що бурхливо впроваджується у великій кількості навчальних проектів. Але вже зараз, не зважаючи на елементи соціально-конструктивістського підходу, цей інструментарій не повністю відповідає ідеям Web 2.0. Більш сучасним інструментом, наприклад, є середовище *Social Media Classroom (SMC)* [335].

SMC є новим проектом, який започаткував Howard Rheingold на базі відкритих кодів Drupal і використання якого дозволяє викладачам і

студентам створити певне соціальне середовище. SMC включає в себе форуми, блоги, вікі, чат, соціальні закладки, RSS, мікроблогінг, віджети, відеоконференцзв'язок та багато іншого.

Коротко пояснимо суть деяких найбільш нових з вище названих термінів, які відносяться саме до Web 2.0 на основі відомостей з [347].

Блог (мережевий журнал) – веб-сайт, основний зміст якого – записи, що регулярно додаються, а також зображення та мультимедійні засоби. Ці записи відсортовані у зворотному хронологічному порядку. Блоги як правило публічні і припускається наявність сторонніх читачів, що можуть вести полеміку з автором.

Вікі – веб-сайт, структуру і зміст якого користувачі можуть спільно змінювати за допомогою інструментів, що надаються на цьому самому сайті. Найбільш відомим прикладом є Вікіпедія.

Соціальні закладки – це продовження і логічний розвиток ідеї закладок у браузері. На відміну від браузера, соціальні закладки зберігаються не на локальному комп'ютері, а на сервері в мережі Інтернет. Це дозволяє, наприклад, використовувати їх іншими користувачами для обміну адресами цікавих сторінок Інтернету.

RSS, або стрічка новин, як правило містить короткий опис нових повідомлень, що з'явились на сайті, і посилання на її повну версію. Доступ до стрічки новин можна здійснювати за допомогою сучасних браузерів, крім того існують спеціальні програми (RSS-агрегатори), за допомогою яких збирають і опрацьовують повідомлення з RSS-каналів.

Мікроблогінг – це форма блогів, записи яких представляють собою короткі текстові повідомлення, і які можуть бути переглянуті і прокоментовані в режимі чату або ким завгодно, або обмеженою групою осіб, які обрані користувачем. Причому обмін повідомленнями може відбуватися через різноманітні інтерфейси, такі як Web, Wap, SMS, ISQ. Найпопулярнішим зараз є мікроблогінг Twitter.

Віджет – в даному контексті невелика програма, за допомогою якої дані, що постійно оновлюються, виводяться прямо на робочий стіл, або ці дані можуть бути легко переглянуті (наприклад після натиснення відповідної кнопки). Наприклад, виведення відомостей про поточну погоду, курси акцій і таке інше.

Таким чином SMC більше, ніж набір нових засобів масового інформування для освітньої галузі. Кінцевою метою проекту є перехід від традиційної освіти, коли превалює передавання навчальних повідомлень від викладача до учня, до спільної діяльності викладача і учнів, спрямованої на здобування знань.

Але існує і інший підхід до розуміння терміну “Освіта 2.0”, а отже, і до напрямів вдосконалення існуючої системи освіти. Цей підхід запропоновано О.М.Гольдіним у [44]. Аналізуючи парадигму Web 2.0, він помітив її схожість з сучасними інноваційними підходами в освіті. Інтернет вже досить давно перетворився в феномен культури, в тому числі і у освітнє середовище. Базові принципи Web 2.0, такі як інтерактивність, mash-up та соціалізація (про ці принципи йшлося вище) не є новими для педагогів-новаторів. О.М.Гольдін підкреслює, що постіндустріальне суспільство, в якому знання переважають над капіталом, домінуючими виробничими ресурсами є інформаційні, а найбільш цінними якостями працівника є рівень освіти, здатність навчатися і підвищувати кваліфікацію протягом всього життя, креативність, вступають у протиріччя з класно-урочною освітньою системою. Тому за О.М.Гольдіним “Освіта 2.0” – це сукупність таких базових принципів і основаних на них систем, які адекватні призначенню і завданням освіти в постіндустріальну епоху – створенню умов для найбільш повного розкриття особистісного потенціалу кожного учня, розвитку в нього особистої заповзятливості, навичок самоосвіти, вміння приймати відповідальні рішення в ситуації вибору.

Як і в Web 2.0, в “Освіті 2.0” О.М.Гольдіним виділено *три базових принципи*: *суб’єктність*, *надлишковість* і *співробітництво*.

1. *Принцип суб'єктності.* В "Освіті 2.0" змістом освіти є діяльність, що направлена на удосконалення особистої картини світу учня в тій чи іншій предметній галузі. Таким чином в "Освіті 2.0" не існує складених завчасно, у відриві від учнів, навчального плану та програм. Зміст освіти завжди суб'єктивний, тобто формується не авторами програм, а самими учнями в процесі їх руху у світі великої культури за індивідуальними освітніми траєкторіями. Знання тут розуміється як надбудова над власним досвідом, продукт психічної діяльності стосовно осмислення і структуризації цього досвіду. Школа ж – це спеціально організований простір для обміну особистими знаннями учасників освітнього процесу. Підручник же із "носія об'єктивних відомостей" перетворюється лише в один з елементів освітнього середовища.
2. *Принцип надлишковості.* Цей принцип є необхідним для реалізації принципу суб'єктності. Особисті знання учнів не формуються за освітньою програмою, а розвиваються в спеціально організованому надлишковому освітньому середовищі. Під надлишковістю тут слід розуміти насичення освітнього простору носіями всеможливих відомостей – різновіковий склад учнів, наявність різноманітної літератури (а не тільки підручників), можливість роботи з експертами (не обов'язково професійними педагогами), з телекомунікаційними мережами (Інтернет, локальні електронні ресурси), організація предметно-практичної діяльності (робота з лабораторним обладнанням, реальна продуктивна діяльність тощо). Навчально-пізнавальна діяльність в такому освітньому середовищі дає можливість кожному учневі накопичити необхідний для розвитку особистого знання досвід діяльності, побудувати власну освітню траєкторію. Завдання ж педагога полягає в організації різноманітної діяльності учнів у освітньому середовищі.

3. *Принцип співробітництва.* Цей принцип треба розуміти насамперед як рівноправність учасників освітнього процесу - учнів і вчителів. Викладач є не стільки носієм навчальних повідомлень, скільки рівноправним партнером в навчальних комунікаціях. Він теж вчиться, розвиває себе, для чого йому потрібні комунікації з учнями. Як у викладача, так і в учня повинен бути особистий статус, який динамічно змінюється. Пропонується чотири види такого статусу: відвідувач, клієнт, постійний член групи для занять, експерт. Статус не назначається, а природним шляхом визначається самою освітньою спільнотою, причому у однієї і тієї самої людини може бути багато статусів в різних предметних галузях. Оцінки успішності повинні бути замінені на моніторинг особистих освітніх досягнень у формі відкритих резюме. Причому мова йде не про оцінювання учня викладачем, а про взаємооцінювання досягнень освітньою спільнотою [44].

Використання інформаційних технологій (розподілена робота над документами, обмін електронними повідомленнями, конференції, блоги тощо) в значній мірі сприяє реалізації принципів надлишковості і співробітництва, але не змінюють саму суть освітнього процесу.

Для свого функціонування освітня система, що базується на принципах “Освіти 2.0”, повинна задовольняти двом принципам:

1) відкритість освітньої системи;

2) наявність адекватної моделі знань.

Відкритість освітньої системи – це можливість вільного вибору кожним учасником освітнього процесу групи для занять, об’єктів для пізнання, видів діяльності і партнерів в навчальних комунікаціях.

Відкритій освіті притаманні всі етапи процесу управління розвитком керованих систем: планування траєкторії розвитку (через визначення бажаного стану системи у встановлений або обраний момент часу у майбутньому – термін або горизонт планування) та планових поточних

станів системи (у певні, попередньо встановлені моменти часу, що лежать у межах планового горизонту); аналіз процесу розвитку системи, його характеру (на основі визначення співвідношення її планового і поточного станів); прийняття і реалізація управлінських рішень (щодо приведення поточних станів системи у відповідність до запланованих) [15].

Адекватна когнітивна модель заснована на уявленнях про знання, як психічне утворення, що базується на особистому досвіді індивіда і розвивається під впливом навчальних комунікацій у надлишковому культурно-освітньому середовищі.

Близькими до принципів "Освіти 2.0" О.М.Гольдін вважає освітні системи Селестена Френе (Франція) і Саммерхілл-скул Олександра О'Нілла у Великобританії. В США, Канаді, Німеччині, Бельгії Нідерландах і Японії успішно функціонує мережа приватних шкіл Садбері Веллі – більше 35 шкіл. На його думку цим принципам відповідає і рух за продуктивну освіту.

Аналізуючи ці два підходи, можна побачити в них багато спільного, наприклад певна зміна ролі вчителя, нагальна потреба постійного його самовдосконалення, більш активне спілкування з учнями, співробітництво з ними. Пропозиції О.М.Гольдіна є більш радикальними і спірними. Слід зауважити, що впровадження елементів "Освіти 2.0" більш реальне для вузів, ніж для шкіл, насамперед тому, що студент є більш розвиненим, відповідальним і вмотивованим для вибору власної "траєкторії освіти", і це вже частково діє в зв'язку з введенням кредитно-модульної системи формування відповідних систем компетентностей майбутніх фахівців. Хоча в умовах України кредитно-модульна система певним чином вступає в протиріччя з пануючою прив'язаністю студента до академічної групи і дуже малою можливістю вибирати курси, викладачів, тобто будувати "власну траєкторію освіти". Дистанційна освіта теж вже широко впроваджена в багатьох ВНЗ і її вдосконалення за рахунок елементів "Web 2.0" тільки питання часу. В деяких вузах можна побачити і оцінювання

студентами лекцій (курсу) за критеріями актуальності навчального матеріалу, форми подання знань. Важливою перспективою є можливість вибору студентом курсу з більшим рейтингом. Більші перспективи у вузів і до динамічності знань, модифікації курсів, пристосувань їх до потреб суспільства, ринку. Також дуже важливо, щоб викладач був не тільки педагогом, але і професіоналом у своїй галузі знань.

Не секрет, що у теперішній системі вищої освіти викладач бере участь у навчанні такої кількості студентів, що у нього залишається мало часу на індивідуальні консультації, тому в багатьох іноземних університетах, наприклад, у Гарварді, студенти старших курсів навчають і консультують студентів молодших курсів, отримуючи за це невелику платню [21]. Їх називають teaching assistant або research assistant. Такі студенти займаються і науковими дослідженнями під керівництвом викладачів. Питання реалізації таких способів роботи в українських вузах також актуальні. Тому необхідно провести ще велику роботу щодо впровадження і розвитку наведених вище позитивних тенденцій.

РОЗДІЛ 2
ІНФОРМАЦІЙНЕ МОДЕЛЮВАННЯ УРОЗРОБЦІ ПЕДАГОГІЧНИХ ПРОГРАМНИХ ЗАСОБІВ

2.1. Парадигми програмування та технології програмування

За тлумачним словником парадигма – набір теорій, стандартів та методів, які спільно утворюють спосіб організації наукового знання. Або іншими словами, спосіб бачення світу [78].

Багато різних авторів дають дещо відмінні тлумачення поняттю «парадигма програмування». Так, наприклад:

1. Діоміидіс Спінелліс дає таке означення «Слово парадигма використовується в програмуванні для визначення сімейства позначень (нотацій), через які визначається загальний спосіб (методика) реалізації програми» [345].
2. Деніел Бобров визначає парадигму як «стиль програмування, як опис намірів програміста» [320].
3. Брюс Шрайвер визначає парадигму програмування як «модель чи підхід до розв’язування проблеми» [342].
4. Лінда Фрідман – як «підхід до розв’язування проблем програмування» [327].
5. Пітер Вегнер парадигму визначає, як «правила класифікації мов програмування у відповідності з деякими умовами, які можуть бути перевірені» [348].
6. Тімоті Бадд пропонує розуміти термін «парадигма», як «спосіб концептуалізації того, що означає “виконувати обчислення”, і як задачі, які підлягають розв’язуванню за допомогою комп’ютера, мають бути структуровані і організовані» [321].

Виходячи із сказаного поняття парадигми програмування можна сформулювати наступним чином. *Парадигми програмування* – це

сукупність ідей та понять, через які визначається стиль написання програми.

Парадигма в першу чергу визначається базовою програмною одиницею. В сучасній індустрії програмування дуже часто парадигма програмування визначається набором інструментів програміста, а точніше, мовою програмування і бібліотеками, що використовуються.

Відомо кілька основних парадигм програмування, найважливішими з яких на даний момент є парадигми імперативного, функціонального, декларативного, об’єктно-орієнтованого програмування.

Імперативне програмування – це парадигма програмування, за якою процес обчислення описується у вигляді інструкцій, за якими змінюється стан програми і даних. Тобто це послідовність операцій, які повинні бути виконані над даними за допомогою комп’ютера.

В функціональному програмуванні наголошується на застосуванні функцій, на відміну від імперативного програмування, в якому наголошується на змінах в стані та виконанні послідовностей команд. Іншими словами, функціональне програмування є способом створення програм, в яких єдиною дією є виклик функції, єдиним способом поділу програми є створення нового імені функції та задання для цього імені виразу, за яким обчислюється значення функції, а єдиним правилом композиції є оператор суперпозиції функцій. Жодних комірок пам'яті, операторів надання значень, циклів, чи, тим більше, блок схем чи передавання управління [296].

Декларативне програмування – це парадигма програмування, за якою описують, яким повинен бути результат виконання програми, замість опису впорядкованого набору дій для отримання цього результату. До мов програмування, які базуються на декларативній парадигмі, можна віднести, наприклад мову Пролог, мову SQL.

Об’єктно-орієнтоване програмування – це парадигма програмування, в якому основними концепціями є поняття об’єктів і класів.

Парадигмою програмування визначається, в яких термінах програміст описує логіку програми. Наприклад, в імперативному програмуванні програма описується, як скінчений упорядкований набір дій, а в функціональному програмуванні подається у вигляді виразу і множини визначень функцій. В популярному об’єктно-орієнтованому програмуванні програму прийнято розглядати як набір взаємопов’язаних об’єктів. Об'єктно-орієнтоване програмування є по суті імперативне програмування, яке доповнене принципом інкапсуляції даних і методів в об’єкті.

Важливо відмітити, що парадигма програмування не визначає однозначно мову програмування. Багато сучасних мов програмування є мультипарадигменними, тобто в них допускається використання різних парадигм. Так мовою C++ можна описувати програми як з використанням об’єктно-орієнтованого підходу, так і без використання, а мовою Ruby, в основу якої в значній мірі покладена об’єктно-орієнтована парадигма, можна описувати програми за стилем функціонального програмування.

Крім парадигми програмування важливою є технологія програмування.

Технології програмування – це система методів, засобів і прийомів розробки і налагодження програм. Технологія програмування розуміється як технологія розробки програмних засобів, включаючи всі процеси від моменту зародження ідеї програмного засобу. Використання ефективної технології програмування дає можливість суттєво підвищити продуктивність праці програмістів та якість програмних продуктів.

Серед технологій програмування можна виділити структурну, модульну та об’єктно-орієнтовану.

В основі *структурної технології програмування* лежить подання програми у вигляді ієрархічної структури блоків. У відповідності з даною методологією:

1) будь-яка програма є структурою, побудованою з трьох типів базових конструкцій (послідовне виконання, розгалуження, цикл);

2) фрагменти програми, що повторюються, можуть оформлюватися у вигляді так званих підпрограм;
3) розробка програми відбувається покроково, за методом «згори донизу».

Модульна технологія програмування полягає в поділі програми на логічні частини, які називають програмними модулями. Програмний модуль – це будь-який фрагмент опису процесу, оформлений як самостійний програмний продукт, який можна використовувати в описах процесу.

Об'єктно-орієнтована *технологія програмування* полягає в поданні програми у вигляді сукупності об'єктів, кожен з яких є екземпляром певного класу (класи утворюють ієрархію наслідування) і оснащений інтерфейсом у вигляді набору методів для взаємозв'язків один з одним.

2.2. Підходи до створення математичних ППЗ.

Майже одночасно з появою в школі комп'ютерів почали створюватися комп'ютерні програми, призначені для комп'ютерної підтримки навчання школярів програмування. Потім з'явилися програми для комп'ютерної підтримки навчання інших предметів. За такими програмами та інструментальними засобами закріпився термін „педагогічні програмні засоби".

Педагогічні програмні засоби (ППЗ) – це сукупність комп'ютерних програм, призначених для комп'ютерної підтримки навчання та досягнення конкретних навчальних цілей [30]. ППЗ є головною частиною комп'ютерно-орієнтованого програмно-методичного комплексу, який включає в себе, крім ППЗ, методичні та дидактичні матеріали і настанови, розраховані на використання вказаних програм для комп'ютерної підтримки навчально-пізнавальної діяльності.

Проте основною проблемою використання ППЗ в навчанні є те, що дуже часто програмні засобі спочатку створюються, а потім розпочинаються способи знайти їм місце в навчальному процесі. Ці

програми створюються частіше за все не так, як це потрібно учневі та вчителеві, а так, як це зручно і зрозуміло розробнику. Так наприклад, чи не першими ППЗ стали численні „автоматизовані навчаючі системи” – просто тому, що подібні програмні засоби використовувалися на підприємствах і в момент появи комп’ютерів в масовій школі стали актуальними задачі автоматизації деяких робіт та створення „автоматизованих робочих місць”. Зручність і простота розробки викликали появу великої кількості електронних задачників – текстів з навчальним матеріалом, які супроводжувалися в кращому випадку графічними ілюстраціями; багато контролюючих програм, в яких зазвичай потрібно з кількох запропонованих відповідей вибрати правильну.

В зв’язку з цим можна вказати на системи, інколи досить потужні, призначені для розв’язування навчальних задач. Основний їх недолік в тому, що в таких системах незрозуміло, навіщо взагалі потрібен учень – майже все за нього робиться автоматично за програмою.

В іншому випадку, коли розробкою ППЗ займається вчитель, мало обізнаний з особливостями і можливостями застосування мов програмування, програми виявляються досить недосконалими і “безпорадними” з точки зору характеристик вибраного середовища програмування. В них використовується комп’ютер не завжди ефективно, проте найчастіше в їх основу покладена грамотна методика навчання предмету і програми часто дидактично досить коректно обґрунтовані.

Разом з тим при розробці методики навчання кожного навчального предмета в свою чергу враховуються особливості відповідної науки, тому правомірно говорити про методичні вимоги до ППЗ, в яких враховується специфіка конкретної предметної галузі і відповідного їй навчального предмету. Визначаючи педагогічні вимоги, яким повинні задовольнятися ППЗ, необхідно враховувати також особливості добору змістового наповнення ППЗ, аргументованого певними методичними цілями, і забезпечувати перевірку педагогічної ефективності використання ППЗ.

Можна з впевненістю сказати, що лише невелика частина існуючих ППЗ можуть ефективно використовуватися при навчанні різних предметів у школі.

Тому є необхідність навчати студентів відповідних спеціальностей у педагогічних вузах не просто програмувати, а і створювати ППЗ різного призначення.

Проте з іншого боку перш ніж створювати ППЗ, студент повинен вміти програмувати. Як писав Чарльз Уезерелл [294], навчання програмування – справа майже безнадійна, його вивчення – непосильна праця. Викладач може по-різному займатися зі студентами, читати лекції, робити критичні зауваження, направляти певним шляхом. Студент може все ретельно записувати, запам’ятовувати, читати, складати заліки, дискутувати. Проте всі зусилля марні, якщо студент не буде практикуватися в написанні програм, оскільки навички програмування набуваються тільки у практиці (це стосується і розв’язування різноманітних задач з математики, інформатики, фізики та ін.). Більше того, вчитися потрібно на “справжніх” програмах, а не на дуже спрощених прикладах, якими переповнені підручники з програмування. Особливо актуальними є ці думки стосовно вивчення об’єктно-орієнтованого програмування (ООП), оскільки ООП є інструментом для створення, перш за все, великих програм. Тільки в процесі написання великої програми студент може по-справжньому навчитися моделювати об’єкти предметної галузі, встановлювати зв’язки між ними, будувати відповідне дерево класів та програмувати методи цих класів.

Розробка програмного забезпечення – складний і трудомісткий процес. При традиційному, не об’єктно-орієнтованому підході, відбувається лавиноподібне нарощування складності розробки програми [22]. Подолати ці проблеми можна шляхом правильного використання об’єктно-орієнтованого підходу до програмування.

Вбачається два шляхи створення сучасних ППЗ. У першому випадку програмуванням займається професійний програміст (колектив програмістів) при активній участі педагогів – знавців тієї галузі, до якої відноситься ППЗ. В цій ситуації володіння педагогом основами ООП дозволить підвищити ефективність взаємодії його з програмістом, прискорити процес розробки ППЗ. У другому випадку сам педагог, який оволодів компетентностями в галузі ООП, розробляє ППЗ.

Зрозуміло, що на даному етапі досить серйозне вивчення програмування можливе тільки на тих спеціальностях у вищих навчальних закладах, де інформатика є основною спеціальністю, або спеціалізацією. Але тут постає питання, а як же ефективно вивчати основи ООП у педагогічному вищому навчальному закладі, щоб майбутні педагоги могли плідно брати участь у розробці ППЗ? Досвід свідчить, що таке вивчення повинно базуватися на прикладах вже розроблених успішних ППЗ. В цьому випадку студенти не тільки вивчать основи ООП, але і познайомляться з внутрішньою будовою таких проектів, ієрархією класів, особливостями реалізації методів опрацювання математичних даних. З іншого боку недоцільно повністю відкривати програмний код. Це пов’язано з багатьма причинами. За час вивчення таких проектів код неможливо збагнути пересічному студенту. Велика кількість нюансів у реальних класах відволікатиме студентів від основної ідеї. Є проблеми і з авторським правом. Тому дуже бажано, якщо це можливо, на основі реального ППЗ підготувати завдання, що пов’язані з навчальними ієрархіями класів, реалізаціями методів, які посильні для опанування студентами, що дозволить показати принципи і особливості використання ООП при розробці ППЗ. Розробляючи такі ієрархії класів, студенти зможуть набути навички створення реальних проектів, освоять особливості аналізу і опрацювання математичних об’єктів.

2.3. Вимоги до педагогічних програмних засобів.

Загальновідомо, що розробка програмних засобів, які використовуються в навчальних цілях, являє собою дуже складний процес, що вимагає колективної праці не тільки вчителів, методистів, програмістів, але й психологів, гігієністів, дизайнерів. У зв'язку із цим правомірно пред'являти комплекс вимог до створюваних ППЗ, щоб їх використання не призводило до негативних (у психолого-педагогічному або фізіолого-гігієнічному розумінні) наслідків, а служило цілям інтенсифікації навчального процесу, розвитку особистості учня.

Виходячи з цього можна перерахувати основні вимоги, які ставляться до ППЗ:

- педагогічні вимоги (дидактичні, методичні, обґрунтування вибору тематики);
- технічні вимоги;
- ергономічні вимоги;
- фізіологічно-гігієнічні вимоги;
- естетичні вимоги;
- вимоги до оформлення документації.

Зупинимося більш детально на сутності педагогічних вимог, які висуваються до ППЗ.

Дидактичні вимоги до ППЗ. Вимога забезпечення науковості змісту навчання передбачає подання за допомогою ППЗ науково вірогідних відомостей (якщо можливо – за методами досліджуваної науки). При цьому можливість моделювання, імітації досліджуваних об'єктів, явищ, процесів (як реальних, так і "віртуальних") може забезпечити проведення експериментально-дослідницької діяльності, що ініціює самостійне "відкриття" закономірностей перебігу досліджуваних процесів, і разом з тим наближає шкільний експеримент до сучасних наукових методів дослідження.

Вимога забезпечення *доступності* навчального матеріалу означає, що навчальний матеріал, який вивчається з використанням комп’ютерних програм, форми й методи організації навчальної діяльності повинні відповідати рівню підготовки учнів і їхнім віковим особливостям. Встановлення того, чи доступний розумінню учня запропонований за допомогою ППЗ навчальний матеріал, чи відповідає він раніше набутим знанням, умінням і навичкам, відбувається за допомогою тестування. Від отриманих результатів залежить подальший хід навчання з використанням ППЗ.

Вимога *адаптивності* (пристосовуваності ППЗ до індивідуальних можливостей учня) передбачає реалізацію індивідуального підходу до учня, врахування індивідуальних можливостей сприймання навчального матеріалу. Реалізація адаптивності може забезпечуватися різними засобами наочності, кількома рівнями диференціації при поданні навчального матеріалу за складністю, обсягом, змістом.

Вимога забезпечення *систематичності й послідовності* навчання з використанням ППЗ передбачає необхідність засвоєння учнем системи понять, фактів і способів діяльності в їх логічному зв'язку з метою забезпечення послідовності й наступності в оволодінні знаннями, уміннями й навичками.

Вимога забезпечення *комп'ютерної візуалізації навчального матеріалу* за допомогою пропонованого ППЗ передбачає реалізацію можливостей використання сучасних засобів візуалізації (наприклад, засобів комп'ютерної графіки, технології мультимедіа) об'єктів, процесів, явищ (як реальних, так і "віртуальних"), а також їх моделей, подання динаміки їх розвитку, в часовому та просторовому русі, зі збереженням можливостей управління виконанням програми.

Вимога забезпечення *свідомого* навчання, *самостійності й активізації діяльності* учнів передбачає забезпечення на основі використання засобів програми самостійних дій, спрямованих на

здобування навчальних відомостей при чіткому розумінні конкретних цілей і завдань навчальної діяльності. Активізація діяльності учня може забезпечуватися можливістю самостійного управління ситуацією на екрані, вибору режиму навчальної діяльності; варіативності дій у випадку ухвалення самостійного рішення; створення позитивних стимулів, що спонукують до навчальної діяльності, підвищують мотивацію навчання (наприклад, часткове застосування ігрових ситуацій, коректність, тактовність, доброзичливість підказок і повідомлень, використання різних засобів візуалізації).

Вимога забезпечення *міцності засвоєння результатів навчання* передбачає забезпечення усвідомленого засвоєння учнем змісту, внутрішньої логіки й структури навчального матеріалу, що надаються за допомогою засобів ППЗ. Ця вимога досягається здійсненням самоконтролю й самокорекції; забезпеченням контролю на основі зворотного зв'язку, з діагностикою помилок за результатами навчання і оцінюванням результатів навчальної діяльності, поясненням сутності допущених помилок; тестуванням, що констатує просування в навчанні.

Вимога забезпечення *діалогу учнів між собою та вчителем* передбачає необхідність його організації за умови забезпечення можливості вибору варіантів змісту досліджуваного матеріалу, а також режиму навчальної діяльності з використанням ППЗ.

Вимога розвитку *інтелектуального потенціалу* учня передбачає забезпечення: розвитку мислення (аналітичного, алгоритмічного, програмістського, наочно-образного, синтетичного, теоретичного); формування вміння приймати оптимальні рішення або варіативні рішення в складній ситуації; формування вмінь опрацювання різноманітних матеріалів, відомостей, повідомлень (наприклад, на основі використання систем опрацювання даних, інформаційно-пошукових систем, баз даних).

Методичні вимоги до ППЗ включають необхідність: враховувати своєрідність і особливості конкретного навчального предмета, специфіку

відповідної предметної галузі, її понятійного апарату, особливості методів дослідження відповідних закономірностей; реалізації сучасних методів опрацювання даних.

Вибір тематики навчального предмета (курсу) при розробці ППЗ необхідно обгрунтувати педагогічною доцільністю використання такого ППЗ в навчальному процесі.

Ергономічні вимоги до змісту й оформлення ППЗ обумовлюють необхідність:

- враховувати вікові й індивідуальні особливості учнів, різні типи організації нервової діяльності, різні типи мислення, закономірності відновлення інтелектуальної й емоційної працездатності;
- забезпечувати підвищення рівня мотивації навчання, позитивні стимули при роботі учня з ППЗ (доброзичлива й тактовна форма звертання до учня, можливість кількаразового звертання до програми у випадку невдалої спроби, можливість в разі необхідності використання в програмі ігрових ситуацій);
- встановлювати вимоги до подання матеріалу на екрані (кольорова гама, розбірливість, чіткість зображення), до ефективності зчитування зображення, до розташування тексту на екрані ("віконне", табличне, у вигляді тексту, що заповнює весь екран, і т.д.), до режимів роботи з ППЗ.

Естетичні вимоги до ППЗ обумовлюють: відповідність естетичного оформлення функціональному призначенню; відповідність кольорового колориту призначенню ППЗ і ергономічним вимогам; упорядкованість і виразність графічних і образотворчих елементів ППЗ.

Окремі елементи ергономічних та естетичних вимог обумовлюють створення зручного та зрозумілого інтерфейсу користувача.

В *програмно-технічних вимогах* до ППЗ передбачаються: забезпечення стійкості до помилкових і некоректних дій користувача; мінімізація часу у відповідь на дії користувача; ефективне використання

технічних ресурсів (у тому числі й зовнішньої пам'яті); відновлення системи перед завершенням роботи програми; захист від несанкціонованих чи некоректних дій користувача; відповідність функціонування ППЗ опису в експлуатаційній документації.

У *вимогах до оформлення документації* на розробку й використання ППЗ встановлюється єдиний порядок побудови й оформлення основних документів на розробку й використання ППЗ, що створюються в установах і організаціях незалежно від їх відомчої належності.

2.4. Критерії якості ППЗ

Для ППЗ, як і будь-якого іншого ПЗ, можна застосовувати такі критерії визначення його якості:

- функціональність,
- надійність,
- зручність і легкість застосування,
- ефективність,
- мобільність.

Функціональність – це придатність ПЗ до ефективного використання при виконанні різноманітних функцій відповідно до завдань чи можливих потреб користувачів, в межах галузі застосування ПЗ.

Надійність ПЗ – це його властивість бути безвідмовно застосовним при виконанні визначених функцій за заданих умов протягом заданого періоду часу з достатньо великою надійністю. При цьому під відмовою в ПЗ розуміють виникнення помилкових результатів і ситуацій під час його використання. В надійному ПЗ не виключається наявність помилок, однак важливо, щоб ці помилки при практичному використанні даного ПЗ за заданих умов проявлялись достатньо рідко. Впевнитися, що ПЗ відповідає даній вимозі можна шляхом тестування, а також при його практичному застосуванні.

Зручність і легкість застосування – це характеристика ПЗ, на основі якої забезпечується мінімізація зусиль користувача щодо підготовки

вхідних даних, у використанні ПЗ і аналізі отриманих результатів, а також позитивність емоцій користувача при роботі з ПЗ.

Ефективність – це відношення обсягів і рівня послуг, що надаються користувачеві за допомогою ПЗ, до обсягів використаних ресурсів.

Мобільність – це придатність ПЗ до перенесення з однієї операційної системи до іншої без особливих зусиль стосовно переналаштувань.

До критеріїв якості потрібно віднести також виконання специфічних вимог, що стосуються ППЗ даного класу.

2.5. Педагогічний програмний засіб Gran1 для комп'ютерної підтримки математичного моделювання

На основі розглянутих вище підходів щодо створення математичних ППЗ, вимог до педагогічних програмних засобів, критеріїв якості ППЗ було розроблено ППЗ Gran1.

2.5.1. Історія створення і призначення ППЗ Gran1. Використання засобів сучасних ІКТ дозволяє на якісно новому рівні вивчати не тільки власне інформатику, але і але і будь-які інші дисципліни, зокрема математику, застосовуючи та розвиваючи навички інформаційного моделювання при розв'язуванні математичних задач за допомогою спеціалізованих пакетів математичних програм. Оволодіння такими математичними пакетами у курсі інформатики дозволить студентам зосередити основну увагу якраз на етапі пошуку адекватних інформаційних моделей задач із відповідної предметної галузі, побудові на їх основі математичних моделей та аналізу цих моделей засобами математичної програми для встановлення числових характеристик відповідних об'єктів, взаємозв'язків між об'єктами, їх візуалізації без проведення громіздких обчислень. Все це спонукує студентів до проведення досліджень, експериментів, здогадок та відкриттів, привносить елементи творчості у їхню навчальну діяльність.

На даний момент набула широкої популярності програма Gran1 для підтримки навчання математики в школі та у вищих педагогічних

навчальних закладах. Перші версії програми розроблялися ще для класів КУВТ-2 Пеньковим А.В. та Горошком Ю.В. Версію для операційної системи MS-DOS розробляв Горошко Ю.В. Версію для операційної системи Windows також розробляв і продовжує вдосконалювати Горошко Ю.В. у співробітництві зі своїми учнями – Вінниченком Є.Ф. та Костюченком А.О. На даний момент програма Gran1 є складовою частиною програмного комплексу “Gran”, авторське право на який мають Жалдак М. І., Вітюк О. В. та Горошко Ю. В. [1]. В даному параграфі наводиться, в основному, опис програми Gran1 з епізодичним використанням найпростіших прикладів її застосування. Приклади використання цієї програми для навчання студентів інформаційного моделювання буде наведено в інших параграфах. Взагалі прикладів використання цієї програми за час її існування накопичилось досить багато, познайомитися з деякими з них можна у [4, 38, 43, 69, 75, 93, 95, 97, 98, 100, 101, 103, 104, 113, 114, 115, 126, 141, 184, 195, 196, 197, 243, 245, 303, 351]. Вивчення програми Gran1 планується у курсі інформатики 11 класів загальноосвітніх навчальних засобів (академічний рівень, профільний рівень) за підручником [126].

В навчальній програмі з математики для загальноосвітніх навчальних закладів, 10-11 класи (академічний рівень) [178] та навчальній програмі з математики для загальноосвітніх навчальних закладів, 10-11 класи (профільний рівень) [179] вказано, що підвищенню ефективності уроків з математики в старших класах сприяє використання програм Gran1, Gran2D, Gran3D, DG, EUREKA, бібліотек електронних наочностей тощо.

В програмі “Gran1” передбачено можливості працювати з багатьма математичними об’єктами:

- Залежностями між змінними x і y (функціями), заданими явно у вигляді $y=y(x)$;
- залежностями, заданими параметрично у вигляді $x=x(t)$, $y=y(t)$;
- залежностями, заданими у полярних координатах у вигляді $r=r(f)$;

- неявно заданими залежностями у вигляді $g(x,y)=0$;
- таблично заданими функціями, для яких будується апроксимуючий поліном;
- ламаними;
- статистичними вибірками.

Також передбачено створення об'єкта «Коло», що описується неявно заданою функцією і має зручний інтерфейс для задання кола через центр і радіус або центр і будь-яку точку на колі.

Для об'єктів всіх типів можна будувати відповідні графіки, проводити різноманітні обчислення та аналіз залежностей, причому з відповідними графічними ілюстраціями. Програма Gran1 почала розроблятися досить давно, в другій половині 80-х років минулого століття, спочатку для операційної системи MS-DOS, а із зростанням популярності Windows була розроблена версія і для цієї операційної системи, яка вдосконалюється до цього часу.

2.5.2. Особливості інтерфейсу програми Gran1. Стандартний віконний інтерфейс – система меню; виведення у вікна, положення і розмір яких можна змінювати; використання контекстних меню (що викликаються при натисненні правої кнопки миші); використання допоміжних панелей.

Практично необмежена кількість об'єктів, які можна ввести.

Робота з *Буфером обміну*. Графіки, вирази та відповіді за допомогою стандартного меню *«Правка»* можна скопіювати до *Буферу обміну* для використання їх в іншій програмі, наприклад для підготовки за допомогою текстового процесора звіту про розв'язок задачі, реферату, статті тощо.

Відповіді записуються до окремого вікна *«Відповіді»*, і зберігаються там до вилучення їх користувачем. За допомогою команд роботи з *Буфером обміну* відповіді можна використати при створенні нових об'єктів та у подальших обчисленнях.

Підтримка чотирьох мов у інтерфейсі програми: української, російської, англійської, польської, причому підтримку інших мов легко додати шляхом створення текстових файлів, що містять значення елементів інтерфейсу та повідомлень програми потрібною мовою. Розширення імені такого файлу повинне бути **LNG**, він повинен знаходитися у папці програми.

Вікна програми

В головному вікні програми є меню для доступу до всіх послуг програми, а також 3 вікна, розмір і положення яких можна змінювати в межах головного вікна (Рис. 2.1).

Використовуючи перемикач *„Виправлення/Авторозміри вікон”* можна розмістити підлеглі вікна без перекривання у головному вікні програми та автоматично змінювати їх розміри при зміні розмірів головного вікна програми. Якщо цей перемикач вимкнути, тоді можна довільно змінювати розміри та положення підлеглих вікон. Звернення до послуги *„Виправлення/Початкові розміри та положення вікон”* дозволяє повернутися до початкових характеристик вікон, запропонованих розробниками програми.

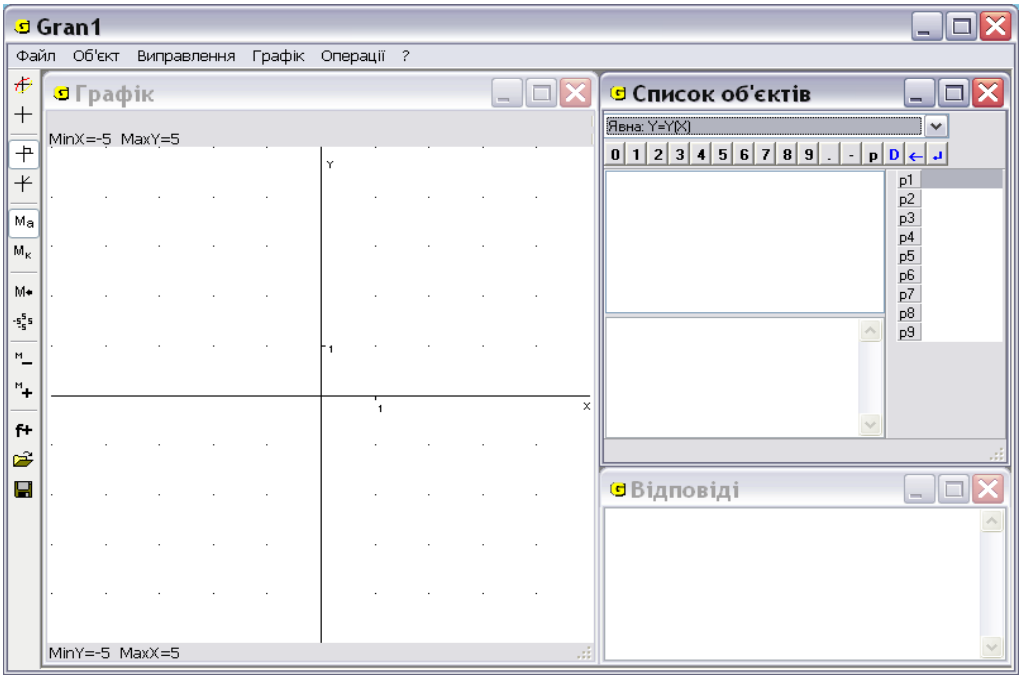

Рис. 2.1. Вікна програми Gran1.

У вікні *«Список об’єктів»* є список для вибору типу об’єкта, що буде створюватися, список введених об’єктів, панель виведення даних про поточний об’єкт (об’єкт, виділений кольором у списку об’єктів), список доступних параметрів об’єктів та панель кнопок, використання яких дозволяє змінити значення поточного параметра (виділеного кольором). Операції за програмою виконуються над поточним об’єктом або над відміченими об’єктами (позначеними відмітками у списку об’єктів). Щоб відмітити об’єкт (або зняти відмітку), треба, скориставшись мишкою, вказати на квадрат зліва від об’єкта (встановити курсор мишки у потрібний квадрат на натиснути її ліву клавішу).

У вікні *«Графік»* будуються графічні зображення об’єктів. При переміщуванні мишки у вікні *«Графік»* автоматично переміщується і покажчик координат (хрестик) та подаються значення координат його місця знаходження у верхній частині вікна. Цей покажчик можна переміщувати і за допомогою клавіш управління курсором, коли вікно *«Графік»* є поточним. За допомогою послуги меню *«Графік/Параметри вікна „Графік”»* можна встановити:

- масштаб вздовж осей (мінімальне і максимальне значення координат на осях);
- вид масштабування (автоматичний масштаб, що визначається за програмою перед кожною побудовою, та масштаб користувача, який залишається незмінним при подальших побудовах);
- зображення осей;
- назви осей;
- зображення розмітки;
- тип координат (полярні чи декартові);
- кольори основних елементів вікна;
- шрифт міток та назв осей.

За допомогою кнопки *«Y->X»* встановлюється мінімальне і максимальне значення координат на осі OX такі самі, як на осі OY, за допомогою кнопки *«X->Y»* – навпаки.

За допомогою перемикача «Графік/Масштаб/Показувати масштаб» можна вмикати або вимикати подання поточного масштабу у вікні «Графік».

Після зміни масштабу можна повернутися до попереднього (послуга «Графік/Масштаб/Попередній масштаб») або початкового (від -5 до 5 на кожній з осей) (послуга «Графік/Масштаб/Початковий масштаб») (Рис. 2.2, Рис. 2.3, Рис. 2.4).

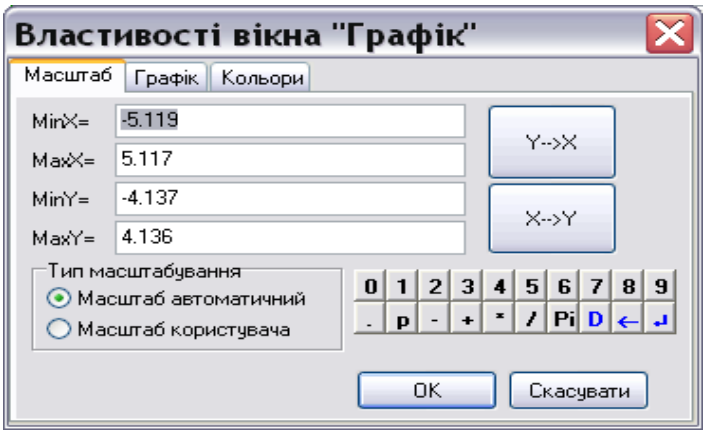

Рис. 2.2. Параметри масштабування.

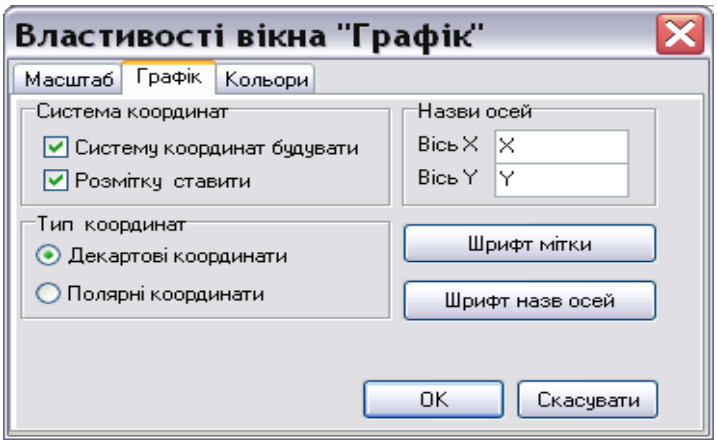

Рис. 2.3. Параметри вікна "Графік".

У вікні «Графік» можна встановлювати довільні текстові мітки. Щоб ввести мітку у позиції курсору, можна скористатися командою „Ввести мітку” із контекстного меню вікна «Графік». Мітка може бути із

зображенням точки біля неї, або без такого зображення. Для цього призначений перемикач у вікні введення мітки. Мітку можна довільно переміщувати в межах вікна *«Графік»* за допомогою мишки. Вилучити непотрібну мітку можна за командою *„Вилучити мітку”* із контекстного меню вікна *«Графік»*.

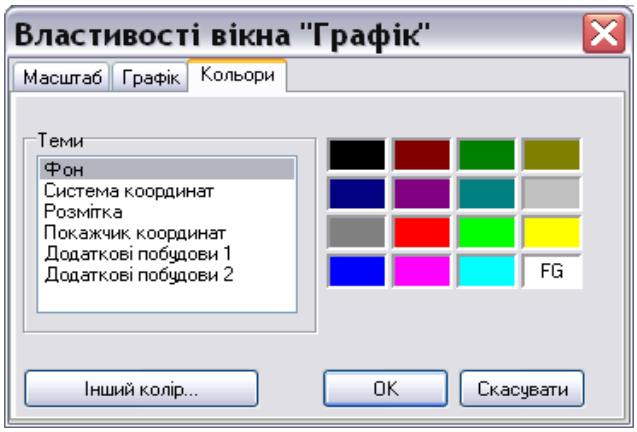

Рис. 2.4. Допоміжна вкладка для вибору кольорів.

Для роботи з мітками також призначена команда *„Графік/Мітки...”*. У вікні, що відповідає цій команді, наведені дані про всі введені мітки, а також є перемикач *„Відображати мітки”*, за допомогою якого можна ховати мітки з екрану без їх вилучення, або знову показувати їх (Рис. 2.5).

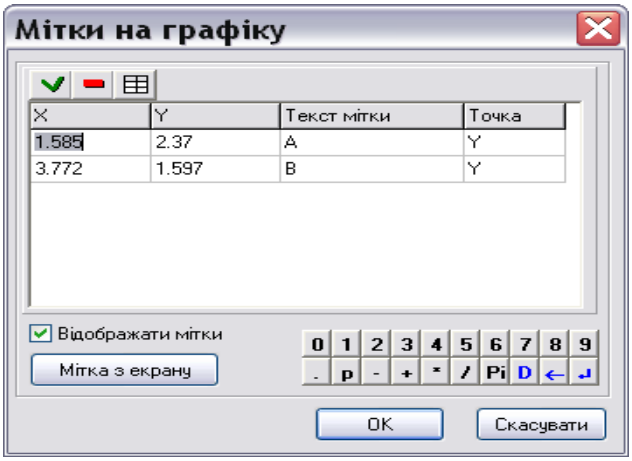

Рис. 2.5. Параметри міток.

Програму оснащено інструментальною панеллю, на якій розміщено кнопки для роботи з вікном *«Графік»* (побудувати, очистити, декартові або полярні координати, масштаб автоматичний або користувача, попередній масштаб, початковий масштаб, зменшити масштаб вздовж обох осей в 2

рази, збільшити масштаб вздовж обох осей в 2 рази), а також кнопки для створення нового об’єкта, відкривання та збереження файлу.

У вікні *«Відповіді»* записуються результати виконання операцій за програмою. При необхідності це вікно можна очистити, звернувшись до послуги *«Операції/Відповіді/Очистити»*.

Робота з Буфером обміну

Робота з *Буфером обміну* відбувається через меню *«Правка»*. Команда *«Правка/Скопіювати»* призначена для копіювання даних до *Буфера обміну.* Якщо поточним є вікно *«Графік»*, до *Буферу обміну* буде скопійовано графічне зображення з цього вікна (без рядка поточних координат та масштабу), якщо поточним є вікно *«Список об’єктів»* – вираз поточного об’єкта, в інших випадках – текстові дані, що відмічені. За командою *«Правка/Вирізати»* забираються до *Буфера обміну* відмічені текстові дані, а за командою *«Правка/Вставити»* вставляються текстові дані з *Буфера обміну* в позицію текстового курсору.

Встановлення загальних параметрів роботи за програмою

Для встановлення загальних параметрів роботи за програмою використовують послугу *„Виправлення/Налагодження параметрів роботи за програмою...”*. Такими параметрами є кількість десяткових знаків у поданні результатів обчислень за програмою та мова інтерфейсу програми (Рис. 2.6).

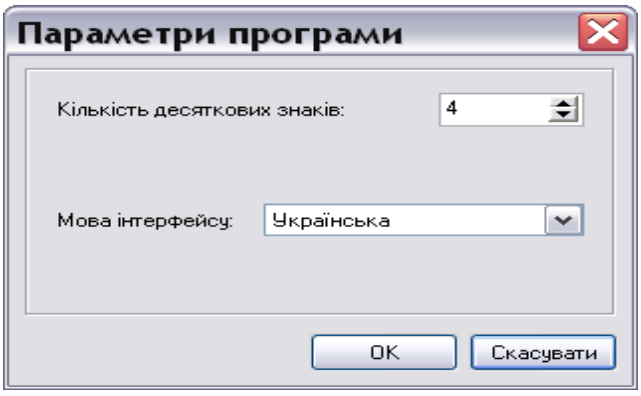

Рис. 2.6. Допоміжне вікно вибору загальних параметрів роботи за програмою.

Створення об'єктів

Для створення нового об'єкта необхідно встановити його тип (обравши тип у списку типів, що знаходиться у верхній частині вікна *«Список об'єктів»*) і звернутися до послуги *«Об'єкт/Створити»*. Після цього на екран виводиться відповідна до типу допоміжна панель. Також можна скористатися командою *"Створити"* із контекстного меню вікна *«Список об'єктів»*.

Задання залежностей між змінними

Для задання залежностей між змінними (виду $y=y(x)$, у полярних координатах, заданих параметрично чи неявно) використовується допоміжна панель для введення виразів залежностей (Рис. 2.7).

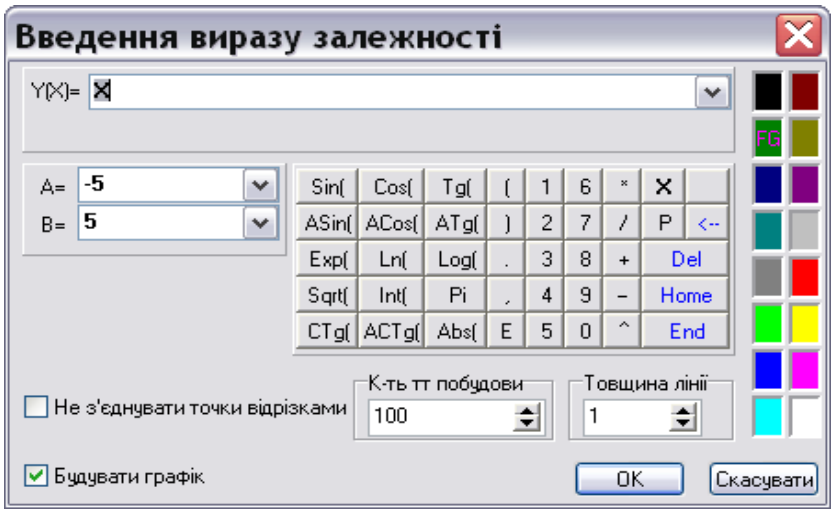

Рис. 2.7. Допоміжна панель для введення виразів залежностей.

Ця панель дещо змінюється в залежності від виду задання залежності, що створюється чи редагується. На панелі розміщено кнопки для внесення у вираз назв стандартних функцій програми та кнопки редагування. Крім виразу залежності необхідно також вказати відрізок (область) задання залежності.

При бажанні можна встановити бажаний колір графіка залежності. Якщо перемикач *«Не з'єднувати точки відрізками»* ввімкнено, графік буде зображуватися точками, в іншому випадку графік буде зображуватись відрізками (неперервною лінією).

Поле редагування *«К-ть тт побудови»* призначене для встановлення кількості точок (мінімальної кількості відрізків), яка буде використана для побудови графіка. Також тут можна встановити товщину ліній (точок) якими буде зображуватися графік залежності. Для залежностей, заданих неявно, товщину ліній вказати неможливо.

Якщо для певного об'єкта побудова графіка в даний момент не потрібна, досить зняти позначку з перемикача *„Будувати графік”***.**

Створення кола

Об'єкт „Коло” є частинним випадком об'єкта „Неявно задана залежність” із специфічним способом задання. Для створення об'єкта «Коло» використовується допоміжна панель, на якій є дві закладки відповідно до двох способів задання кола:

За центром і радіусом. Координати центра кола можна ввести з клавіатури, а можна визначити, вказавши центр кола у вікні *«Графік»*. Для цього треба перейти у вікно *«Графік»*, натиснувши кнопку *«Вибір центра кола з екрану»*, вказати за допомогою мишки на потрібну точку у вікні *«Графік»* і натиснути кнопку *«ОК»*, щоб повернутися до допоміжного вікна, при необхідності відредагувати у допоміжному вікні отримані координати. Радіус кола вводять з клавіатури.

За двома точками (центром кола і точкою на колі). Координати точок вводять у відповідні поля допоміжної панелі або визначають ці точки у вікні *«Графік»* (натиснути кнопку *«Вибір точок з екрану»*, вказати центр кола та точку на колі, натиснути кнопку *«ОК»*).

При виборі точок з екрану їх можна переміщувати в межах вікна *„Графік”* за допомогою мишки.

Створення таблично заданої функції

При створенні таблично заданої функції використовується наступна допоміжна панель (Рис. 2.8).

Для створення таблично заданої функції треба сформувати список точок на координатній площині. Оскільки для такої функції формується апроксимуючий поліном, необхідно вказати його степінь.

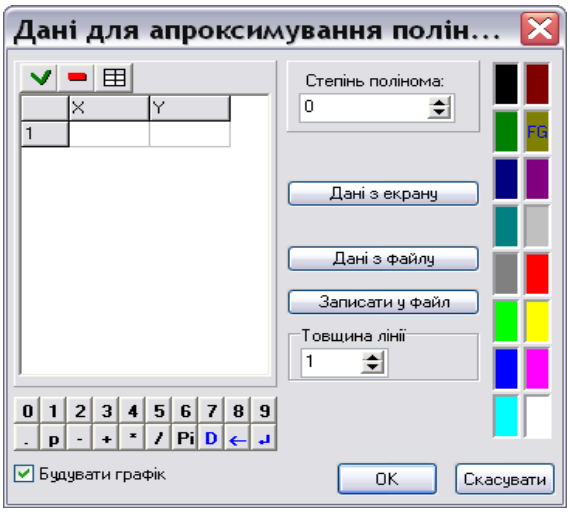

Рис. 2.8. Допоміжна панель для введення таблично заданої функції.

Список точок можна сформувати наступними способами.

Ввести їх координати з клавіатури або за допомогою мишки з використанням числової панелі: 

Завантажити з текстового файлу (числа в файлі повинні розділятися пропусками або починатися з нового рядка), натиснувши кнопку *«Дані з файлу»*.

За допомогою кнопки *«Дані з екрану»* перейти до вікна *«Графік»*, визначити точки, встановлюючи курсор мишки у необхідних місцях та натискуючи її ліву кнопку, і далі повернутися до допоміжної панелі, натиснувши кнопку *«ОК»*. При введенні точок з екрану їх можна переміщувати в межах вікна *«Графік»* за допомогою мишки.

Кнопка  призначена для вставляння в список порожнього рядка (відповідна клавіатурна комбінація – Ctrl-Ins). Кнопка  призначена для вилучення рядка із списку (відповідає комбінації клавіш Ctrl-Del). Кнопка  призначена для очищення списку.

У виразах координат точок можна використовувати параметри. При зміні значень параметрів будуть змінюватися відповідні координати точок

і автоматично переобчислюватиметься апроксимуючий поліном. Якщо на екрані побудовано графік такої таблично заданої функції, він буде автоматично перебудовуватися при зміні параметрів.

Після визначення точок необхідно вказати степінь апроксимуючого полінома (в межах від 0 до 7; степінь добирають, виходячи з умов задачі), бажаний колір графіка, товщину його ліній.

Створення ламаної

Для створення ламаної використовується така сама допоміжна панель, як і для створення таблично заданої функції, але після формування списку точок (вершин) ламаної необхідно вказати, замкнена чи незамкнена ламана. При створенні ламаної в координатах її точок також можна використовувати параметри.

Редагування об'єктів

Для редагування виразів математичних об'єктів (тобто зміни їх характеристик) використовують послугу меню *«Об'єкт/Змінити»* або команду *"Змінити"* з контекстного меню вікна *«Список об'єктів»*. Редагується поточний об'єкт, при цьому на екран викликається та сама допоміжна панель, яка використовувалася при створенні об'єкту.

Вилучення об'єктів

Для вилучення об'єктів використовують послугу меню *«Об'єкт/Вилучити»*. Вилучаються всі відмічені об'єкти. Послуга *«Об'єкт/Вилучити останній»* призначена для вилучення останнього об'єкта із списку об'єктів, а при зверненні до послуги *«Об'єкт/Вилучити поточний»* вилучається об'єкт, виокремлений кольором у списку об'єктів. Команди вилучення продубльовані у контекстному меню вікна *"Список об'єктів"***.**

Збереження та завантаження об'єктів

Для запису у файл виразів всіх математичних об'єктів із списку об'єктів використовують послугу *«Файл/Записати»*. Разом з об'єктами записуються також мітки з вікна *"Графік"*. Якщо збереження відбувається

перший раз, за програмою запитується ім’я файлу. Для збереження об’єктів у файл з новим іменем використовують послугу *«Файл/Записати як…»***.** За допомогою послуги *«Файл/Відкрити…»* можна завантажити до програми об’єкти, збережені у файлі, при цьому всі попередні об’єкти із програми будуть вилучені, і якщо вони не були збережені у файлі, буде запропоновано зберегти їх. Для додавання до існуючого списку об’єктів з файлу використовують послугу *«Файл/Додати…»*. За замовчуванням файли, створені за програмою Gran1, мають розширення *GR1***.** При зверненні до послуги *«Файл/Новий»* вилучаються всі введені об’єкти та мітки. Якщо дані не записувалися, перед вилученням користувачеві буде запропоновано записати їх у файл.

2.5.3. Виконання операцій над об’єктами програми. Доступ до операцій, які можна виконувати над математичними об’єктами, здійснюється через меню *«Операції»*.

Послуга *«Операції/Інтеграли/Інтеграл...»* призначена для знаходження суми інтегралів від відмічених функцій за введеними межами інтегрування.

Послуги *«Операції/ Інтеграли/ Об’єм, вісь Ох...»* та *«Операції/ Інтеграли/ Об’єм, вісь Оу...»* призначені для знаходження суми об’ємів фігур обертання відповідно навколо осей Ох та Оу графіків відмічених функцій за введеними межами інтегрування.

Зауваження. Інтегрувати можна функції виду y=y(x), таблично задані, функції щільності розподілу статистичних ймовірностей для статистичних вибірок з неперервними розподілами статистичних ймовірностей.

Послуга *«Операції/Інтеграли/Довжина дуги...»* призначена для знаходження суми довжин дуг відмічених об’єктів за введеними межами інтегрування.

Зауваження. Знаходити довжини дуг можна для графіків залежностей, заданих явно у вигляді y=y(x), а також заданих таблично, у полярних координатах, параметрично. При обчисленні інтегралів та

довжин дуг можна використовувати параметри. Наприклад, параметри можуть міститися у виразах для меж інтегрування (Рис. 2.9).

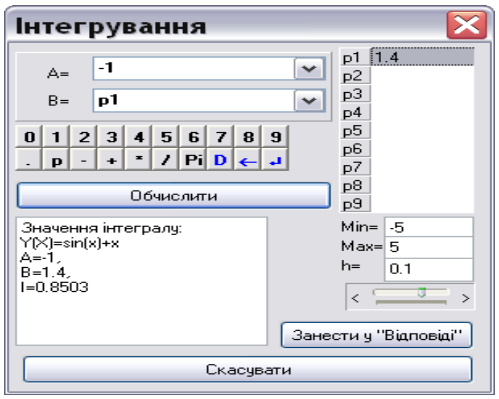

Рис. 2.9. Допоміжне вікно для встановлення параметрів інтегрування.

Змінюючи значення параметра, можна спостерігати, як змінюється площа криволінійної трапеції і відповідні числові значення у допоміжному вікні "*Інтегрування*". За допомогою кнопки *„Занести у Відповіді"* числові дані можна перенести до загального вікна *„Відповіді"*.

Послуга *«Операції/Ламані/Довжина ламаної...»* призначена для знаходження довжини поточної ламаної за введеними номерами початкової та кінцевої вершин (Рис. 2.10).

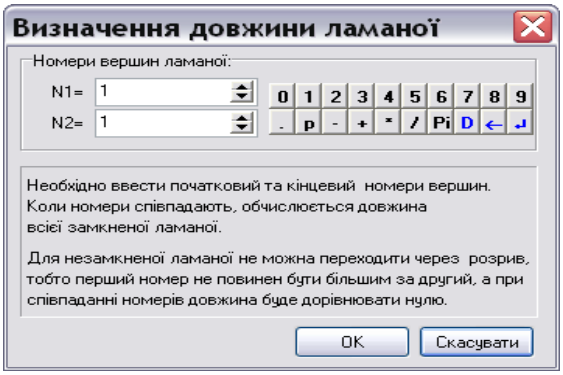

Рис. 2.10. Допоміжне вікно для визначення довжини ламаної.

Послуга *«Операції/Ламані/Площа многокутника»* призначена для знаходження суми площ відмічених многокутників (утворених замкненими ламаними).

Послуга *«Операції/Ламані/Перетворення ламаної...»* призначена для запису до списку об'єктів нової ламаної, створеної шляхом деформації,

повороту або паралельного перенесення поточної (Рис. 2.11, Рис. 2.12, Рис. 2.13).

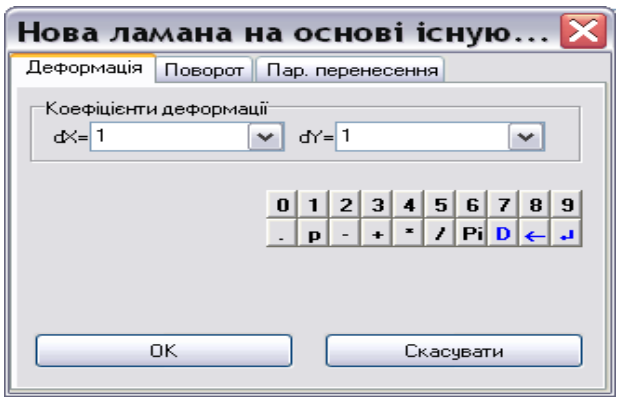

Рис. 2.11. Встановлення параметрів деформації ламаної.

Послуги «*Операції/Ламані/Об'єм та площа поверхні тіла обертання, вісь Ox*» та «*Операції/Ламані/Об'єм та площа поверхні тіла обертання, вісь Oy*» призначені для знаходження суми об'ємів та площ поверхонь фігур, утворених обертанням відмічених ламаних навколо відповідної осі. Ламана не повинна перетинати вісь обертання (Рис. 2.14).

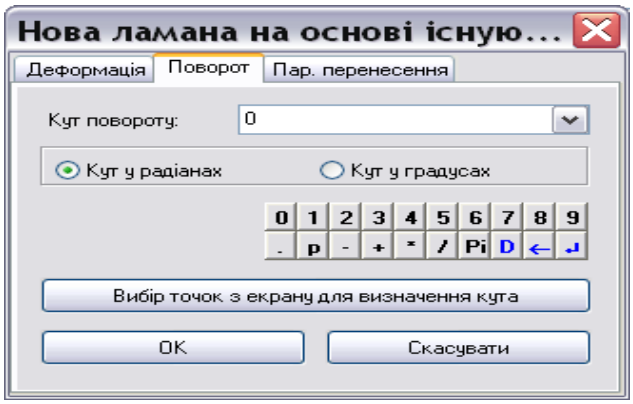

Рис. 2.12. Встановлення параметрів повороту ламаної.

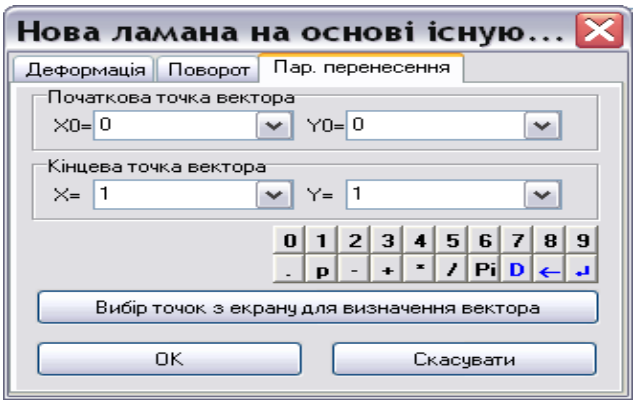

Рис. 2.13. Встановлення параметрів паралельного перенесення ламаної.

Зауваження. У наведених вище послугах, пов'язаних з обертанням ламаної, фігура, що утворюється в результаті обертання, заштриховується (Рис. 2.14).

Послуга *«Операції/Нерівності/С-ма нерівностей y(x)<(>)c...»* призначена для розв'язування вказаної системи нерівностей для відмічених функцій виду *y=y(x)*. Значення константи *c* та знак нерівності вводяться у допоміжному вікні (Рис. 2.15).

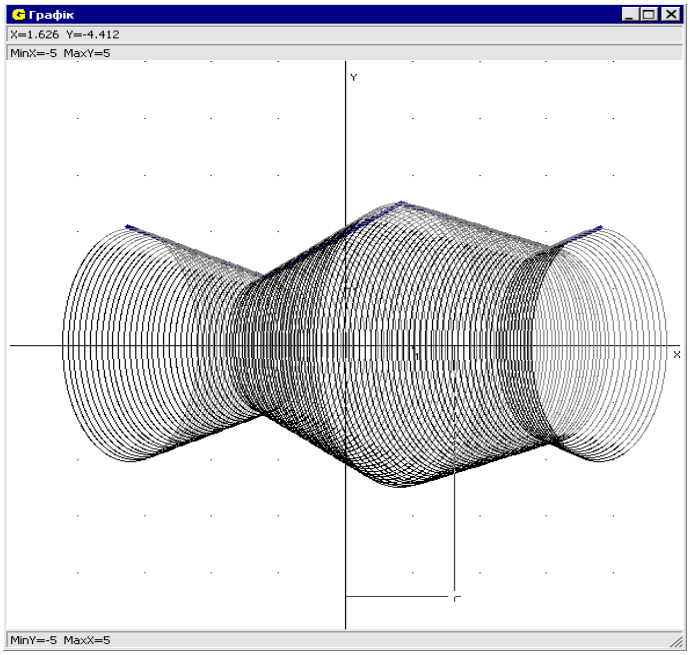

Рис. 2.14. Зображення фігури обертання.

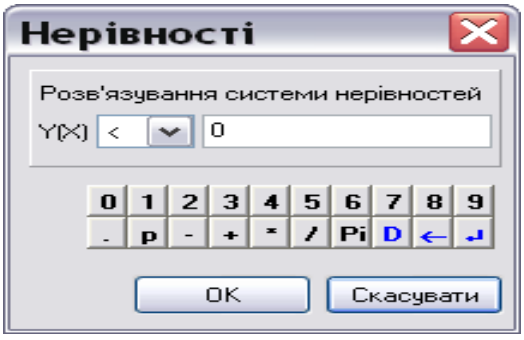

Рис. 2.15. Встановлення параметрів для розв'язування нерівностей.

Корені системи відповідних рівнянь записуються у вікно *«Відповідь»*, а проміжки, на яких система нерівностей задовольняється, зображуються у вікні *«Графік»* (Рис. 2.16).

Послуга *«Операції/Нерівності/С-ма нерівностей G(x,y)<(>)0»* призначена для розв’язування вказаної системи нерівностей для відмічених неявно заданих залежностей виду *0=G(x,y)*.

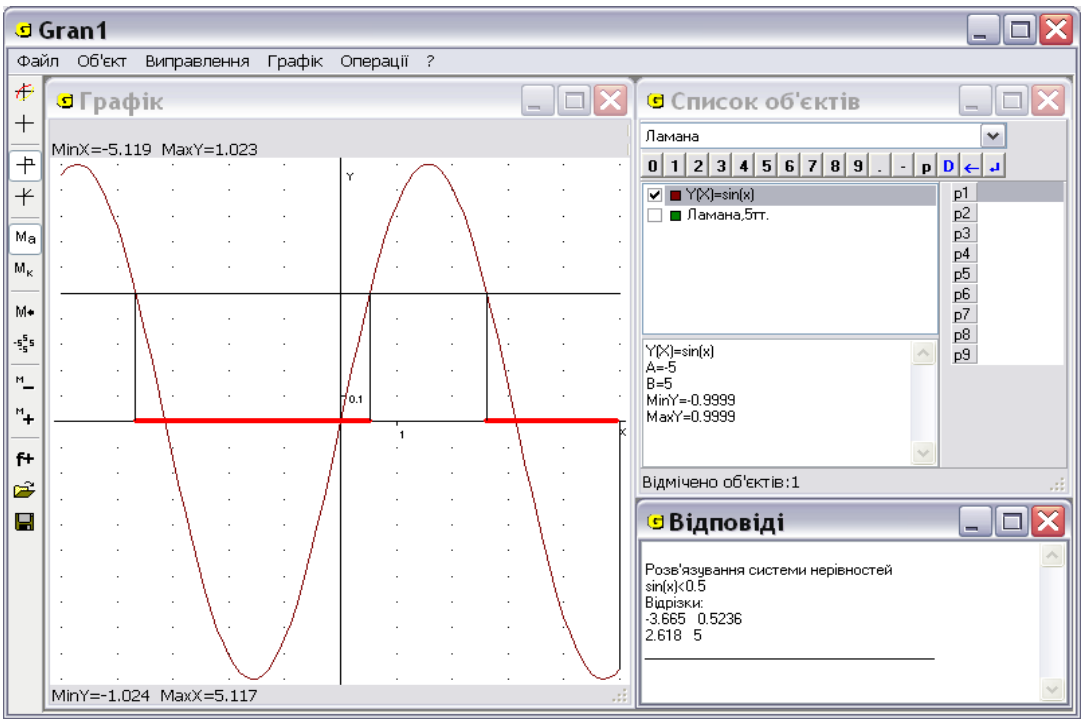

Рис. 2.16. Результат розв'язування нерівності для залежності виду *y=y(x).*

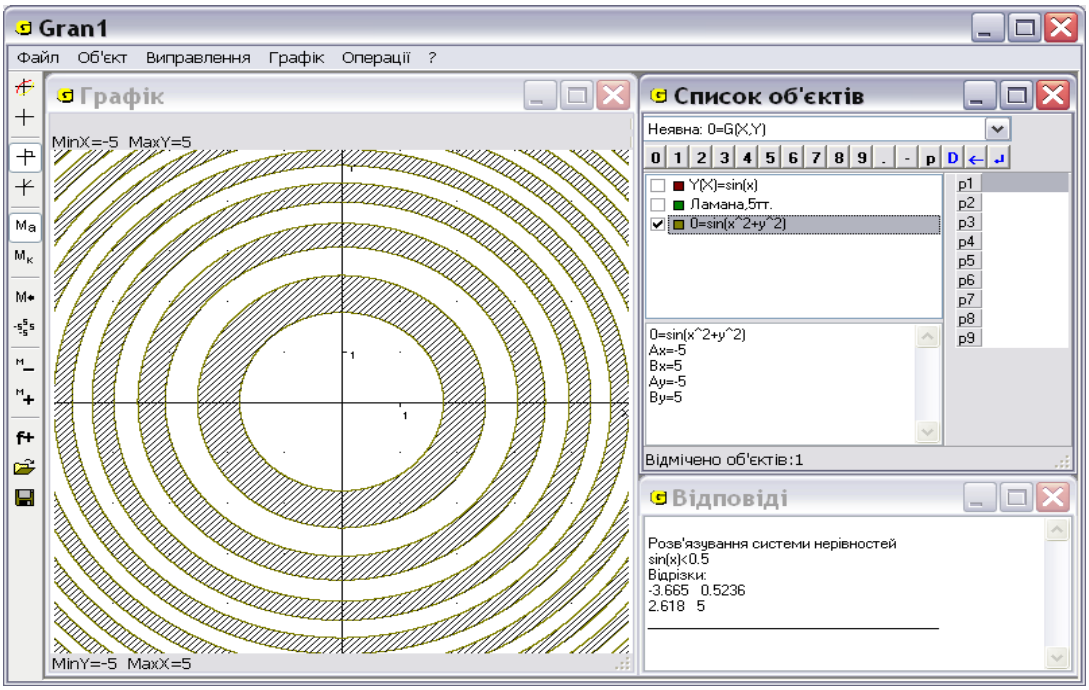

Рис. 2.17. Результат розв'язування нерівності для залежності виду *g(x,y)=0.*

Області, в яких система нерівностей задовольняється, заштриховуються у вікні *«Графік»* (Рис. 2.17).

Послуга *«Операції/Похідна…»* призначена для обчислення приросту функції за введеним приростом аргументу та для обчислення похідної для поточної явно заданої залежності виду *y=y(x)*. Користувач повинен вказати значення аргументу (*X*) та значення його приросту (Δ*X*) (Рис. 2.18). За допомогою перемикачів *„Будувати січну”* та *„Будувати дотичну”* включають чи виключають побудову дотичної та січної для заданих *X* та Δ*X*. Сама побудова виконується після натиснення кнопки *„Побудова”*. Значення аргументу та його приросту, значення функції для цього аргументу та відношення приросту функції до приросту аргументу обчислюється автоматично при включеному перемикачеві *„Будувати січну”***.** Якщо також включено перемикач *„Будувати дотичну”*, додатково обчислюється значення похідної. Всі ці значення виводяться у вікні *„Похідна”* (Рис. 2.18)**.**

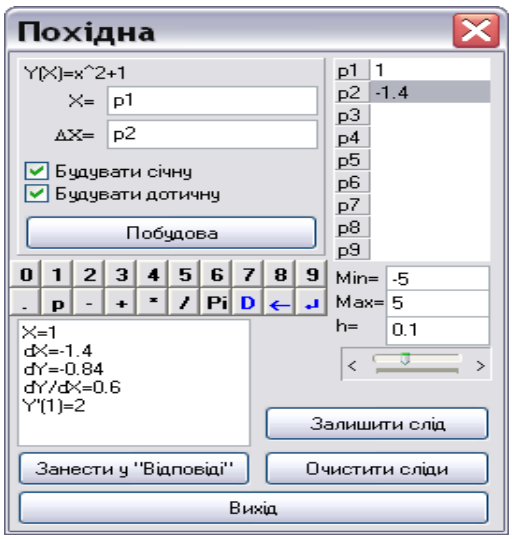

Рис. 2.18. Введення параметрів для знаходження січних і дотичної.

Після натиснення кнопки *„Занести у відповіді”* всі обчислені значення копіюються у вікно *„Відповіді”*.

Натиснення кнопки *„Залишити слід”* призводить до фіксації січної (дотичної) стосовно поточних значень аргументу та його приросту, причому після зміни цих параметрів всі залишені сліди залишаються на

екрані і будуть вилучені тільки після натиснення кнопки *„Очистити сліди”***.** При цьому у вікні *„Похідна”* можна використовувати параметри як у виразі для аргументу, так і у виразі для приросту аргументу. При зміні значення параметра миттєво виконуються відповідні побудови та обчислення.

Послуга *«Операції/Відстань до точки»* дозволяє визначити відстань від поточного положення курсору у вікні *«Графік»* до вказаної точки. Точку, до якої буде вимірюватися відстань, вибирають за допомогою мишки у потрібному місці вікна *„Графік”* (встановивши курсор у потрібну точку та натиснувши ліву клавішу мишки). Початкова точка *(0;0)*. Поточні значення координат курсору та значення відстані до точки *(R)* записуються у верхній частині вікна *«Графік»*. Щоб закінчити роботу з послугою, треба натиснути кнопку *«ОК»*.

Послуга *«Операції/Значення функції G(x,y)»* призначена для знаходження значення функції *G(x,y)*, що визначена як права частина виразу для поточної неявно заданої залежності *0=G(x,y)*. Значення функції обчислюються в точці, в яку встановлено курсор у вікні *«Графік»*. Щоб закінчити роботу з послугою, треба натиснути кнопку *«ОК»*.

2.5.4. Робота з параметрами. Стандартним підходом розв’язування з використанням програми Gran1 прикладів, у яких фігурують параметри, є створення множини об’єктів, для кожного з яких параметр набуває нового числового значення. Аналізуючи одержану множину побудованих графіків, можна робити висновки про розв’язки рівнянь та їх систем. Однак цей спосіб має принаймні два недоліки:

- доводиться створювати достатньо велику кількість майже однакових об’єктів, що розрізняються між собою лише коефіцієнтами при невідомих, тобто значеннями параметра;
- побудова великої кількості об’єктів на одній координатній площині ускладнює сприймання (для покращення сприймання в системах

рівнянь рекомендується графіки різних рівнянь для однакового значення параметра *a* робити одного кольору).

Враховуючи значні методичні переваги динамічних моделей, в останній версії програми Gran1 передбачено побудову об’єктів, у виразах для опису яких використовуються динамічні параметри. При цьому відбулися незначні зміни як в інтерфейсі програми, так і в окремих елементах роботи з нею.

Основних змін зазнав зовнішній вигляд вікна *„Список об’єктів”* (Рис. 2.19, 2.20). Праворуч від списку об’єктів тепер фігурує таблиця із дев’яти елементів з підписами *p1*, *p2*, ... *p9*. Кожен рядок цієї таблиці відповідає одному з *динамічних параметрів*, які можуть бути використані в аналітичних виразах при створенні відповідних об’єктів типів *„Явна: Y=Y(X)”*, *„Параметрична: X=X(T), Y=Y(T)”*, *„Полярна: R=R(F)”*, *„Неявна: 0=G(X,Y)”*, *„Коло”*, *“Таблична: Xi, Y(Xi)”*, *“Ламана”*, *“Статистична вибірка”* .

Якщо жоден з параметрів не використаний – таблиця порожня. Для параметрів, які використовуються, у відповідних рядках таблиці містяться поточні значення цих параметрів. Якщо ж параметр не використаний в жодному з виразів для опису об’єктів, відповідний рядок порожній (Рис. 2.19, Рис. 2.20).

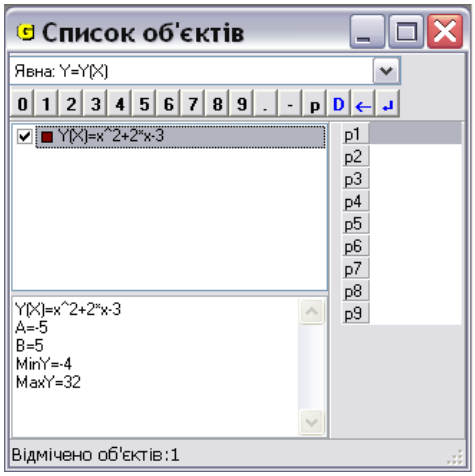

Рис. 2.19. Вікно "Список об'єктів" без визначених користувачем параметрів.

При створенні об’єкту вираз, через який задається залежність між змінними, може містити один або кілька параметрів, наприклад:

*p1*x+3, sin(p1*F+p2), cos(pi/3+p5*x)/(sin(y)-p1).*

Порядок використання параметрів у виразах, що вводяться при створенні відповідних об’єктів, може бути довільним. Після створення об’єкта, вираз для опису якого містить параметр, що ще не використовувався, у відповідний рядок таблиці заноситься початкове значення параметра, рівне 1, а також встановлюються нижня та верхня межі відрізка, на якому відбувається зміна параметра – *Min=-5*, *Max=5*, та крок зміни параметра *h=1*. Для кожного з параметрів при потребі можна встановлювати будь-які значення *Min*, *Max* та *h* у відповідних рядках, дотримуючись умов *Min<Max, h>0*. Якщо параметр вже використовувався у виразах, що вводились при створенні одного з попередніх об’єктів, значення *Min*, *Max* та *h* залишаються незмінними.

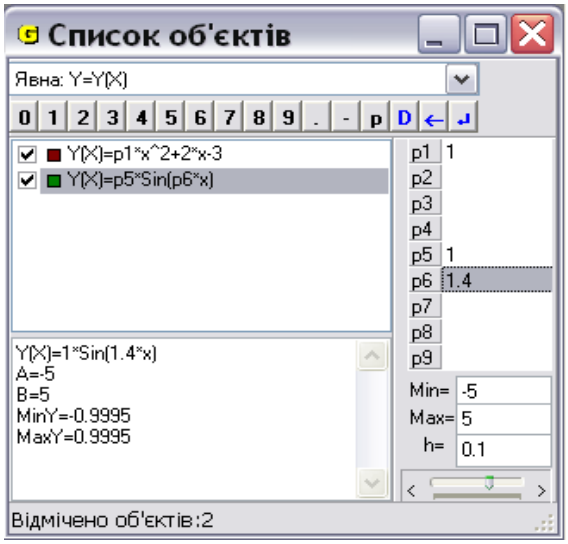

Рис. 2.20. Вікно "Список об'єктів" з визначеними користувачем параметрами.

В нижній частині вікна *„Список об’єктів”* знаходиться бігунець, за допомогою якого можна змінювати значення *поточного параметра*, який в таблиці виділений кольором. Зміну можна проводити за допомогою маніпулятора мишка або за допомогою клавіш управління курсором. Переміщення бігунця на одну позицію вліво або вправо приводить до

зменшення або збільшення значення параметра на приріст *h*. Граничні позиції бігунця відповідають значенням *Min* та *Max*. Якщо в процесі роботи необхідно надати параметру уточнюючого значення, це можна зробити, ввівши значення параметра безпосередньо в таблицю.

При побудові графіків залежностей в аналітичний вираз підставляється поточне значення параметра. Одержаний вираз записується в лівій нижній частині вікна *„Список об'єктів",* де також розміщується значення меж відрізка, на якому задано залежність, а також мінімальне і максимальне значення виразу на цьому відрізку при вказаних значеннях параметрів. Зміна значення будь-якого з динамічних параметрів призводить до того, що графіки всіх об'єктів, описи яких містять цей параметр, перебудовуються. При запису даних у файл на диску записуються також поточні значення параметрів, межі їх зміни та значення приросту. Слід зауважити, що при зміні значення параметра зафіксовуються масштаб відображення у вікні *„Графік"* (встановлюється *„Масштаб користувача"*). Якщо необхідно, масштабування можна знов переключити в автоматичний режим.

Наведемо кілька прикладів розв'язування задач з параметрами [104].

Приклад 1. Розглянемо функцію $y=kx+b$ та дослідимо, як змінюється її графік в залежності від зміни значень коефіцієнтів *k* та *b*.

Для цього в програмі Gran1 створимо новий об'єкт типу *„Явна: Y=Y(X)"* з аналітичним виразом виду $y=p1*x+p2$ на відрізку *[-5, 5]* та побудуємо його графік. Змінюючи значення параметрів *p1* та *p2* можна спостерігати за зміною кута нахилу прямої, її положенням відносно координатних осей.

Приклад 2. Розглянемо функцію $y=ax^2+bx+c$ та дослідимо, як впливають на вигляд графіка значення коефіцієнтів *a, b* та *c*.

Створюємо об'єкт, аналітичний вираз якого *y=p1*x^2+p2*x+p3*. Для кожного з параметрів можна встановити крок зміни *h=0,25*. Змінюючи значення параметра *p1*, що відповідає коефіцієнту *a*, можна побачити, що

він впливає на напрям та „крутизну” гілок параболи; значення параметра *p3*, що відповідає коефіцієнту *c*, впливає на відстань від вершини параболи до осі абсцис. Досить цікавим виявляється той факт, що при зміні параметра *p2* (тобто коефіцієнта *b*, при $a \neq 0$) вершина даної параболи ковзає вздовж іншої параболи, рівняння якої $y = -ax^2 + c$ [104].

Приклад 3. Достатньо ефективно вказані послуги програми можна використовувати при формуванні в учнів уявлень про геометричні перетворення графіків функцій: паралельне перенесення, розтягування або стиснення, симетрію відносно точки та координатних осей – коли графік функції $y=f(x)$ порівнюється з графіками функцій $y=af(x)$, $y=f(ax)$, $y=f(x+a)$, $y=f(x)+a$.

Часто вивчення цієї теми відбувається на прикладі графіка функції $y=x^2$, оскільки учні добре ознайомлені з виглядом цього графіка, властивостями функції, а сам графік досить легко будувати за точками. Разом з цим корисно, використовуючи програму Gran1, як базову розглядати функцію $y=\sin x$. Геометричні перетворення, що відбуваються з графіком цієї функції значно наочніші, ніж для функції $y=x^2$.

Для цього в програмі спочатку створимо об’єкт типу *„Явна: Y=Y(X)”* без використання параметрів, що відповідає базовій функції $y=\sin x$. Графік даної функції буде вважатись за еталон. Далі створюємо чотири об’єкти, що відповідають кожному з перетворень (можна використати як один і той самий параметр для кожного з чотирьох об’єктів, так і різні): $y=p1*\sin(x)$, $y=\sin(p2*x)$, $y=\sin(x+p3)$, $y=\sin(x)+p4$. Кожен з п’яти об’єктів доцільно задати на відрізку $[-2\pi;2\pi]$. Рекомендується перед початком дослідження для всіх параметрів *p1-p4* вказати значення кроку зміни $h=0,1$, а параметрам *p3* та *p4* надати значення 0, що відповідатиме залежності $y=\sin x$.

Змінюючи окремо кожен з параметрів *p1-p4*, можна бачити, як змінюється графік розглядуваної функції у порівнянні з еталоном та зробити відповідні висновки. Наостанок можна створити об’єкт

*y=p1*sin(p2*x+p3)+p4*, що відповідатиме функції *y=Asin(ωx+φ)+B*, та зробити відповідні узагальнення (Рис. 2.21).

Приклад 4. Знайти всі значення параметра *a*, при кожному з яких нерівність $|x+a|+x^2<2$ має хоча б один додатній розв'язок.

Дана нерівність рівносильна нерівності $|x+a|+x^2-2<0$. Створимо об'єкт *„Явна: Y=Y(X)"*, ввівши відповідний вираз з параметром: *y=abs(x+p1)+x^2-2* на відрізку [-5;5]; для параметра *p1* встановимо крок зміни *h=0.05* та побудуємо графік створеного об'єкта.

Нерівність має розв'язки, якщо *y<0*, тобто частина графіка знаходиться нижче від осі абсцис. Враховуючи умову, що нерівність повинна мати хоча б один *додатній* розв'язок, можна зробити висновок: необхідно знайти такі значення параметра *a*, при яких хоча б одна точка на графіку функції $y=|x+a|+x^2-2$ знаходилась у IV координатній чверті. Змінюючи значення параметра *p1* та спостерігаючи за змінами графіка функції, одержуємо відповідь $a\in(-2{,}25;\ 2)$.

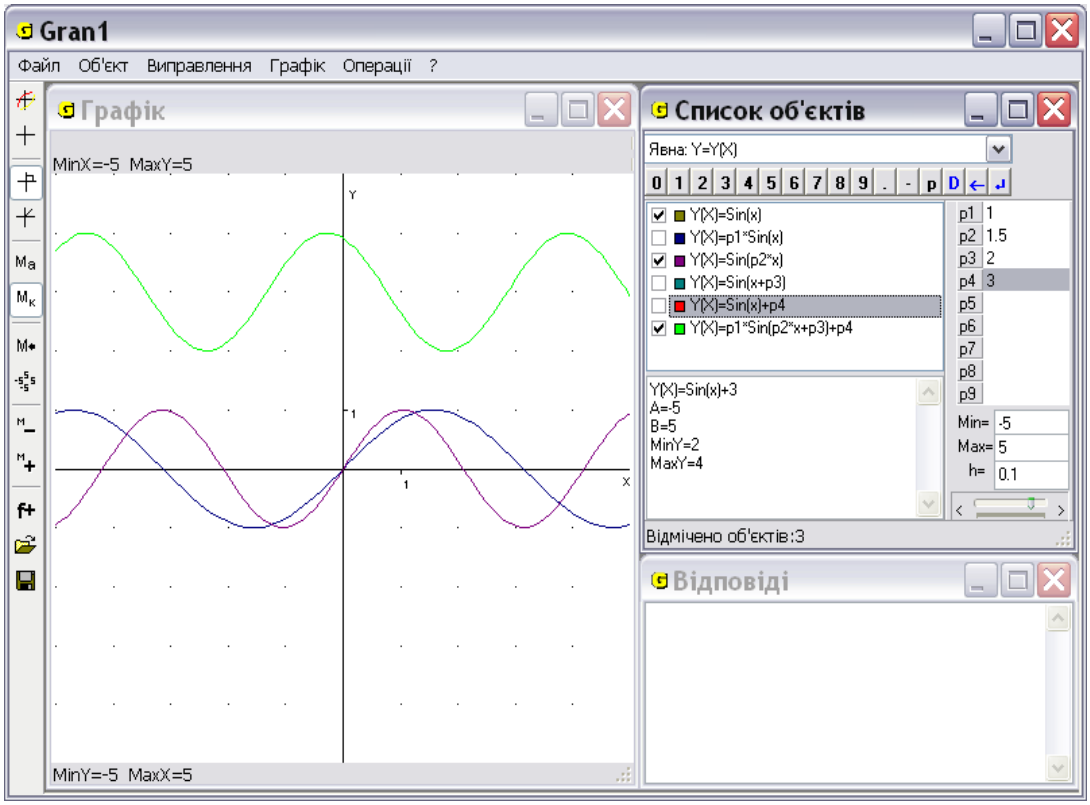

Рис. 2.21. Дослідження властивостей функції *y=a*sin(bx+c)*.

Приклад 5. При яких значеннях параметра a система $\begin{cases} x^2 + y^2 - 2x \le 1, \\ x - y + a = 0 \end{cases}$ має єдиний розв'язок?

Першу нерівність системи перепишемо як $x^2+y^2-2x-1\le 0$, що дає змогу створити відповідний об'єкт в програмі Gran1. Створюємо об'єкт *„Неявна: 0=G(X,Y)"*, ввівши для *G(x,y)* вираз *x^2+y^2-2*x-1*; графіком цієї залежності є коло. Скориставшись послугою *„Нерівності/Система нерівностей G(x,y)<0"*, одержимо графічний розв'язок першої нерівності системи: вся внутрішня частина круга, включаючи граничне коло (оскільки в умові наявний знак нестрогої нерівності).

Для другого рівняння системи можна створити об'єкт *„Явна: Y=Y(X)"* з використанням параметра: *y=x+p1*, графіком якого є пряма (Рис. 2.22). Можна зробити висновок, що система має *єдиний* розв'язок в тому випадку, коли пряма *дотикається* до кола. Змінюючи параметр *a*, одержимо, що пряма дотикається кола при *a=-3*, та при *a=1*.

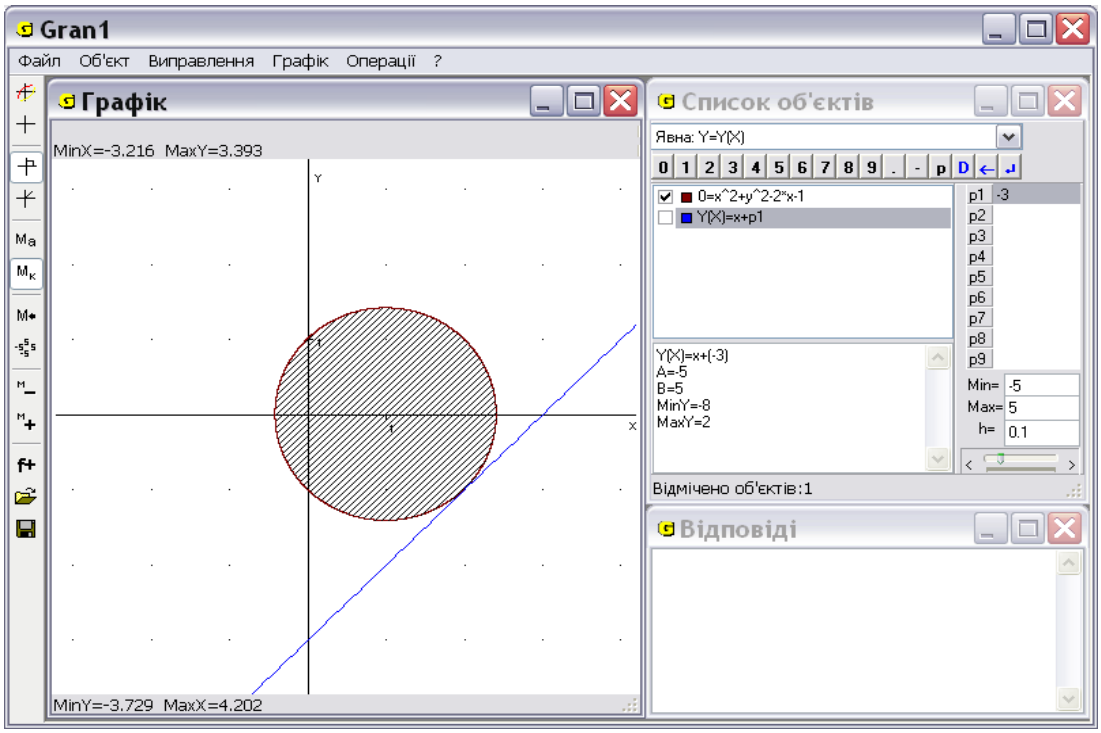

Рис. 2.22. Ілюстрація до прикладу 5.

Для отримання розв'язку задачі довелося створити об'єкт *"Коло"* за допомогою неявно заданої залежності. Побудова графіків таких залежностей вимагає значних математичних обчислень і відповідно займає певний час. Саме тому для неявно заданих залежностей передбачено

можливість встановлювати точність побудов за допомогою бігунця у вікні створення об’єкту: чим вищою вимагається точність побудов, тим більше часу для цього потрібно, і навпаки – чим менш точно будується графік, тим швидше відбувається його побудова. Взагалі відповідність між швидкістю та точністю обчислень добирається для кожного об’єкта експериментальним шляхом, але при побудові графіків достатньо великої кількості залежностей не вимагається велика точність (коло, еліпс, гіпербола тощо), тому при побудові таких об’єктів рекомендується встановлювати більшу швидкість обчислень.

Приклад 6. Досить цікавими для учнів можуть стати експерименти із різноманітними цікавими кривими (епіциклоїдою, гіпоциклоїдою, багатопелюстковою трояндою тощо), рівняння яких подаються в полярній системі координат або задаються параметрично. Змінюючи значення динамічних параметрів, можна відслідковувати зміну вигляду такої кривої, знаходити частинні випадки (кардіоїда, астроїда) [104].

Зокрема, створимо об’єкт типу *„Полярна: R=R(F)”,* задавши вираз із параметром – *R=sin(p1*F)* на відрізку *[0; 20π]*. Кількість точок побудови для функцій, заданих в полярних координатах та параметрично, рекомендується встановлювати 200–400. Для параметра *p1* вкажемо *min=0*, *max=5*, *h=0,1*. Побудувавши графік цієї залежності, одержимо криву, яка при різних значеннях параметра утворює коло, багатопелюсткову троянду тощо [104, 196].

Створивши об’єкт типу *„Параметрична: X=X(T), Y=Y(T)”*, що задається рівняннями *x=cos(p2*T)-cos(T-p2*T)*, *y=sin(p2*T)-sin(T+p2*T)* та вказавши відрізок задання, кількість точок побудови, межі та крок зміни динамічного параметра такими, як і для попередньої функції, отримуємо різні види епіциклоїд (зокрема, при *p2=1* – равлик Паскаля) [100, 103, 104, 196].

Приклад 7. Використання змінних параметрів дозволяє ефективно застосовувати Gran1 для емуляції різного роду фізичних експериментів, що можуть бути описані за допомогою функціональних залежностей.

Наприклад, створимо об'єкт – залежність, задану параметрично у вигляді *x=p1*sin(p2*T+p3)*, *y=p4*sin(p5*T+p6)*, на відрізку *[0; 20π]*. Для всіх параметрів *p1 – p6* можна вказати значення кроку зміни *h=0,1*. Графіками даної залежності при різних значеннях параметрів є добре відомі в фізиці фігури Ліссажу (Рис. 2.23) [100, 103, 104, 196].

Приклад 8. На площині *xOy* знайти точку *M(x, y)*, сума відстаней якої від п'яти точок *A(0, 0), B(4, 0), C(5, 3), D(2, 8), E(0, 5)* була б найменшою (задача Штейнера) [104].

Враховуючи, що сума відстаней точки *M* від точок *A, B, C, D, E* визначається виразом

$$d(x,y)=\sqrt{x^2+y^2}+\sqrt{(x-4)^2+y^2}+$$

$$+\sqrt{(x-5)^2+(y-3)^2}+\sqrt{(x-2)^2+(y-8)^2}+\sqrt{x^2+(y-5)^2}$$

для окремих значень параметра *p1* побудуємо графіки відповідних залежностей виду *d(x, y0)-p1=0*

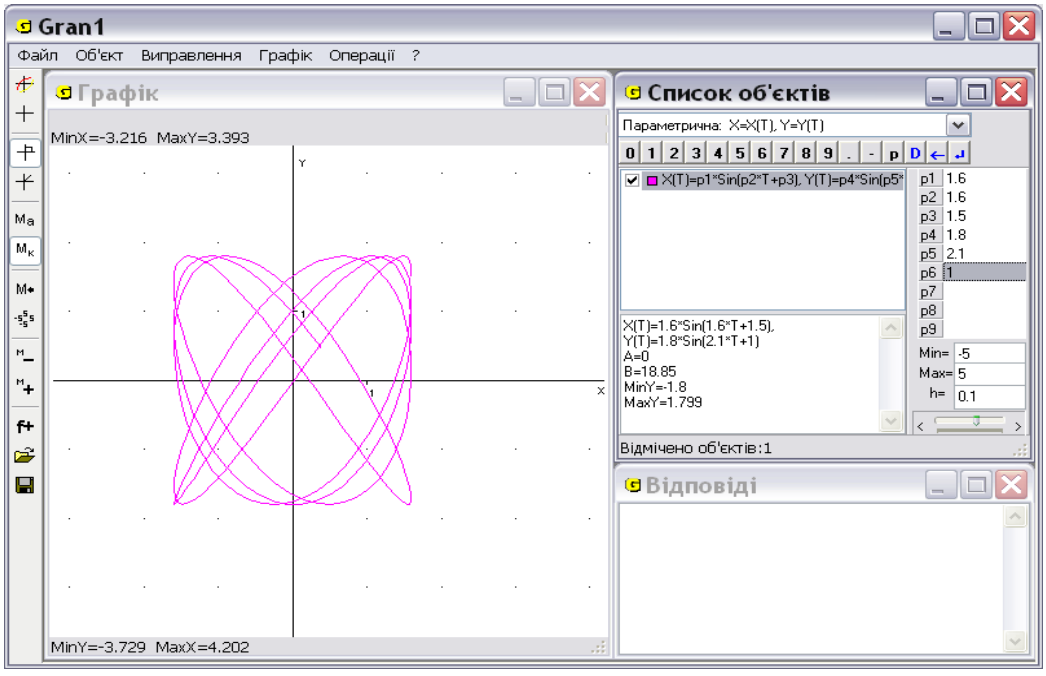

Рис. 2.23. Фігури Ліссажу.

Як видно з Рис. 2.24, найменшого значення, наближено рівного 17.983, розглядувана функція $z=d(x, y)$ набуває в точці $x\approx 2.4,\ y\approx 2.96$ [104].

Приклад 9 [104]. Мандрівник хоче перейти з точки $A(x_0, y_0)$ в точку $B(x_3, y_3)$. При цьому в точках, для яких $x_0 \le x \le x_1$, він може рухатися зі швидкістю V_1, у точках, для яких $x_1 \le x \le x_2$ – з швидкістю V_2, у точках, для яких $x_2 \le x \le x_3$ – зі швидкістю V_3 (мається на увазі, що $x_i\text{-}1 < x_i$, $i=1, 2, 3$).

До яких точок (x_1, y_1), (x_2, y_2) на прямих $x=x_1$ і $x=x_2$ відповідно він повинен йти, щоб дістатися з точки A до точки B якомога швидше?

Загальний час, який мандрівник затратить на весь шлях, через невідомі y_1 і y_2 виражається таким чином:

$$t(y_1, y_2) = \frac{\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}}{V_1} + \frac{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}{V_2} + \frac{\sqrt{(x_3 + x_2)^2 + (y_3 - y_2)^2}}{V_3}$$

(де x_0, x_1, x_2, x_3, y_0, y_3, V_1, V_2, V_3 – задані).

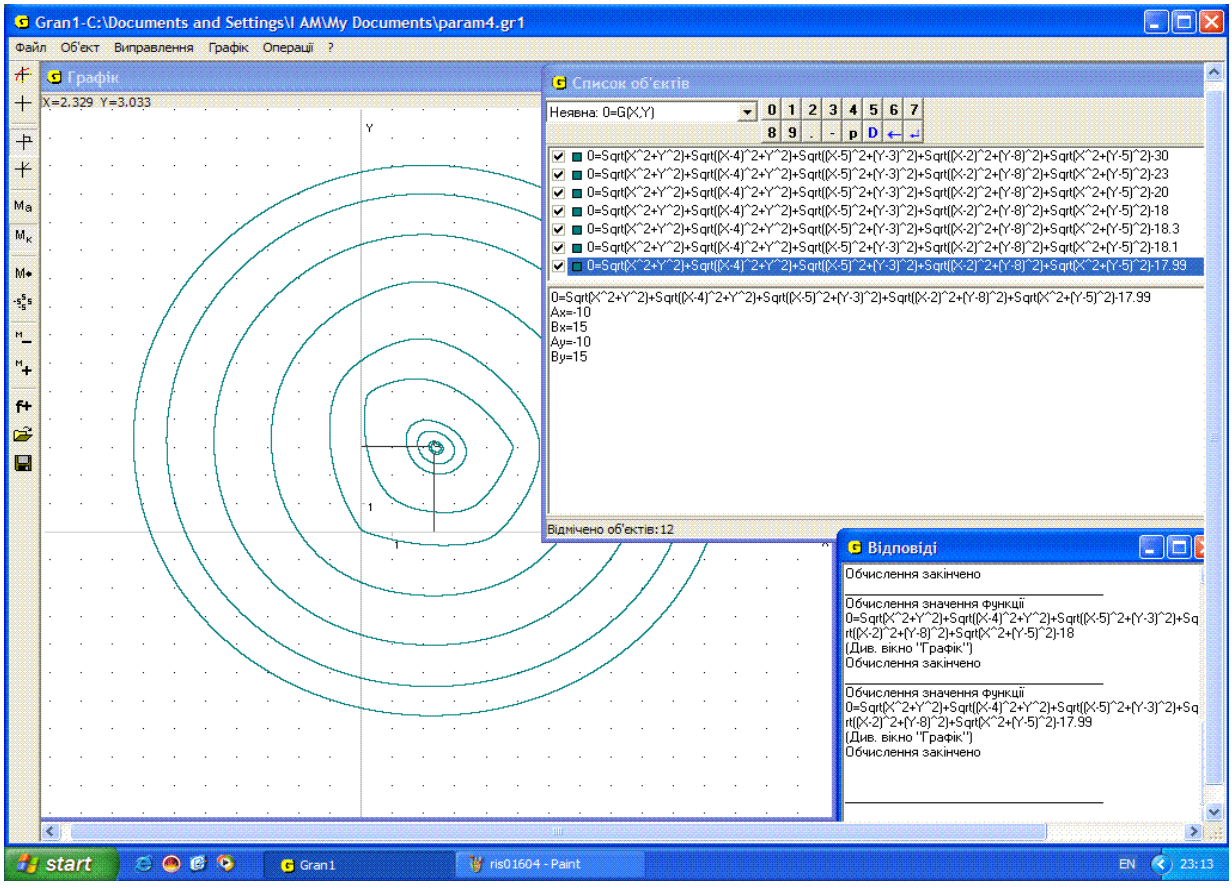

Рис. 2.24. Ілюстрація до задачі Штейнера.

Таким чином, потрібно знайти найменше значення функції $t(y_1, y_2)$ з двома невідомими. Нехай $x_0=0$, $y_0=0$, $x_3=3$, $y_3=3$, $x_1=1$, $x_2=2$, $V_1=0.5$, $V_2=1.5$, $V_3=1$.

Перепозначимо невідомі y_1 і y_2 через x і y відповідно, t – через G і побудуємо графіки залежностей для різних значень *P1*. Надаючи *P1* різних значень одержимо: найменшого значення, рівного 4.8, функція $z=G(x, y)$ досягає в точці $x=0.30$, $y=2.26$ (Рис. 2.25).

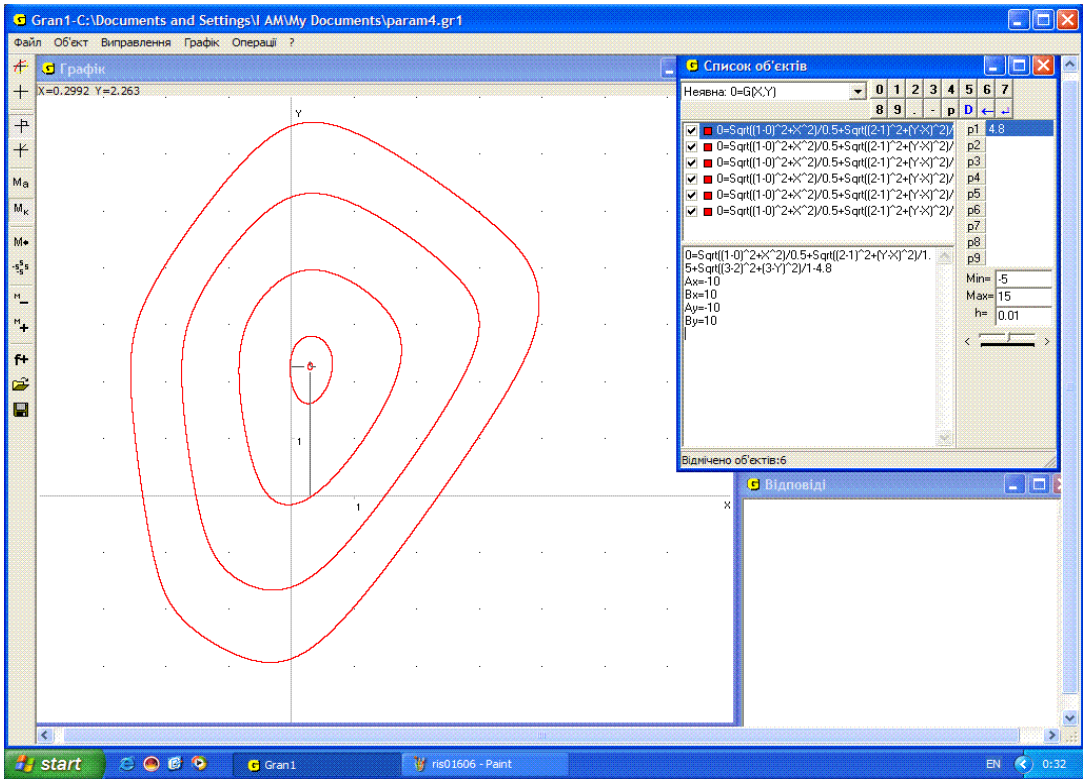

Рис. 2.25. Ілюстрація до Прикладу 9.

Таким чином, щоб дістатися з точки *A(0, 0)* до точки *B(3, 3)* якнайшвидше за даних умов, мандрівник повинен йти спочатку від точки *(0, 0)* до точки *(1, 0.30)*, потім до точки *(2, 2.26)*, і далі до точки *(3, 3)*.

Якщо на кожній із прямих $x=x_i$, ($i=1, 2$), мандрівник може йти не до будь-якої точки, а до однієї із скінченної множини наперед вказаних точок, тоді одержується задача дискретної оптимізації.

Нехай, наприклад, на прямій $x=1$ дозволяється проходити лише через одну з точок *(1, 0)*, *(1, 1.5)*, *(1, 3)*, а на прямій $x=2$ – через одну з точок *(2, 0.5)*, *(2, 2.5)*. Тоді, визначивши з використанням послуги *"Операції/Значення виразу G(x, y)"* значення зазначеної функції $z=G(x, y)$ (при *P1=0*) у точках *(0, 0.5)*, *(1.5, 0.5)*, *(3, 0.5)*, *(0, 2.5)*, *(1.5, 2.5)*, *(3, 2.5)*, одержимо, що найбільш швидко за заданих умов мандрівник перейде з

точки *A(0, 0)* до точки *B(3, 3)*, якщо вибере маршрут *(0, 0) – (1, 0) – (2, 2.5) – (3, 3)*.

Інший спосіб – надаючи параметру *P1* різних значень, знайдемо, що лінія найнижчого рівня (серед допустимих) функції *z=G(x, y)* проходить через точку *(0, 2.5)* (Рис. 2.25) [104].

2.5.5. Опрацювання статистичних даних. Елементи стохастики – важлива складова сучасної шкільної математики, але специфічність цієї дисципліни полягає у опрацюванні досить великих масивів даних, що пов'язано із великою кількістю обчислень, тому логічним є широке використання засобів ІКТ для їх виконання. У педагогічному програмному засобі Gran1 передбачено послуги для роботи з статистичними даними.

Щоб задати нову вибірку, необхідно спочатку у вікні *"Список об'єктів"* встановити тип залежності *"Стат. вибірка"*, а потім скористатися пунктом меню *"Об'єкт/Створити…"* або командою *"Створити"* контекстного меню вікна *"Список об'єктів"*.

На екрані з'явиться допоміжна панель *"Дані для статистичної вибірки"* (Рис. 2.26):

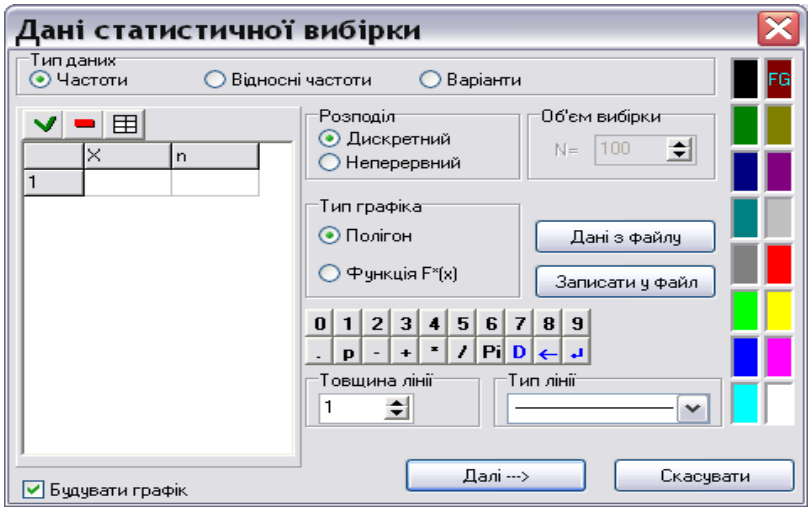

Рис. 2.26. Допоміжна панель для введення даних про вибірку.

Як відомо [93], існує лише два типи розподілів відносних частот (або, що те саме, статистичних ймовірностей) –дискретний і неперервний. Тому спочатку за допомогою радіокнопок *"Модель даних"* необхідно

встановити, який з цих типів розподілів розглядатиметься. Вибір моделі даних залежить від типу розподілу статистичних ймовірностей на множині значень випадкової величини, яку досліджують. Так, наприклад, якщо досліджувана випадкова величина – це кількість розбитих яєць у лотку, зрозуміло, що ця величина може набувати скінченну кількість значень від 0 до загальної кількості яєць у лотку, а якщо досліджувана величина – добовий приріст ваги тварини, то це неперервна величина.

Після встановлення моделі даних за допомогою радіокнопок *"Тип даних"* встановлюють, яким чином будуть введені дані вибірки. Це можуть бути:

- частоти;
- відносні частоти;
- варіанти.

У випадку дискретного розподілу статистичних ймовірностей (відносних частот) необхідно ввести:

- у першому випадку значення варіант *(X)* та відповідних цим варіантам частот *(n)*;
- у другому випадку значення варіант *(X)* і відповідних їм відносних частот *(w)*;
- у третьому випадку значення варіант.

У випадку неперервного розподілу статистичних ймовірностей (відносних частот) необхідно ввести:

- у першому випадку значення середин відрізків *(X)* та кількість варіант *(n)*, що потрапили у ці відрізки, при цьому відрізки повинні мати однакову ширину;
- у другому випадку значення середин відрізків *(X)* та відповідних їм статистичних ймовірностей (відносних частот) *(w)*, при цьому відрізки повинні мати однакову ширину;
- у третьому випадку значення варіант.

Вводити дані можна як за допомогою клавіатури, так і за допомогою числової панелі у допоміжному вікні.

Кнопка  призначена для вставляння в список даних порожнього рядка (відповідна клавіатурна комбінація – Ctrl-Ins). Кнопка  призначена для вилучення рядка із списку даних (відповідна клавіатурна комбінація Ctrl-Del). Кнопка  призначена для очищення списку даних.

Для вибірки, заданої варіантами і відповідними їм відносними частотами, необхідно за допомогою поля редагування *"Об'єм вибірки"* встановити кількість варіант у вибірці. За програмою також перевіряється рівність суми відносних частот одиниці, а при невиконанні цієї умови виводиться відповідне повідомлення.

За допомогою кнопки *"Дані з файлу"* можна завантажити дані у список даних з текстового файлу (дані у файлі повинні бути розділені комами, пропусками або переходами на новий рядок).

Зауваження 1. Для неперервного розподілу відносних частот та типу даних "Частоти" або "Відносні частоти" за програмою перевіряється рівновіддаленість середин відрізків. Якщо рівновіддаленість середин відрізків порушується, вважається, що просто введено певним чином згруповані варіанти і далі робота з даними буде відбуватися відповідним чином.

Зауваження 2. При зміні типу задання вибірки після введення даних всі введені дані будуть вилучені.

Для встановлення типу графіка призначені радіокнопки *"Тип графіка"*. При дискретному розподілі статистичних ймовірностей можливі такі типи графіка:

- многокутник розподілу статистичних ймовірностей (полігон відносних частот);
- функція $F^*(x)$ (функція розподілу статистичних ймовірностей (відносних частот)).

Для неперервного розподілу:

- гістограма (графік щільності розподілу статистичних ймовірностей (відносних частот));
- функція *F*(x)* (функція розподілу статистичних ймовірностей (відносних частот)).

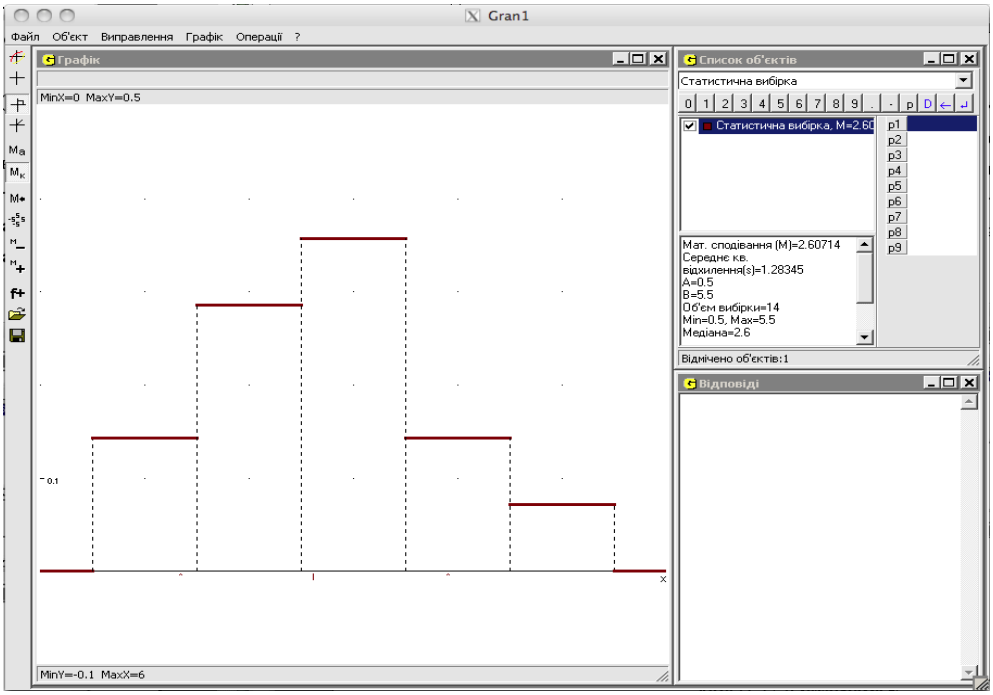

Рис. 2.27. Гістограма.

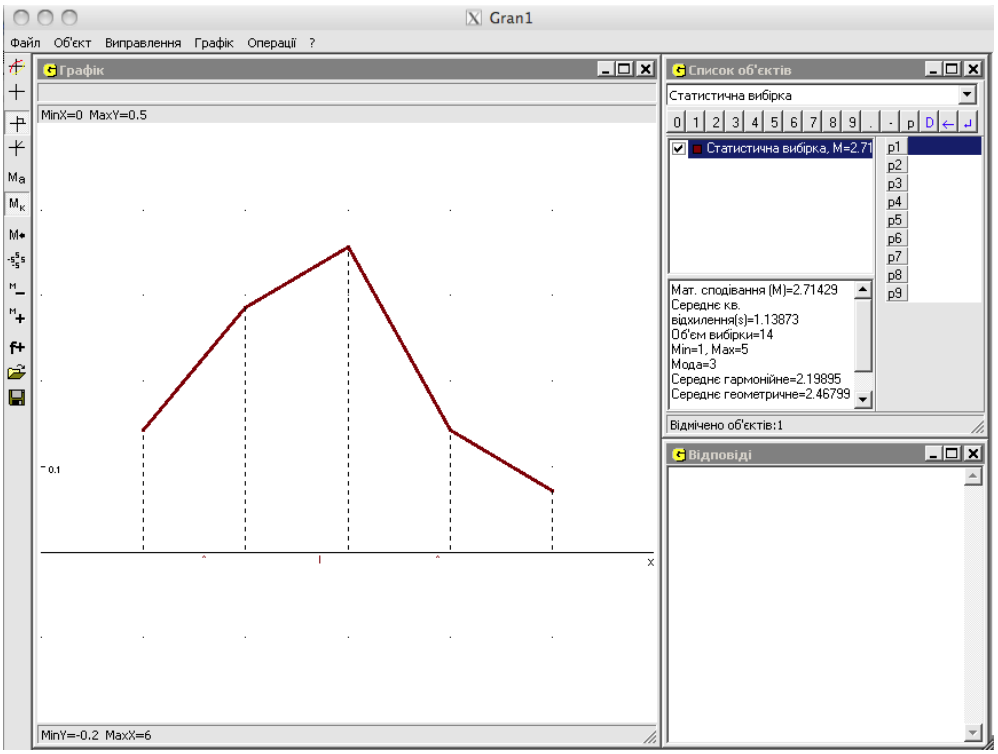

Рис.2.28. Полігон.

Вигляд функції розподілу статистичних ймовірностей (відносних частот) буде залежати від типу розподілу. Для дискретного розподілу графік функції розподілу статистичних ймовірностей буде ступінчастим, а для неперервного – неперервним у вигляді ламаної лінії [92, 100, 103, 104].

На рис. 2.27 зображено гістограму, на рис. 2.28 – полігон, на рис. 2.29 – функцію дискретного розподілу статистичних ймовірностей, на рис. 2.30 – функцію неперервного розподілу статистичних ймовірностей.

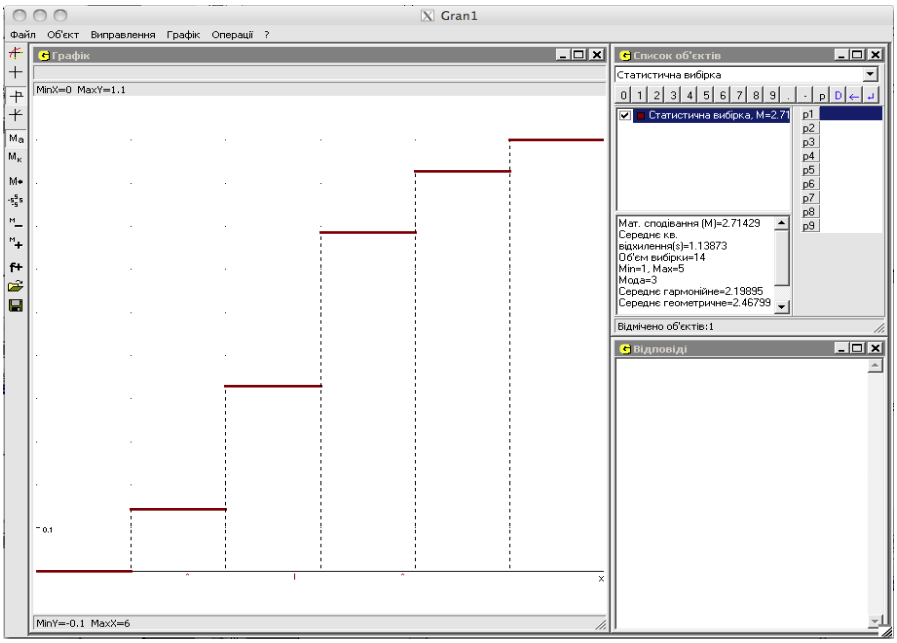

Рис. 2.29. Функція дискретного розподілу статистичних ймовірностей.

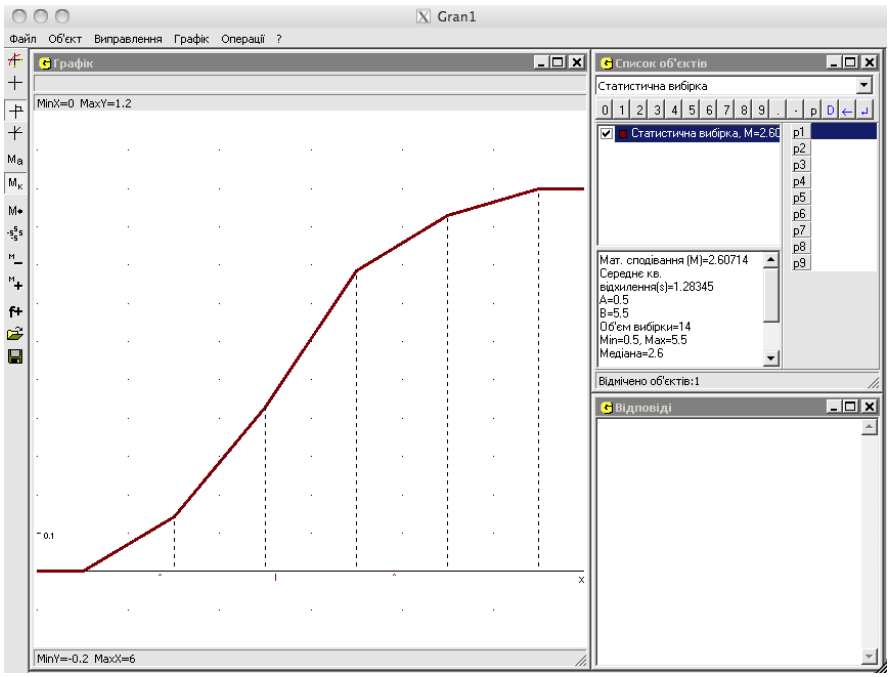

Рис. 2.30. Функція неперервного розподілу статистичних ймовірностей.

Також для графіка можна вибрати товщину лінії та її колір. Для будь-якого виду графіка, що стосується статистичної вибірки, на осі *Ox* відповідними позначками зображується середнє арифметичне (|) та середнє квадратичне відхилення (^).

Після введення всіх даних необхідно натиснути кнопку *“Далі --->”*.

Для неперервного розподілу статистичних ймовірностей, типу даних “Частоти”, у випадку, якщо значення варіант не є рівновіддаленими, після натиснення кнопки *“Далі --->”*, викликається додаткова допоміжна панель (Рис. 2.31):

Спочатку треба ввести межі відрізка задання даних; ці межі, як правило, визначають, виходячи з умови задачі, але слід пам’ятати, що ліва межа відрізка задання даних (*A*) повинна бути не більша за значення мінімальної з варіант, а права межа (*B*) – не менша за значення найбільшої з варіант. За допомогою кнопки *“Відрізок за вибіркою”* встановлюється ліва межа відрізка, рівна мінімальній варіанті, а права межа – максимальній варіанті.

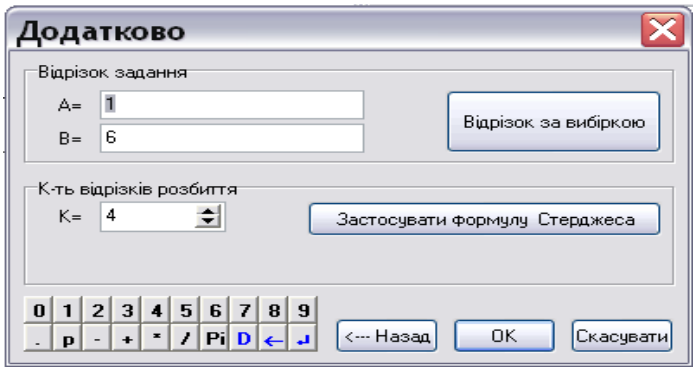

Рис. 2.31. Допоміжні параметри статистичної вибірки.

Далі потрібно встановити кількість підвідрізків (відрізків поділу), на які буде поділено відрізок задання даних для підрахунку кількості варіант, що потрапили в кожний відрізок поділу. Як правило кількість відрізків поділу залежить від об’єму вибірки. Для визначення кількості відрізків поділу можна скористатися формулою Стерджеса (*k=1+3.322*LG(N)*, *N* – об’єм вибірки). Для цього слід натиснути кнопку *“Застосувати формулу Стерджеса”*.

Після натиснення кнопки *“ОК”* закривається вікно *“Додатково”* і введена вибірка записується до списку об’єктів. Натиснувши кнопку *“←Назад”,* можна повернутися до попереднього вікна і змінити вибірку, а при натисненні кнопки *“Скасувати”* закривається вікно без збереження введених даних.

Для редагування вибірки потрібно виділити цю вибірку у вікні *“Список об’єктів”* і скористатися командою меню *“Об’єкт/Змінити…”*. Перед редагуванням даних вибірки можна вибрати бажаний тип цих даних, але слід пам’ятати, що при зміні типу даних після редагування результати редагування будуть скасовані.

Такі характеристики розподілу статистичних ймовірностей, як центр розподілу (середнє арифметичне, мода медіана), виправлена дисперсія та виправлене середньоквадратичне відхилення та деякі інші характеристики визначаються автоматично після створення або редагування вибірки і відображуються в нижній частині вікна *“Список об’єктів”*.

Задачі типу “Визначити кількість лотків з не більш ніж трьома розбитими яйцями у партії з 10000 лотків, на основі аналізу вибірки з 100 лотків” легко розв’язувати за допомогою частотної таблиці (*“Операції/Статистика/Частотна таблиця”*), в якій подані частоти та накопичені частоти вибірки [92, 93, 97, 98, 100, 103, 109, 110].

Перевірка статистичних гіпотез

В програмі Gran1 передбачена можливість перевіряти гіпотези:

Про статистичну однаковість двох вибірок,

Про те, що вказана функція є коректним наближенням функції щільності розподілу статистичних ймовірностей для даної вибірки.

Перевірка цих гіпотез здійснюється за критерієм Пірсона.

Зауваження 1. Вибірки повинні мати однакову кількість різних варіант (для дискретних розподілів статистичних ймовірностей) або однакову кількість відрізків поділу (для неперервних розподілів статистичних ймовірностей).

Зауваження 2. Гіпотетична щільність розподілу ймовірностей, повинна задовольняти властивості щільності розподілу ймовірностей, зокрема інтеграл від цієї функції на відрізку визначення вибірки повинен дорівнювати одиниці.

Зауваження 3. Гіпотезу про те, що вказана функція є коректним наближенням щільності розподілу ймовірностей можна перевіряти тільки для вибірок з неперервним розподілом статистичних ймовірностей.

Критерій Пірсона полягає у визначенні деякого числа χ^2, за яким характеризується розходження між гіпотетичними та статистичними розподілами ймовірностей на множині значень дискретної випадкової величини або між щільністю розподілу статистичних ймовірностей та гіпотетичною щільністю розподілу ймовірностей, і порівнянні його з деяким числом $\chi^2_{крит.}$ $\chi^2_{крит}$ визначають за рівнем значущості (α) та кількістю ступенів вільності. Рівень значущості – це ймовірність того, що $\chi^2 < \chi^2_{крит.}$ [110]. Кількість ступенів вільності у більшості випадків можна визначити за формулою *s=n-1*, де *n* – кількість відрізків поділу для неперервного розподілу статистичних ймовірностей чи кількість різних варіант для дискретного розподілу статистичних ймовірностей.

Якщо $\chi^2 > \chi^2_{крит}$, тоді вважається що гіпотеза не узгоджується з отриманими статистичними (експериментальними) даними.

Якщо $\chi^2 < \chi^2_{крит}$ – тоді вважають, що гіпотеза узгоджується з експериментальними даними.

Щоб скористатися критерієм Пірсона для перевірки гіпотези, необхідно відмітити дві вибірки або вибірку і гіпотетичну функцію щільності розподілу ймовірностей і звернутися до послуги *“Операції/Статистика/Критерій Пірсона…”*. У вікні *“Критерій Пірсона”*, що з’явиться на екрані, потрібно вказати рівень значущості та кількість ступенів вільності.

Результат перевірки гіпотези виводиться у вікні *“Відповідь”*.

Досить часто зустрічаються задачі, в яких необхідно перевірити гіпотезу про те, що на множині значень досліджуваної випадкової величини ймовірності розподілені за нормальним законом. В цьому випадку щільність $f_n^*(x)$ розподілу статистичних ймовірностей необхідно порівнювати з функцією щільності нормального розподілу ймовірностей [110]. Вираз цієї функції досить громіздкий, тому в програмі є послуга *"Операції/Статистика/Функція щільності нормального розподілу за вибіркою"*, при зверненні до якої додається до списку об'єктів функція щільності нормального розподілу ймовірностей з параметрами *M* та *s*, визначеними за вибіркою. Перед тим, як скористатися цією послугою, потрібно встановити покажчик об'єкта на позначення вибірки.

2.6. Моделювання і створення математичних ППЗ

Розглянуті вище підходи до створення математичних ППЗ, вимоги до них, критерії якості ППЗ та інформаційні моделі, покладені в основу функціонування ППЗ Gran1, були взяті за основу для формування у студентів компетентностей щодо створення математичних ППЗ.

2.6.1. Особливості об'єктно-орієнтованого програмування засобами мови Object Pascal. Перш ніж розпочати розгляд основ створення ППЗ, необхідно розглянути особливості середовища програмування, в межах якого будуть створюватися майбутні ППЗ. Зокрема зупинимося на середовищі Lazarus, що базується на об'єктно-орієнтованій мові Object Pascal і перш за все визначимо деякі поняття.

Клас – це визначений користувачем тип даних, якому відповідає його стан (його подання: поля чи властивості), ряд операцій (його поведінка: методи) та подій, з якими пов'язуються певні реагування.

Об'єкт – це екземпляр класу чи змінна типу даних, що відноситься до цього класу.

Атрибут класу – це поля і методи, за допомогою яких характеризують клас.

Атрибути об’єкту – це значення полів, що характеризують об’єкт в його класі. Атрибути об’єкту можуть бути постійні і змінні в процесі роботи з об’єктом.

Будь-яку об’єктно-орієнтовану мову програмування характеризують три основні поняття.

Інкапсуляція (encapsulation) – це механізм об’єднання даних та коду, призначеного для маніпулювання цими даними, а також захисту того і іншого від зовнішнього втручання або неправильного використання.

Концепція інкапсуляції часто позначається „Чорним ящиком”. Немає необхідності турбуватися про його вміст, необхідно лише знати, як працювати з цим „ящиком”, не з'ясовуючи внутрішню структуру. Розділ „Як ним користуватися” називається інтерфейсом класу.

Наприклад, за мірою розвитку програмного продукту розробники можуть прийняти рішення змінити внутрішній устрій певних об’єктів для економії пам’яті чи покращення продуктивності. При цьому перевагою використання інкапсуляції є можливість внесення змін в опис об’єкта, не змінюючи інтерфейсу, без впливу на інші об’єкти.

Наслідування (inheritance) – можливість будувати на основі вже створеного класу класи-нащадки, в яких наслідуються властивості (як поля, так і методи) свого „батька”, а крім того можуть міститися нові дані або методи. Набір класів, пов’язаних наслідуванням, називається ієрархією класів.

Поліморфізм (polymorphism) (від грецького polymorphos) – це властивість, за якою те саме ім'я може використовуватися для виконання схожих, але технічно різних завдань, які відносяться до різних класів, пов’язаних між собою.

Ефект поліморфізму об’єктів полягає в тому, що одне і те саме повідомлення, будучи відісланим до різних об’єктів, може призвести до виконання різних дій (виклику різних методів) в залежності від того, якого

конкретного об’єкта на етапі виконання програми стосується це повідомлення.

При поліморфізмі деякі частини (методи) батьківського класу замінюються новими, за якими реалізуються специфічні дії для даного нащадка. Виконання кожної конкретної дії буде визначатися за типом даних.

З поняттям поліморфізм тісно пов’язане поняття „Пізнього зв’язування”.

Важливо, щоб програма, яка створюється, була надійною. Надійність програм полягає в тому, що в них повинні опрацьовуватися помилки, що виникають, у спосіб, відповідний ситуації, а потім, якщо це можливо, забезпечуватися повернення до виконання коду програми. В протилежному випадку необхідно коректно припинити роботу. У мові Object Pascal стан помилок програми відображається у виключеннях. Виключення – це об’єкти, що містять дані про тип помилки, яка виникла, і місце її виникнення. Опрацювання виключних ситуацій може полягати у виконанні коду завершення, або в корекції стану, через який виникло виключення. Виконання коду завершення є простішим випадком і полягає в гарантованому виконанні певного коду. Як правило, цей різновид реакції на виключення, що виникло, використовується для гарантії того, що за програмою будуть звільнені ті ресурси, які були розподілені згідно з нею, при цьому не звертається увага на стан помилки. У програмі це може виглядати так:

```
 {розподіл ресурсу}
try
 {операції, при виконанні яких можуть виникнути виключні
 ситуації}
finally
 {звільнення ресурсу}
End;
```

Особливо гостро питання надійності постають у програмах, пов’язаних з опрацюванням математичних об’єктів, таких як, наприклад, функції, оскільки існує проблема області визначення, наявність розривів тощо, що призводить до помилок у обчисленнях з плаваючою комою. В якісній програмі повинні опрацьовуватися такі помилки за рахунок надання відповідній змінній певного значення або, якщо це необхідно, повідомлення про виникнення помилки користувачеві. У програмі Gran1 при виникненні помилки обчислення відповіді надається певне значення, яке далі можна аналізувати при побудові графіка, обчисленні інтеграла тощо. Опрацювання такої ситуації здійснюється за допомогою наступної синтаксичної конструкції:

```
try
 {оператори, при виконанні яких може виникнути помилка
(виключення)}
except
 {оператори опрацювання виключення}
End;
```

Оператори в розділі except виконуються тільки тоді, коли виключення виникає під час виконання операторів у розділі try (або у довільній процедурі, що викликана з розділу try).

При використанні виключень можна за короткий проміжок часу написати “правильні” оператори алгоритму, а потім дописати опрацювання виключних ситуацій, якщо це потрібно.

Зважаючи на те, що програмування в Lazarus базується на модульному принципі, тобто групи споріднених класів, які мають розроблятися і змінюватися разом, а також дані, що при цьому використовуються, описуються в окремих модулях. В результаті цього головна програма залишається простою і зрозумілою. Крім того кожен модуль можна компілювати окремо від інших.

Кожен модуль має жорстку структуру, яка автоматично генерується в середовищі програмування при його створенні. Модуль складається з

чотирьох частин, будь яка з яких може бути порожньою: інтерфейсної частини, частини реалізації, частини ініціалізації і частини завершення (останні дві частини є необов'язковими). Спочатку вказується заголовок модуля – ключове слово unit та ім'я модуля (ідентифікатор), яке має співпадати з назвою файлу, в якому зберігається модуль.

```
unit Unit1;
interface        //Інтерфейсна частина
  uses
implementation   //Частина реалізації
  uses
initialization   //Частина ініціалізації
finalization     //Частина завершення
end.
```

Інтерфейсна частина завжди є першою і починається з ключового слова interface. В цій частині містяться оголошення всіх глобальних елементів модуля (типів, констант, змінних і підпрограм), до яких мають бути доступними звернення з основної програми та з інших модулів (до яких буде під'єднано даний). *Частина реалізації* починається з ключового слова implemention і містить коди підпрограм, які були оголошені в інтерфейсній частині. Також в ній можуть бути оголошені локальні для модуля елементи – додаткові типи, константи, змінні і підпрограми. Елементи, описані в частині реалізації, доступні лише в даному модулі.

Такий поділ модуля на частини дозволяє створювати і розповсюджувати модулі у відкомпільованому вигляді, додаючи до них лише опис інтерфейсної частини, щоб програмісти могли використовувати цей модуль, не маючи його вихідних кодів. При цьому внести зміни в такий модуль неможливо. Такий підхід дозволяє, по-перше повторно використовувати раніше написані для інших програм модулі, по-друге, розмежовує доступ до модуля його автора і програмістів, які його використовують, по-третє, дозволяє поділити програму на набір логічно незалежних модулів.

Особливості мови Object Pascal тут розглянуті досить коротко, оскільки питання створення ППЗ будуть вивчатися студентами після засвоєння основ об'єктно-орієнтованого програмування.

2.6.2. Gran1. Базовий клас TFunc та його нащадок TRng. Розглянемо приклади опису класів, їх дерево наслідування, використання поліморфізму на прикладі класів програми Gran1. На вершині ієрархії описано клас TFunc, призначений для зберігання даних про вираз функції. Наведемо фрагмент опису цього класу. Деякі несуттєві для розгляду у даному курсі поля та методи в подальших описах будуть опущені

```
type
  TFunc=class(TObject)
    private
    { опис полів }
     ...
    public
    constructor Create(SF: String);
    function Deriv(Arg: Extended): Extended;
    function PointX(Arg:extended): Extended; virtual;
    function PointY(Arg:extended): Extended; virtual;
    function GetTx: String;
    function Simpson(AI, BI: Extended; IntegralType: Byte):
Extended;
    procedure ChangeFunc(SF: String; var Chng:
Boolean);virtual;
    function Eval(ArgX, ArgY: Extended): Extended;
    ...
  End;
```

Батьківським класом для `TFunc` є базовий клас мови Object Pascal `TObject`. Навіть якщо при описі нового класу не вказати батьківський клас, то за замовчуванням ним буде клас `TObject`.

Клас `Tfunc` містить конструктор `Create(SF: String)`. Конструктор – це специфічний метод, призначений для створення і ініціалізації об'єктів класу. Конструктор призначений для виділення пам'яті під об'єкт і

ініціалізації його полів. Для створення об'єкту викликають його конструктор певним чином. У класі `TFunc` конструктор `Create` має єдиний параметр – вираз функції. Наведемо приклад створення об'єкту типу `TFunc`, що міститиме вираз `'x'`:

```
Var f: TFunc;
…
f:=TFunc.Create('x');
```

В описі конструктора потрібно спочатку викликати конструктор батьківського класу, а потім виконати додаткові дії, пов'язані з ініціалізацією полів:

```
constructor TFunc.Create(SF: String);
...
Begin
  Inherited Create;
  ...
End;
```

При зверненні до функції

```
function Eval(ArgX, ArgY: Extended): Extended;
```

повертається значення виразу для вказаних значень аргументів. Оскільки в програмі Gran1 вирази бувають як з однією, так і з двома змінними, при зверненні до даної функції вказуються значення двох аргументів. Якщо вираз містить тільки одну змінну, значення другого аргументу ігнорується. Якщо у процесі обчислення значення виразу виникла помилка (ділення на нуль, добування квадратного кореня із від'ємного числа тощо), за функцією `Eval()` повернеться значення константи `AllErr=1E+100` за допомогою опрацювання виключень. Це дозволяє аналізувати в подальшому такі помилки.

При зверненні до функції

```
function Deriv(Arg: Extended): Extended;
```

повертається значення похідної для вказаного аргументу. Похідні від виразів, що містять дві змінні, у програмі Gran1 не розглядаються, тому аргумент в цій функції тільки один.

Оскільки на основі виразу будуть визначатися різноманітні функції, для яких потрібно надалі будувати графік, опис класу `TFunc` містить функції

```
function PointX(Arg:extended): Extended; virtual;
function PointY(Arg:extended): Extended; virtual;
```

при зверненні до яких повертаються відповідно значення X та Y координати точки графіка у декартовій системі координат.

Дані функції визначені віртуальними, тому що зміст їх буде змінюватися у деяких нащадках, наприклад для залежності, заданої параметрично, або у полярних координатах (використання поліморфізму).

Функція

```
function Simpson(AI, BI: Extended; IntegralType: Byte):
Extended;
```

призначена для обчислення визначеного інтегралу за методом Сімпсона для даної функціональної залежності.

Процедура

```
procedure ChangeFunc(SF: String; var Chng: Boolean); virtual;
```

призначена для зміни значення виразу; `SF: String` – це новий вираз, у параметрі `var Chng: Boolean` повертається значення `True`, якщо зміна відбулася, і `False` в протилежному випадку (зміна може не відбутися, якщо вираз `SF` є некоректним). Ця процедура також є віртуальною, тому що буде змінюватися у нащадках.

Об’єкти, які опрацьовуються за програмою Gran1, наприклад такі, як явно задана функція Y=Y(X), параметрично задана функція, функція, що описує залежність між полярними координатами, визначаються не тільки виразом, але і відрізком задання. В свою чергу відрізок задання може задаватися виразами, через які визначаються значення лівого та правого кінців відрізка, тому потрібен клас для визначення кінця відрізка задання. Очевидно, що даний клас буде наслідником класу `TFunc`. Він описується наступним чином:

```
TRng=class(TFunc)
```

```
        private
        FValue: Extended;
        function GetValue: Extended;
        public
        constructor Create(SA: String);
        procedure ChangeFunc(SF: String; var Chng: Boolean);
override;
        published
        property  Value: Extended read GetValue write FValue;
      End;
```

У порівнянні з батьківським класом тут з'явилась властивість `Value`, описана наступним чином:

```
property  Value: Extended read GetValue write FValue;
```

Значення цієї властивості визначається шляхом обчислення значення виразу, через який задається кінець відрізка.

2.6.3. Gran1. Клас для відображення системи координат TMasshImage. Для відображення графіків об'єктів програми Gran1 потрібен певний візуальний елемент, пов'язаний з декартовою системою координат. Для створення такого компоненту в якості батьківського був вибраний стандартний компонент `TImage`. Розглянемо опис цього компоненту та деяких допоміжних типів даних.

```
type
  MmType=record
    MinArg, MaxArg, MinFunc, MaxFunc: Extended
  End;
```

Цей тип даних призначений для зберігання мінімального та максимального значень на кожній із осей координат.

```
type
  CoordType=(ctDec,ctPolar);
```

За цим типом даних визначається, яка система координат зображується на екрані — декартова чи полярна.

```
      TMasshImage = class(TImage)
      private
        { Private declarations }
```

```
        FOsiPoints: Boolean;
        FGraphPoints: Integer;
        FMasshAuto: Boolean;
        FMasshUpdate: TNotifyEvent;
        FCoord: CoordType;
        FXBasic:Extended;
        FYBasic:Extended;
        FIsMarkBuild:Boolean;
        FgrTextColor,FgrBackColor,FgrMouseColor,
        FgrMarkColor,FgrAxeColor,FgrAddColor: TColor;
        FgrXName,FgrYName: String;
        FgrLabelFont,FgrAxeFont: TFont;
        FgrAdd2Color:TColor;
      public
        { Public declarations }
        GraphRang: MmType;
        OldGraphRang: MmType;
        NumOfPoints: Integer;
        constructor Create(AOwner: TComponent); override;
        destructor Destroy; override;
        function XWin(X: Extended): Integer;
        function YWin(Y: Extended): Integer;
        function WinX(X: Integer): Extended;
        function WinY(Y: Integer): Extended;
        function WinR(X,Y: Integer): Extended;
        function WinF(X,Y: Integer): Extended;
        procedure ChangeGraphOps;
        procedure SetGraphRang(G: MmType);
        procedure UpToDate;
        procedure SetCoord(C: CoordType);
        procedure SetMasshAuto(M: Boolean);
      published
        property OsiPoints: Boolean read FOsiPoints write
FOsiPoints;
        property GraphPoints: Integer read FGraphPoints write
FGraphPoints;
```

```
        property MasshAuto: Boolean read FMasshAuto write
SetMasshAuto;
        property OnMasshUpdate: TNotifyEvent read FMasshUpdate
write FMasshUpdate;
        property Coord: CoordType read FCoord write SetCoord
default ctDec;
        property XBasic: Extended read FXBasic write FXBasic;
        property YBasic: Extended read FYBasic write FYBasic;
        property IsMarkBuild: Boolean read FIsMarkBuild write
FISMarkBuild;
        property grTextColor: TColor read FgrTextColor write
FgrTextColor;
        property grBackColor: TColor read FgrBackColor write
FgrBackColor;
        property grMarkColor: TColor read FgrMarkColor write
FgrMarkColor;
        property grMouseColor: TColor read FgrMouseColor write
FgrMouseColor;
        property grAxeColor: TColor read FgrAxeColor write
FgrAxeColor;
        property grAddColor: TColor read FgrAddColor write
FgrAddColor;
        property grXName: String read FgrXName write FgrXName;
        property grYName: String read FgrYName write FgrYName;
        property grLabelFont: TFont read FgrLabelFont write
FgrLabelFont;
        property grAxeFont: TFont read FgrAxeFont write
FgrAxeFont;
        property grAdd2Color: TColor read FgrAdd2Color write
FgrAdd2Color;
      End;
```

У цьому класі описано значну кількість властивостей, через які визначається зовнішній вигляд системи координат і графіків, наприклад за властивістю

```
property Coord: CoordType read FCoord write SetCoord default
ctDec;
```

визначається, яку систему координат показувати при побудові графіків – декартову чи полярну, за замовчуванням визначена декартова система координат.

За властивістю

```
property grAxeColor: TColor read FgrAxeColor write
FgrAxeColor;
```

визначається колір осей координат.

Призначення інших властивостей досить легко визначити за їх назвами. Зауважимо, що такий опис властивостей дозволяє встановити їх початкові значення у візуальному редакторі Lazarus.

Початок системи координат компоненти `TImage` знаходиться у лівому верхньому кутку області зображення, а координатами для цієї області зображення є цілі невід'ємні числа. В свою чергу в об'єктах програми Gran1 використовуються дійсні значення координат, а центр координат може бути де завгодно, як всередині області зображення, так і за її межами, тому потрібно передбачити методи для перетворення координат компоненти `TImage` (екранних координат) в координати об'єктів Gran1 (математичні координати) і навпаки.

При зверненні до функцій

```
function XWin(X: Extended): Integer;
function YWin(Y: Extended): Integer;
```

повертаються відповідно екранні X та Y для математичних X та Y координат.

Для прикладу наведемо код методу `XWin`:

```
function TMasshImage.XWin(X: Extended): Integer;
  var t:extended;
Begin
  with GraphRang do
  t:=(X-MinArg)*Width/(MaxArg-MinArg);
```

```
    if abs(t)<30000 then Xwin:=Round(t) else
Xwin:=Sign(t)*30000
  End;
```

За допомогою функцій

```
    function WinX(X: Integer): Extended;
    function WinY(Y: Integer): Extended;
```

виконуються зворотні перетворення. Їх можна, наприклад, використати, щоб відображати математичні координати при русі курсору мишки над компонентою, створеною на основі класу `TMasshImage`, оскільки координати курсору мишки в стандартних візуальних компонентах є екранними.

Для прикладу наведемо код методу `WinX`:

```
function TMasshImage.WinX(X: Integer): Extended;
Begin
  with GraphRang do
  WinX:=(maxarg-minarg)*x/Width+minarg;
End;
```

За допомогою програми Gran1 можна будувати графіки функцій у полярних координатах, але оскільки компонента типу `TMasshImage` пов’язана з декартовими координатами, побудова графіків таких функцій відбувається шляхом перетворення полярних координат у декартові. В свою чергу існує потреба відображати полярні координати при переміщенні курсору мишки над компонентою, створеною на основі класу `TMasshImage`. В цьому випадку можна скористатися функціями

```
    function WinR(X,Y: Integer): Extended;
    function WinF(X,Y: Integer): Extended;
```

при зверненні до яких за екранними координатами повертаються полярні координати ρ та φ відповідно.

Для прикладу наведемо код методу `WinR`:

```
function TMasshImage.WinR(X,Y: Integer): Extended;
  var xrf,yrf: Extended;
Begin
  with GraphRang do
```

```
    Begin
      xrf:=(maxarg-minarg)*X/Width+minarg;
      yrf:=(maxfunc-minfunc)*(Height-Y)/Height+minfunc;
    End;
    WinR:=Sqrt(Sqr(xrf)+Sqr(yrf));
End;
```

Для побудови графіків потрібно знати мінімальне та максимальне значення на кожній з координатних осей (масштаб). Ці значення зберігаються в полі `GraphRang` типу `MmType`. Крім того, ці значення можуть бути визначені користувачем, або визначатися автоматично за списком об'єктів програми Gran1, відібраних користувачем для побудови їх графіків. Вид масштабування зберігається у властивості

```
FMasshAuto: Boolean;
```

якщо її значення `True`, масштабування автоматичне, в протилежному випадку масштаб визначається користувачем.

Метод

```
procedure SetGraphRang(G: MmType);
```

призначений для встановлення нового масштабу, якщо масштаб визначається користувачем.

Метод

```
procedure UpToDate;
```

призначений для побудови графіків. Оскільки компонент типу `TMasshImage` "не знає", графіки яких об'єктів програми Gran1 треба будувати, за цим методом генерується подія

```
FMasshUpdate: TNotifyEvent;
```

яка буде опрацьовуватися власником компоненти типу `TMasshImage`. Наведемо код даного методу

```
procedure TMasshImage.UpToDate;
  var Save_Cursor:TCursor;
Begin
  Save_Cursor:=Screen.Cursor;
  Screen.Cursor:=crHourglass;
  try
```

```
  try
    OnMasshUpdate(Self);
  finally
    Screen.Cursor:=Save_Cursor;
  End;
  except
  End;
End;
```

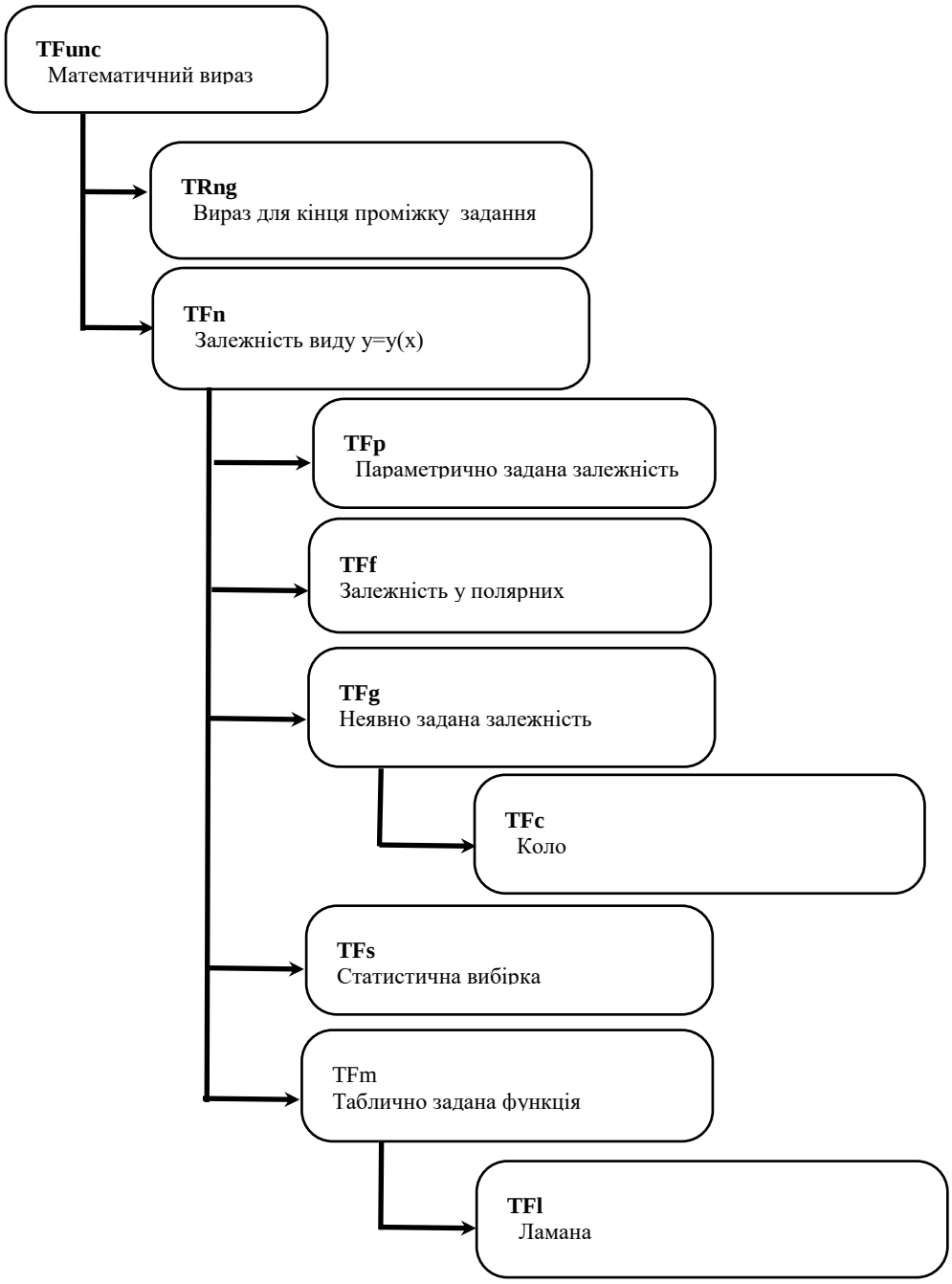

Рис. 2.32. Дерево класів програми Gran1.

У даному методі використано опрацювання виключних ситуацій, наприклад у процесі побудови графіків курсор мишки набуває форми пісочного годинника і гарантується, що після закінчення побудови він набуде тієї форми, якої був до початку побудови.

2.6.4. Gran1. Ієрархія класів. Класи для зберігання даних про математичні об'єкти. Розглянемо тепер, як здійснюється опис і реалізація основних об'єктів програми Gran1, таких як явно задана залежність Y=Y(X), параметрично задана залежність, залежність у полярних координатах, неявно задана залежність тощо. Сукупність цих об'єктів утворюють ієрархію, на вершині якої знаходиться математичний вираз (Рис. 2.32).

Розглянемо клас, де описується залежність `Y=Y(X)` у програмі Gran1.

```
TFn=class(TFunc)
   FArg:Char;
   Mm: MmType;
   A, B: TRng;
   FMark: Boolean;
   FColor: TColor;
   FGraph: Boolean;
   GBuild: Boolean;
   BGraphPoints: Integer;
   BuildByPoints: Boolean;
   WLine: Integer;
   constructor Create(SFy,SA,SB: String; GColor: TColor);
   constructor Load(f: TIniFile; Sec: String);
   procedure Save(f: TIniFile; Sec: String); virtual;
   destructor Destroy; override;
   procedure MmF; virtual;
   function GetType: FType; virtual;
   function GetTypeName: String; virtual;
   function GetName: String; virtual;
   function GetNameP: String; virtual;
   function MarkPossible: Boolean; virtual;
   function IntPossible: Boolean; virtual;
```

```
        procedure GraphBuild(I: TMasshImage); virtual;
        procedure Change(var Chng: Boolean); virtual;
        procedure Info(M: TMemo); virtual;
        procedure Nerav(AA: Extended; Sgn: Byte; I: TMasshImage;
N: TNerDlg);
        procedure IntImage(AI, BI: Extended; IntegralType: Byte;
II: TMasshImage); virtual;
        function GetColor: TColor;
        function LineLong(AI, BI: Extended): Extended;
        procedure LLongImage(AI, BI: Extended; II: TMasshImage);
        function GetHead: String; virtual;
        function Spoverh(ai,bi:extended;oss:byte):extended;
        procedure ScanParams; override;
      End;
```

Цей клас `TFn` є нащадком класу `TFunc`, тому наслідує всі його властивості. Розглянемо тепер, які нові властивості з'явилися у цьому класі.

Поле `FArg: Char` містить позначення аргументу. У даному класі це поле набуває значення 'X'.

Через поля A, B: TRng; визначаються лівий та правий кінці відрізка задання функції. Поле `Mm: MmType;` містить найбільші та найменші значення на осях OX та OY. Значення `Mm` обчислюються кожного разу після зміни виразу функції або зміни відрізка задання за допомогою методу `MmF`, в якому організовується цикл від найменшого до найбільшого значення на відрізку задання, на кожному кроці цього циклу за допомогою методів `PointX()` та `PointY()` із класу батьківського класу `TFunc` обчислюються значення X та Y координат відповідної точки у декартовій системі координат і знаходяться найменше і найбільше із цих значень:

```
procedure TFn.MmF;
...
Begin
...
  t:=A.Value;
```

```
  while t<=B.Value do
    Begin
      x:=PointX(t);
      y:=PointY(t);
      if (x<Mm.MinArg) and (x<AllErr) then Mm.MinArg:=x;
      if (x>Mm.MaxArg) and (x<AllErr) then Mm.MaxArg:=x;
      if (y<Mm.MinFunc) and (y<AllErr) then Mm.MinFunc:=y;
      if (y>Mm.MaxFunc) and (y<AllErr) then Mm.MaxFunc:=y;
      t:=t+Step
    End;
...
End;
```

Оскільки об’єкти програми Gran1 можна записувати у файл та зчитувати з файлу, крім звичайного конструктора

```
constructor Create(SFy,SA,SB: String; GColor: TColor);
```

за допомогою якого створюється новий об’єкт класу TFn за переданими виразами для функції, кінців відрізка задання та кольору графіка, у класі TFn передбачено конструктор

```
constructor Load(f: TIniFile; Sec: String);
```

за допомогою якого створюється новий об’єкт на основі даних із вказаного файлу.

Для зберігання даних про об’єкти програми Gran1 використовуються текстові файли спеціального виду – так звані ini-файли. Ці файли мають наступний вигляд

```
[секція 1]
параметр1.1=значення1.1
параметр1.2=значення1.2
параметр1.3=значення1.3
параметр1.4=значення1.4
...
[секція 2 ]
параметр2.1=значення2.1
параметр2.2=значення2.2
параметр2.3=значення2.3
```

```
параметр2.4=значення2.4
...
```

Дані про кожен об’єкт програми записуються у окрему секцію і відповідно зчитуються з неї:

```
constructor TFn.Load(f: TIniFile; Sec: String);
var SFy, SA, SB: String;
    Color: TColor;
Begin
  SFy:=f.ReadString(Sec,'Func','');
  SA:=f.ReadString(Sec,'A','');
  SB:=f.ReadString(Sec,'B','');
  Color:=TColor(f.ReadInteger(Sec,'Color',0));
  Create(SFy,SA,SB,Color);
  ...
End;
```

Для запису даних про об’єкт використовується метод

```
procedure Save(f: TIniFile; Sec: String); virtual;
```

за допомогою якого записуються у вказану секцію тип об’єкту та його параметри. Метод є віртуальним, тому що для різних класів у файл потрібно записувати різні параметри.

Розглянемо тепер метод

```
procedure GraphBuild(I: TMasshImage); virtual;
```

призначений для побудови графіка функції у вікні візуального об’єкту `I: TMasshImage`, що є параметром цього методу. Метод описано віртуальним, тому що у нащадках він може змінюватися (наприклад він повинен змінитися у нащадкові, де описується неявно задана залежність, оскільки алгоритми побудови графіка для залежностей явно і неявно заданих практично не мають нічого спільного).

Реалізувати цей метод можна наступним чином:

організовується цикл від найменшого до найбільшого значення на відрізку задання, на кожному кроці цього циклу за допомогою методів `PointX()` та `PointY()` із батьківського класу `TFunc` обчислюються значення X та Y

координат відповідної точки у декартовій системі координат і будується лінія від попередньої точки до поточної:

```
...
t:=A.Value;
 while t<=B.Value do
   Begin
      x:=PointX(t);
      y:=PointY(t);
      I.Picture.BitMap.Canvas.MoveTo(Xwin(x1),Ywin(y1));
      I.Picture.BitMap.Canvas.LineTo(Xwin(x2),Ywin(y2));
      t:=t+step;
   End;
...
```

Зрозуміло, що наведений алгоритм є дуже схематичним, оскільки тут, наприклад, не передбачена ситуація виникнення розриву. Реальний алгоритм значно складніший.

У модулі, де описано класи функцій, є тип

```
type
  FType=(Normal, Param, Polar, TwoArg, Polinom, Stat, Lom,
Circ);
```

для визначення типу залежності:

Normal – залежність y=y(x);

Param – параметрично задана залежність;

Polar – залежність, задана у полярних координатах;

TwoArg – неявно задана залежність;

Polinom – таблично задана функція, що апроксимується поліномом;

Stat – статистична вибірка;

Lom – ламана;

Circ – коло.

При зверненні до функції `GetType` повертається тип залежності.

Поле `BGraphPoints: Integer`; містить кількість точок, за якими будується графік залежності.

Розглянемо тепер, як описані класи для інших об'єктів програми Gran1.

Клас для залежностей у полярних координатах описується, як нащадок класу залежностей *Y=Y(X)*.

```
TFf=class(TFn)
   constructor Create(SFy,SA,SB: String; GColor: TColor);
   constructor Load(f: TIniFile; Sec: String);
   function PointX(Arg:extended):extended;override;
   function PointY(Arg:extended):extended;override;
   function IntPossible:Boolean;override;
   function GetType:FType;override;
   function GetTypeName: String; override;
   function GetHead: String; override;
End;
```

Конструктор `Create(SFy,SA,SB: String; GColor: TColor);` має такий самий заголовок, як і конструктор предка, але у ньому слід відобразити той факт, що змінюється позначення аргументу з *'X'* на *'F'*, тому його код виглядатиме так:

```
constructor TFf.Create(SFy, SA, SB: String; Gcolor: TColor);
Begin
  inherited Create(SFy, SA, SB, Gcolor);
  FArg:='F'
End;
```

Такі самі зміни будуть і в конструкторі `Load()`:

```
constructor TFf.Load(f: TIniFile; Sec: String);
Begin
  inherited Load(f,Sec);
  FArg:='F'
End;
```

Для того, щоб у декартових координатах правильно будувати графік залежності, заданої у полярних координатах, потрібно переписати також методи `PointX()` та `PointY()`:

```
function TFf.PointX(Arg:extended):extended;
Begin
```

```
  PointX:=Eval(Arg,0)*Cos(Arg)
End;

function TFf.PointY(Arg:extended):extended;
Begin
  PointY:=Eval(Arg,0)*Sin(Arg)
End;
```

Після такого перевизначення батьківський метод побудови графіка може залишатися незмінним, як незмінним може залишатися і метод `MmF()` обчислення найбільшого і найменшого значення на осях *OX* та *OY*. Це є ще одним прикладом використання поліморфізму.

Клас для параметрично заданих залежностей також описується, як нащадок класу залежностей *Y=Y(X)*.

```
  TFp=class(TFn)
    Fx: TFunc;
    constructor Create(SFy, SFx, SA, SB: String; GColor:
TColor);
    constructor Load(f: TIniFile; Sec: String);
    procedure Save(f: TIniFile; Sec: String); override;
    destructor Destroy; override;
    function PointX(Arg: Extended): Extended; override;
    function PointY(Arg: Extended): Extended; override;
    function GetType: FType; override;
    function GetTypeName: String; override;
    function GetName: String; override;
    function GetNameP: String; override;
    function IntPossible: Boolean; override;
    procedure Change(var Chng:Boolean); override;
    function GetHead: String; override;
    procedure ScanParams; override;
    function ParamExist: Boolean; override;
  End;
```

Оскільки параметрично задана залежність описується вже не одним виразом, як залежність *Y=Y(X)*, а двома *Y=Y(T), X=X(T)*, у класі `TFp`

потрібно додати нове поле `Fx: TFunc`, що міститиме вираз для *X=X(T)*. Відповідні зміни потрібно внести і в конструктори:

```
    constructor TFp.Create(SFy,SFx,SA,SB: String; GColor:
TColor);
    Begin
      inherited Create(SFy,SA,SB,GColor);
      Fx:=TFunc.Create(SFx);
      FArg:='T'
    End;

    constructor TFp.Load(f: TIniFile; Sec: String);
    var SFx: String;
    Begin
      inherited Load(f,Sec);
      SFx:=f.ReadString(Sec,'FuncX','');
      Fx:=TFunc.Create(SFx);
      FArg:='T';
    End;
```

а також у метод, призначений для зберігання об’єкту у файлі:

```
    procedure TFp.Save(f: TIniFile; Sec: String);
    Begin
      inherited Save(f,Sec);
      f.WriteString(Sec,'FuncX',Fx.GetTx);
    End;
```

Потребує змін і деструктор, щоб звільнити пам’ять, виділену під поле Fx:

```
    destructor TFp.Destroy;
    Begin
      Fx.Free;
      inherited Destroy;
    End;
```

Також потрібно внести зміни у метод для обчислення *X*-координати, оскільки *X*-координата є результатом обчислення виразу *X=X(T)*:

```
    function TFp.PointX(Arg:extended):extended;
    Begin
```

```
      PointX:=Fx.Eval(Arg,0)
    End;
```

І нарешті розглянемо клас для неявно заданих залежностей виду *G(X,Y)=0*:

```
      TFg=class(TFn)
        FArgY: Char;
        YA,YB: TRng;
        NameGxy: String[5];
        PrecisionGraph: Integer;
        constructor Create(SFy, SAx, SBx, SAy, SBy: String;
GColor: TColor);
        constructor Load(f: TIniFile; Sec: String);
        procedure Save(f: TIniFile; Sec: String); override;
        destructor Destroy; override;
        function GetType: Ftype; override;
        function GetTypeName: String; override;
        function IntPossible: Boolean; override;
        procedure GraphBuild(I: TMasshImage); override;
        procedure Info(M: TMemo); override;
        procedure MmF(II: TMasshImage); override;
        procedure Change(var Chng: Boolean); override;
        function GetHead: String; override;
        procedure ScanParams; override;
      End;
```

Оскільки вираз неявно заданої залежності має два аргументи, така залежність розглядається вже не на відрізку задання, а на прямокутнику задання. Межі цього прямокутника на осі *OX* наслідуються від батька – класу `TFn`, а на осі *OY* описуються у полях `YA,YB: TRng;` також потрібно додати поле `FArgY: Char`; для зберігання назви другого аргумента (*'Y'*).

Повинні змінитися і конструктори:

```
    constructor TFg.Create(SFy,SAx,SBx,SAy,SBy: String; GColor:
TColor);
    Begin
      inherited Create(SFy,SAx,SBx,GColor);
```

```
  YA:=TRng.Create(SAy);
  YB:=TRng.Create(SBy);
  FArg:='X';
  FArgY:='Y';
  PrecisionGraph:=6;
End;

constructor TFg.Load(f: TIniFile; Sec: String);
var SFy, SA, SB, SYA, SYB: String;
    Color: TColor;
Begin
  inherited Load(f,Sec);
  FArg:='X';
  FArgY:='Y';
  SYA:=f.ReadString(Sec,'YA','');
  SYB:=f.ReadString(Sec,'YB','');
  YA:=TRng.Create(SYA);
  YB:=TRng.Create(SYB);
  PrecisionGraph:=f.ReadInteger(Sec,'PrecisionGraph',6);
End;
```

метод збереження даних у файлі:

```
procedure TFg.Save(f: TIniFile; Sec: String);
Begin
  inherited Save(f,Sec);
  f.WriteString(Sec,'YA',YA.GetTx);
  f.WriteString(Sec,'YB',YB.GetTx);
  f.WriteInteger(Sec,'PrecisionGraph',PrecisionGraph);
End;
```

та деструктор:

```
destructor TFg.Destroy;
Begin
  YA.Free;
  YB.Free;
  inherited Destroy;
End;
```

Як вже зазаначалося, побудова графіка неявно заданої залежності докорінно відрізняється від побудови графіків всіх залежностей, що були розглянуті раніше, тому метод `GraphBuild()` повинен бути повністю переписаний. Алгоритм побудови графіка неявно заданої залежності є неочевидним і досить складним, тому наводити його тут не будемо. Ідея полягає у пошуку на прямокутнику задання точок, де вираз G(X,Y) змінює знак.

Значна кількість операцій за програмою Gran1, наприклад побудова графіків, інтегрування, розв'язування нерівностей може виконуватися над групою об'єктів. Доцільно таку групу подати у вигляді списку, що базується на основі стандартного списку `TList`:

```
type
 TFuncs=class(TList)
   GRI:TMasshImage;
   ...
   constructor Create;
   procedure LineLongs;
   procedure Kasat;
   procedure Neravs;
   procedure LLong;
   procedure SParams;
   procedure SquareLs;
   procedure VSoxyLs(OpsType: Integer);
   procedure Pirson;
   procedure NormFunc;
   procedure SysNerav;
   procedure FgNerav(Z: Char);
   procedure DoFixParam;
   {procedure PodobLam(var Fl: TFl; var CR: Boolean);}
 public
   { Public declarations }
  ...
 End;
```

Поле GRI є посиланням на візуальний об’єкт, в якому будуються графіки. Це посилання необхідне, оскільки при виконанні деяких операцій, наприклад розв’язування нерівностей, здійснюються допоміжні побудови (у випадку розв’язування нерівностей це зображення кореня на осі OX). Також тут описано методи для виконання тих чи інших операцій над об’єктами з цього списку, наприклад метод SysNerav для розв’язування системи нерівностей, відповідної набору об’єктів, що є у списку.

2.6.5. Gran1. Клас для роботи зі списком об’єктів. Інтерфейс програми Gran1 базується на так званому MDI-стандарті, коли в межах головного вікна програми існують кілька вікон для відображення відповідних даних. Так у вікні “Список об’єктів” зображуються створені користувачем об’єкти програми Gran1, причому ці об’єкти користувач може відмічати для виконання групових операцій. Вікно “Графік” призначене для відображення графіків об’єктів, а у вікно “Відповіді” записуються результати обчислень. Кожному вікну відповідає певний клас, що є нащадком класу TForm. Розглянемо спочатку особливості класу, призначеного для реалізації роботи зі списком об’єктів програми Gran1. Для створення такого класу з ім’ям TObjList доцільно скористатися візуальним конструктором і розмістити у вікні візуальні компоненти ObjListBox: TCheckListBox; для подання списку об’єктів, ObjMemo: TMemo; для виведення даних про поточний об’єкт у списку об’єктів та ObjComboBox: TComboBox; для вибору поточного типу об’єктів (у програмі можна створити тільки об’єкт поточного типу). Також потрібно додати у опис посилання на візуальну компоненту, в якій будуть будуватися графіки: GRI: TMasshImage;

```
type
 TObjList = class(TForm)
    ObjComboBox: TComboBox;
    ObjListBox: TCheckListBox;
    ObjMemo: TMemo;
    ...
```

```
        procedure FormCreate(Sender: TObject);
        procedure FormClose(Sender: TObject; var Action:
TCloseAction);
        procedure NewObjPopupClick(Sender: TObject);
        procedure DelObjPopupClick(Sender: TObject);
        procedure ChngObjPopupClick(Sender: TObject);
        procedure ObjListBoxDrawItem(Control: TWinControl; Index:
Integer;
          Rect: TRect; State: TOwnerDrawState);
        procedure FormDestroy(Sender: TObject);
        procedure ObjComboBoxChange(Sender: TObject);
        procedure SelAllPopupClick(Sender: TObject);
        procedure DeSelAllPopupClick(Sender: TObject);
        procedure ObjListBoxClick(Sender: TObject);
        procedure ObjListBoxKeyDown(Sender: TObject; var Key:
Word;
          Shift: TShiftState);
        procedure ObjLPopupMenuPopup(Sender: TObject);
        procedure ObjListBoxClickCheck(Sender: TObject);
        procedure DelLastObjPopUpClick(Sender: TObject);
        procedure DelAllObjPopupClick(Sender: TObject);
        procedure FormActivate(Sender: TObject);
        procedure FormResize(Sender: TObject);
        procedure DelActiveObjPopUpClick(Sender: TObject);
        ...
      private
        { Private declarations }
      ...
      public
        { Public declarations }
        GRI: TMasshImage;
        GlobFType:FType;
        Funcs:TFuncs;
        FuncsG:TFuncs;
        NeedSave: Boolean;
        procedure MakeFuncsList(Funcs:TFuncs;OLSel:Boolean);
```

```
        procedure SetMasshtab;
        procedure ReadFuncs(FileName: String);
        procedure SaveFuncs(FileName: String);
        procedure SetObjMemo;
        procedure MmFAll;
      End;
```

У програмі Gran1 при побудові графіків розглядаються тільки ті об’єкти, у властивостях яких це визначено, а при виконанні різноманітних операцій, таких як інтегрування, розв’язування нерівностей тощо розглядаються ті об’єкти, які відмічено у візуальному списку об’єктів. Таким чином ці дві множини об’єктів можуть не співпадати. Тому у класі `TObjList` визначено два списки об’єктів: список `FuncsG: TFuncs;` міститиме посилання на об’єкти, графіки яких треба будувати, а список `Funcs: TFuncs;` міститиме посилання на об’єкти, які відмічено у візуальному списку об’єктів.

Для створення нового об’єкта викликається метод `NewObjPopupClick(Sender: TObject)`, який, наприклад, можна прив’язати до команди контекстного меню:

```
    procedure TObjList.NewObjPopupClick(Sender: TObject);
      var Fn:TFn;
          C:Boolean;
    Begin
      case GlobFType of
        Normal: Fn:=TFn.Create('X','-
5','5',GlobColors.NextFColor);
        Polar :
Fn:=TFf.Create('F','0','2*Pi',GlobColors.NextFColor);
        Param : Fn:=TFp.Create('T','T','-
5','5',GlobColors.NextFColor);
        TwoArg: Fn:=TFg.Create('X','-5','5','-
5','5',GlobColors.NextFColor);
        Polinom:Fn:=TFm.Create(GlobColors.NextFColor);
        Lom:    Fn:=TFl.Create(GlobColors.NextFColor);
        Stat:   Fn:=TFs.Create(GlobColors.NextFColor);
```

```
    Circ:   Fn:=TFc.Create('X*X+Y*Y-1','-1','1','-
1','1',GlobColors.NextFColor);
      End;
      Fn.Change(C);
      if C then
        Begin
          NeedSave:=True;
          Fn.MmF(GRI);
          ObjListBox.Items.AddObject(Fn.GetName, Fn);
          ObjListBox.Checked[ObjListBox.Items.Count-1]:=True;
          ObjListBox.ItemIndex:=ObjListBox.Items.Count-1;
          ...
        end
      else
        Fn.Free;
    End;
```

У цьому методі створюється об’єкт програми Gran1, тип якого визначається згідно значення змінної `GlobFType`, а значення цієї змінної залежить від вибору у списку типів ObjComboBox. Після створення об’єкта викликається метод зміни його параметрів (виразу функції, виразів для кінців відрізка задання тощо `Fn.Change(C);`), вказується, що список об’єктів змінився і потребує запису у файл при виході із програми `(NeedSave:=True;`), обчислюються мінімальні і максимальні значення на осях для автоматичного масштабування (`Fn.MmF(GRI)`); створений об’єкт додається до списку об’єктів

(`ObjListBox.Items.AddObject(Fn.GetName, Fn);`),

відмічається

```
 (ObjListBox.Checked[ObjListBox.Items.Count1]:=True;
 ObjListBox.ItemIndex:=ObjListBox.Items.Count-1;).
```

За методом

```
    procedure TObjList.MakeFuncsList(Funcs:TFuncs; OLSel:
Boolean);
```

формується список об’єктів програми Gran1 для виконання тієї чи іншої операції; якщо параметр цього методу `OLSel` має значення `True`, у список потраплять всі відмічені у візуальному списку об’єкти, якщо ж цей параметр має значення `False`, до списку потрапить тільки поточний об’єкт з візуального списку.

Метод

```
procedure TObjList.MakeFuncsListG(FuncsG:TFuncs);
```

призначений для формування списку об’єктів, графіки яких потрібно будувати:

```
procedure TObjList.MakeFuncsListG(FuncsG:TFuncs);
  var j: Integer;
Begin
  FuncsG.Clear;
  with ObjListBox do
  Begin
    if Items.Count=0 then Exit;
    for j:=0 to Items.Count-1 do
      if TFn(Items.Objects[j]).GBuild then
        FuncsG.Add(Items.Objects[j]);
  End;
End;
```

Для зміни поточного об’єкта викликається метод `ChngObjPopupClick(Sender: TObject)`, який також можна прив’язати до команди контекстного меню:

```
procedure TObjList.ChngObjPopupClick(Sender: TObject);
  var C,S:Boolean;
      I: Integer;
Begin
  with ObjListBox do
    Begin
      if Items.Count>0 then
        Begin
          (Items.Objects[ItemIndex] as TFn).Change(C);
          if C then
```

```
                Begin
                  NeedSave:=True;
                  (Items.Objects[ItemIndex] as TFn).MmF(GRI);
                  I:=ItemIndex;
                  S:=Checked[I];
                  Items[ItemIndex]:=(Items.Objects[ItemIndex] as
TFn).GetName;
                  Checked[I]:=True;
                  Checked[I]:=S;
                  FindParams;
                End;
              C:=C and (Items.Objects[ItemIndex] as TFn).FGraph;
              if C then
                Begin
                  SetMasshtab;
                  GRI.UpToDate;
                End;
            End;
        End;
      ObjListBoxClick(Sender);
    End;
```

Метод для запису списку об’єктів на диск у ini-файл виглядає наступним чином:

```
    procedure TObjList.SaveFuncs(FileName: String);
    var f: TIniFile;
        tf: TextFile;
        i: Integer;
        Sec: String;
    Begin
      try
      try
      AssignFile(tf,FileName);
      Rewrite(tf);
      Writeln(tf,'[Gran1]');
      for i:=1 to ObjListBox.Items.Count do
        Writeln(tf,'[Obj'+IntToStr(i)+']');
```

```
  CloseFile(tf);
  f:=TIniFile.Create(FileName);
  f.WriteInteger('Gran1','Num',ObjListBox.Items.Count);
  f.WriteString('Gran1','XName', GRI.grXName);
  f.WriteString('Gran1','YName', GRI.grYName);
  f.WriteFloat('Gran1','MinArg', GRI.GraphRang.MinArg);
  f.WriteFloat('Gran1','MaxArg', GRI.GraphRang.MaxArg);
  f.WriteFloat('Gran1','MinFunc', GRI.GraphRang.MinFunc);
  f.WriteFloat('Gran1','MaxFunc', GRI.GraphRang.MaxFunc);
  f.WriteInteger('Gran1','Coord', Integer(GRI.Coord));
  f.WriteBool('Gran1','MasshAuto',GRI.MasshAuto);
  for i:=0 to ObjListBox.Items.Count-1 do
  Begin
    Sec:='Obj'+IntToStr(i+1);
    (ObjListBox.Items.Objects[i] as TFn).Save(f,Sec);
    f.WriteBool(Sec,'Checked',ObjListBox.Checked[i]);
  End;
  NeedSave:=False;
  finally
  f.Free;
  End;
  except
    MyMessageDlg('Unable to save!',mtError,[mbOK],0);
  End;
End;
```

Для зчитування списку об’єктів з ini-файлу використовується метод

```
procedure TObjList.ReadFuncs(FileName: String);
var f: TIniFile;
    n,i: Integer;
    Sec,TypeName: String;
    G: Mmtype;
    CT: CoordType;
procedure CreateFunc;
var Fn:TFn;
    C: Boolean;
Begin
```

```
      Sec:='Obj'+IntToStr(i);
      TypeName:=f.ReadString(Sec,'ObjType','');
      C:=f.ReadBool(Sec,'Checked',True);
      if TypeName='Normal' then
        Fn:=TFn.Load(f,Sec);
      if TypeName='Param' then
        Fn:=TFp.Load(f,Sec);
      if TypeName='TwoArg' then
        Fn:=TFg.Load(f,Sec);
      if TypeName='Circ' then
        Fn:=TFc.Load(f,Sec);
      if TypeName='Polar' then
        Fn:=TFf.Load(f,Sec);
      if TypeName='Stat' then
        Fn:=TFs.Load(f,Sec);
      if TypeName='Lom' then
        Fn:=TFl.Load(f,Sec);
      if TypeName='Polinom' then
        Fn:=TFm.Load(f,Sec);
      Fn.MmF(GRI);
      ObjListBox.Items.AddObject(Fn.GetName, Fn);
      ObjListBox.Checked[ObjListBox.Items.Count-1]:=C;
      ObjListBox.ItemIndex:=ObjListBox.Items.Count-1;
      ObjListBoxClick(Self);
    End;

    Begin
      try
      try
      f:=TIniFile.Create(FileName);

      GRI.grXName:=f.ReadString('Gran1','XName','X');
      GRI.grYName:=f.ReadString('Gran1','YName','Y');

G.MinArg:=f.ReadFloat('Gran1','MinArg',GRI.GraphRang.MinArg);
```

```
G.MaxArg:=f.ReadFloat('Gran1','MaxArg',GRI.GraphRang.MaxArg);

G.MinFunc:=f.ReadFloat('Gran1','MinFunc',GRI.GraphRang.MinFunc);

G.MaxFunc:=f.ReadFloat('Gran1','MaxFunc',GRI.GraphRang.MaxFunc);

CT:=CoordType(f.ReadInteger('Gran1','Coord',Integer(GRI.Coord)));

GRI.MasshAuto:=f.ReadBool('Gran1','MasshAuto',GRI.MasshAuto);
      GRI.SetGraphRang(G);
      GRI.Coord:=CT;
      n:=f.ReadInteger('Gran1','Num',0);
      for i:=1 to n do
        CreateFunc;
      FindParams;
      finally
      f.Free;
      End;
      except
      End;
      GRI.UpToDate;
    End;
```

2.6.6. Gran1. Клас для реалізації вікна “Графік”. Клас для реалізації вікна “Графік” `TGr = class(TForm)` повинен містити візуальну компоненту, в якій і будуть будуватися графіки математичних об’єктів (`I:TMasshImage;`), а також візуальні компоненти для показу мінімальних і максимальних значень на кожній із осей координат і поточних математичних координат курсору мишки:

```
MasshBotStatus: TStatusBar;
 CoordStatus: TStatusBar;
 MasshTopStatus: TStatusBar;
```

Розглянемо тепер деякі важливі методи цього класу.

Метод

```
procedure TGr.GraphsBuild;
```

призначений для побудови графіків відповідних математичних об’єктів. Для побудови призначений цикл

```
with ObjList.FuncsG do
Begin
  if Count=0 then Exit;
  for j:=0 to Count-1 do
    TFn(Items[j]).GraphBuild(I);
End;
```

за яким будуються графіки всіх об’єктів, які є у списку об’єктів, відібраних для побудови.

За методом

```
procedure TGr.Osi;
```

будуються осі координат.

За методом

```
procedure TGr.IMasshUpdate(Sender: TObject);
```

будуються осі координат, виводяться мінімальні і максимальні значення на осях координат і вбудуються графіки математичних об’єктів. Цей метод викликається кожного разу, коли генерується подія `FMasshUpdate: TNotifyEvent;`. Нагадаємо, що цю подію можна згенерувати, викликавши метод `UpToDate()` візуальної компоненти `TMasshImage`, що безпосередньо призначена для побудови. Код цієї процедури:

```
procedure TGr.IMasshUpdate(Sender: TObject);
var j: Integer;
Begin
  if WindowState=wsMinimized then Exit;
  Osi;
  MasshTopStatus.SimpleText:='MinX='+

FloatToStrF(I.GraphRang.MinArg,ffGeneral,ngDigits,ngPower)+
       'MaxY='+
```

```
FloatToStrF(I.GraphRang.MaxFunc,ffGeneral,ngDigits,ngPower);
     MasshBotStatus.SimpleText:='MinY='+

FloatToStrF(I.GraphRang.MinFunc,ffGeneral,ngDigits,ngPower)+
         '  MaxX='+

FloatToStrF(I.GraphRang.MaxArg,ffGeneral,ngDigits,ngPower);
      GraphsBuild;
    ...
    End;
```

За методом

```
procedure TGr.FormResize(Sender: TObject);
```

опрацьовується ситуація зміни розміру вікна “Графік”, оскільки в цьому випадку потрібно змінити і розмір масиву зберігання зображення візуальної компоненти, в якій будуються графіки:

```
procedure TGr.FormResize(Sender: TObject);
Begin
    I.Picture.BitMap.Width:=I.Width;
  I.Picture.BitMap.Height:=I.Height;
  I.UpToDate;
End;
```

Щоб виводити поточні математичні координати при русі курсору мишки, потрібно описати процедуру опрацювання події, що виникає при русі мишки. У найпростішому варіанті ця процедура виглядатиме так:

```
procedure TGr.IMouseMove(Sender: TObject; Shift: TShiftState;
X,
  Y: Integer);
  var tx: Extended;
        ty: Extended;
 Begin
  case I.Coord of
    ctDec:
    Begin
       tx:=I.WinX(X);
```

```
            ty:=I.WinY(Y);
            CoordStatus.SimpleText:='X='+
                    FloatToStrF(tx,ffGeneral,ngDigits,ngPower)+
                    '
Y='+FloatToStrF(ty,ffGeneral,ngDigits,ngPower);
        ctPolar:
        Begin
            tx:=I.WinF(X,Y);
            ty:=I.WinR(X,Y);
            CoordStatus.SimpleText:='R='+
                  FloatToStrF(ty,ffGeneral,ngDigits,ngPower)+
                  '
F='+FloatToStrF(tx,ffGeneral,ngDigits,ngPower)+
                 '
('+FloatToStrF(tx/pi*180,ffGeneral,ngDigits,ngPower)+
                 chr(176)+')';
        End;
    End;
```

У програмі Gran1 цей опис значно складніший, оскільки за його допомогою здійснюється реагування на вказування мишкою ділянки візуальної компоненти, в якій будуються графіки, а також на деякі інші події.

2.6.7. Gran1. Класи для реалізації вікон “Відповіді” та головного вікна програми. Клас для реалізації вікна “Відповіді” `type TAns = class(TForm)` – це звичайне вікно, в якому розміщено компонент AnsMemo: TMemo; для відображення відповідей, які отримуються в результаті виконання тих чи інших операцій за програмою Gran1.

Всі згадані вище вікна об’єднуються у головному вікні програми Gran1, якому відповідає клас `T__Main = class(TForm)`. В даному вікні міститься головне меню, в якому є всі команди для роботи з програмою Gran1. Побудувати таке меню можна за допомогою стандартного візуального конструктора меню. Розглянемо деякі команди, реалізовані у

класі T__Main. За програмою Gran1 її установки зберігаються у так званому .ini файлі, про який згадувалося вище. Це реалізовано у методі

```
procedure T__Main.SaveIni;
var f: TIniFile;
    tf: TextFile;
    s: String;
    n: Integer;
Begin
  s:=Application.ExeName;
  n:=Length(s);
  s[n-2]:='I';
  s[n-1]:='N';
  s[n]:='I';
   try
  try
  f:=TIniFile.Create(s);
  f.WriteInteger('MainWindow','Left',Left);
  f.WriteInteger('MainWindow','Top',Top);
  f.WriteInteger('MainWindow','ClientWidth',ClientWidth);
  f.WriteInteger('MainWindow','ClientHeight',ClientHeight);

f.WriteInteger('MainWindow','WindowState',Integer(WindowState));
  f.WriteString('MainWindow','Language', ActualLanguage);
  f.WriteInteger('MainWindow','Digits', ngDigits);

f.WriteBool('MainWindow','AutoSizeWin',AutoWinItem.Checked);
  f.WriteInteger('Objects','Left',ObjList.Left);
  f.WriteInteger('Objects','Top',ObjList.Top);
  f.WriteInteger('Objects','Width',ObjList.Width);
  f.WriteInteger('Objects','Height',ObjList.Height);

f.WriteInteger('Objects','ObjMemoHeight',ObjList.ObjMemo.Height);
  f.WriteBool('Graph','SGVisible',Gr.SG.Visible);
  f.WriteBool('Graph','MVisible',Gr.MasshTopStatus.Visible);
  f.WriteInteger('Graph','SGHeight',Gr.SG.Height);
  f.WriteInteger('Graph','Left',Gr.Left);
```

```
      f.WriteInteger('Graph','Top',Gr.Top);
      f.WriteInteger('Graph','Width',Gr.Width);
      f.WriteInteger('Graph','Height',Gr.Height);
      f.WriteInteger('Graph','TextColor',Gr.I.grTextColor);
      f.WriteInteger('Graph','BackColor',Gr.I.grBackColor);
      f.WriteInteger('Graph','MarkColor',Gr.I.grMarkColor);
      f.WriteInteger('Graph','AxeColor',Gr.I.grAxeColor);
      f.WriteInteger('Graph','AddColor',Gr.I.grAddColor);
      f.WriteInteger('Graph','MouseColor',Gr.I.grMouseColor);
      f.WriteString('Graph','XName', gr.I.grXName);
      f.WriteString('Graph','YName', gr.I.grYName);
      f.WriteInteger('Answers','Left',Ans.Left);
      f.WriteInteger('Answers','Top',Ans.Top);
      f.WriteInteger('Answers','Width',Ans.Width);
      f.WriteInteger('Answers','Height',Ans.Height);

f.WriteString('Graph','grLabelFontName',gr.I.grLabelFont.Name);

f.WriteInteger('Graph','grLabelFontSize',gr.I.grLabelFont.Size);
      f.WriteString('Graph','grAxeFontName',gr.I.grAxeFont.Name);

f.WriteInteger('Graph','grAxeFontSize',gr.I.grAxeFont.Size);
      finally
      f.Free;
      End;
      except
      End;
    End;
```

Завантаження цих параметрів відбувається автоматично у процесі запуску програми Gran1 шляхом виклику методу

```
    procedure T__Main.LoadIni;
    var f: TIniFile;
        s, FN: String;
        n, i: Integer;
```

```
    Begin
      s:=Application.ExeName;
      n:=Length(s);
      s[n-2]:='I';
      s[n-1]:='N';
      s[n]:='I';
      try
      try
      f:=TIniFile.Create(s);

ActualLanguage:=f.ReadString('MainWindow','Language','”Українська
');
      i:=LanguagesList.LangName.IndexOf(ActualLanguage);
      if i>=0 then
      Begin
        FN:=LanguagesList.LangFileName[i];
        LoadLangdata(Application, FN);
      End;
      Top:=f.ReadInteger('MainWindow','Top',30);
      Left:=f.ReadInteger('MainWindow','Left',0);
      ClientWidth:=f.ReadInteger('MainWindow','ClientWidth',795);

ClientHeight:=f.ReadInteger('MainWindow','ClientHeight',506);
      ngDigits:=f.ReadInteger('MainWindow','Digits',4);

AutoWinItem.Checked:=f.ReadBool('MainWindow','AutoSizeWin',False)
;

WindowState:=TWindowState(f.ReadInteger('MainWindow','WindowState
',Integer(wsNormal)));
      ObjList.Top:=f.ReadInteger('Objects','Top',0);
      ObjList.Left:=f.ReadInteger('Objects','Left',440);
      ObjList.Width:=f.ReadInteger('Objects','Width',326);
      ObjList.Height:=f.ReadInteger('Objects','Height',333);
```

```
ObjList.ObjMemo.Height:=f.ReadInteger('Objects','ObjMemoHeight',1
00);
      Ans.Top:=f.ReadInteger('Answers','Top',333);
      Ans.Left:=f.ReadInteger('Answers','Left',440);
      Ans.Width:=f.ReadInteger('Answers','Width',326);
      Ans.Height:=f.ReadInteger('Answers','Height',167);
      Gr.Top:=f.ReadInteger('Graph','Top',0);
      Gr.Left:=f.ReadInteger('Graph','Left',0);
      Gr.Width:=f.ReadInteger('Graph','Width',440);
      Gr.Height:=f.ReadInteger('Graph','Height',500);

Gr.I.grTextColor:=f.ReadInteger('Graph','TextColor',clBlack);

Gr.I.grBackColor:=f.ReadInteger('Graph','BackColor',clLtGray);

Gr.I.grMarkColor:=f.ReadInteger('Graph','MarkColor',clGray);
      Gr.I.grAxeColor:=f.ReadInteger('Graph','AxeColor',clBlack);
      Gr.I.grAddColor:=f.ReadInteger('Graph','AddColor',clWhite);

Gr.I.grMouseColor:=f.ReadInteger('Graph','MouseColor',clWhite);
      Gr.I.grXName:=f.ReadString('Graph','XName','X');
      Gr.I.grYName:=f.ReadString('Graph','YName','Y');


Gr.I.grLabelFont.Name:=f.ReadString('Graph','grLabelFontName','Sy
stem');

Gr.I.grLabelFont.Size:=f.ReadInteger('Graph','grLabelFontSize',10
);

Gr.I.grAxeFont.Name:=f.ReadString('Graph','grAxeFontName','System
');

Gr.I.grAxeFont.Size:=f.ReadInteger('Graph','grAxeFontSize',10);
```

```
Gr.MasshTopStatus.Visible:=f.ReadBool('Graph','MVisible',True);

Gr.MasshBotStatus.Visible:=f.ReadBool('Graph','MVisible',True);
      Gr.MasshBotStatus.Top:=Gr.ClientHeight-
Gr.MasshBotStatus.Height;
      Gr.SG.Visible:=False;
      if not Gr.SG.Visible then Gr.Splitter1.Free;
      Gr.SG.Height:=f.ReadInteger('Graph','SGHeight',44);
      Gr.FormResize(Self);
      finally
      f.Free;
      End;
      except
      End;
    End;
```

За програмою Gran1 можна копіювати до буфера обміну зображення у вікні “Графік”, якщо поточним є це вікно, або список виразів із вікна “Список об’єктів”, якщо поточним є це вікно. Реалізовано це шляхом надсилання повідомлення `Screen.ActiveControl.Perform(WM_COPY,0,0)` до операційної системи у методі

```
    procedure T__Main.Copy1Click(Sender: TObject);
    Begin
      Screen.ActiveControl.Perform(WM_COPY,0,0);
      if Screen.ActiveForm is TGr then
        ClipBoard.Assign((Screen.ActiveForm as
TGr).I.Picture.BitMap);
      if Screen.ActiveForm is TObjList then
        with (Screen.ActiveForm as TObjList).ObjListBox do
        ClipBoard.AsText:=Items.Strings[ItemIndex];
    End;
```

Тут розглянуто тільки кістяк програми Gran1. Поза розглядом залишилось багато інших методів, за допомогою яких реалізується

функціональність програми. Проте вивчення даного кістяка дозволить студентам отримати певні знання щодо моделювання математичних об'єктів, пов'язаних з функціональними залежностями, взаємозв'язки між такими моделями, принципи і конкретні приклади побудови досить великої програми, і застосувати отримані знання у подальшій педагогічній діяльності, пов'язаній з навчанням основ програмування і об'єктно-орієнтованого програмування, а також у майбутніх наукових дослідженнях, можливо і для написання власних педагогічних програмних засобів для комп'ютерної підтримки навчання математики.

2.6.8. Формування у студентів системи компетентностей стосовно створення математичних ППЗ. Розглянутий вище матеріал щодо створення математичних ППЗ є теоретичною основою формування у студентів відповідних практичних компетентностей. Формування таких компетентностей повинно відбуватися у процесі виконання лабораторних робіт, завдання до яких наведені нижче. В результаті їх виконання студенти отримають набір модулів, які можна буде використовувати при створенні математичних ППЗ, та приклади найпростіших ППЗ.

Завдання 1.

Розробити клас TFunc:

```
TFunc=class(TObject)
  Tx,St: String;
  Arr: RealArray;
  public
    constructor Create(SF: String);
    function PointX(Arg:extended): Extended; virtual;
    function PointY(Arg:extended): Extended; virtual;
    function Simpson(AI, BI: Extended): Extended;
    procedure ChangeFunc(SF: String; var Chng:
Boolean);virtual;
    function Eval(ArgX, ArgY: Extended): Extended;
End;
```

`Tx, St, Arr` – поля для збереження виразу. Конструктор `Create`, методи `Eval` (обчислення значення виразу) та `ChangeFunc` (зміна виразу функції) наведено в додатку.

За методом `PointX` повинно повертатися значення x-координати у декартовій системі координат, за методом `PointY` – значення y-координати у декартовій системі координат, за методом `Simpson` – значення інтеграла від функції на відрізку *AI,BI* (застосувати будь-який чисельний метод знаходження інтеграла, наприклад метод трапецій чи метод парабол).

Клас розмістити у окремому модулі `NGFunc`, підключивши до нього модуль `NGMath` (наведено у додатку). Даний модуль містить процедури, за якими на основі рядка запису виразу будують певну структуру даних, що використовується у методі `Eval()` для обчислення значення цього виразу.

Написати консольну програму, за якою вводиться значення виразу з клавіатури і обчислюється значення інтеграла від цього виразу на відрізку, також введеному з клавіатури.

Завдання 2.

Розробити клас TRng:

```
TRng=class(TFunc)
    private
    FValue: Extended;
    function GetValue: Extended;
    public
    constructor Create(SA: String);
    procedure ChangeFunc(SF: String; var Chng: Boolean);
override;
    published
    property  Value: Extended read GetValue write FValue;
  End;
```

Клас призначено для задання виразу, через який визначається один з кінців відрізка задання майбутньої залежності між змінними. Від класу `NGFunc` він відрізняється наявністю властивості `Value` – значення виразу, за

яким визначається кінець відрізка. Функція `GetValue` повинна бути призначена для обчислення значення виразу, конструктор `Create` – для виклику батьківського конструктора і обчислення значення властивості `Value`; метод `ChangeFunc` – для виклику батьківського методу і у випадку вдалої зміни виразу обчислення значення властивості `Value`. клас розмістити у модулі `NGFuncs`.

Завдання 3.

Розробити клас `TMasshImage`, для опису візуальної компоненти, за якою будуть будуватися графіки залежностей:

```
    TMasshImage = class(TImage)
      private
        { Private declarations }
        FMasshUpdate: TNotifyEvent;
      public
        { Public declarations }
        GraphRang: MmType;
        constructor Create(AOwner: TComponent); override;
        destructor Destroy; override;
        function XWin(X: Extended): Integer;
        function YWin(Y: Extended): Integer;
        function WinX(X: Integer): Extended;
        function WinY(Y: Integer): Extended;
        function WinR(X,Y: Integer): Extended;
        function WinF(X,Y: Integer): Extended;
        procedure SetGraphRang(G: MmType);
        procedure UpToDate;
      published
        { Published declarations }
        property OnMasshUpdate: TNotifyEvent read FMasshUpdate
write FMasshUpdate;
    End;
```

Призначення методів знайти у описі цього класу. Клас розмістити у модулі `MImage`. Передбачити реєстрацію цього класу як візуальної компоненти на закладці `Samples` у середовищі програмування Lazarus.

Завдання 4.

Описати клас `TFn` для визначення залежності Y=Y(X):

```
TFn=class(TFunc)
   FArg:Char;
   Mm: MmType;
   A, B: TRng;
   FColor: TColor;
   BGraphPoints: Integer;
   constructor Create(SFy,SA,SB: String; GColor: TColor);
   destructor Destroy; override;
   procedure MmF; virtual;
   function GetType: FType; virtual;
   procedure GraphBuild(I: TMasshImage); virtual;
   procedure Change(var Chng: Boolean); virtual;
   procedure Info(M: TMemo); virtual;
   function GetColor: TColor;
End;
```

У конструктор передаються вираз залежності, вирази для кінців відрізка та колір графіка.

За методом `Change` повинне викликатися допоміжне вікно, у якому користувач повинен ввести нові вирази для залежності та кінців відрізка, а також вибрати колір графіка. Введені користувачем вирази потрібно перевірити на правильність і якщо перевірка пройшла успішно, замінити попередні. Якщо така зміна відбулась, параметр `Chng` повинен набути значення True.

За методом `Info` у його параметр типу `TMemo` передаються дані про залежність – вираз залежності, вирази кінців відрізка задання, мінімальні та максимальні значення аргументу та залежної змінної.

За функцією `GetColor` повертається колір графіка. Призначення інших методів можна знайти у описі класу. Клас потрібно розмістити у модулі `NGFuncs`.

Завдання 5.

Розробити програму для побудови графіка залежності. Вікно програми повинно містити компонент типу `TLabel` для відображення імені залежності, компонент типу TMemo для виведення даних про залежність, компонент типу `TMasshImage` для побудови в ньому графіка залежності; кнопки “Змінити залежність” та “Побудувати графік”. Залежність з початковим виразом `'x'`, відрізком задання ‘-5’, ‘5’ та червоним кольором графіка повинна бути створена автоматично при запуску програми.

Завдання 6.

Розробити клас `TFp`, для введення залежності, заданої параметрично:

```
TFp=class(TFn)
    Fx: TFunc;
    constructor Create(SFy, SFx, SA, SB: String; GColor:
TColor);
    destructor Destroy; override;
    function PointX(Arg: Extended): Extended; override;
    function PointY(Arg: Extended): Extended; override;
    function GetType: FType; override;
    procedure Change(var Chng:Boolean); override;
    procedure Info(M: TMemo); override;
End;
```

У конструктор передаються вирази $y=y(t)$ та $x=x(t)$, вирази для кінців відрізка та колір графіка.

За методом `Change` повинно викликатися допоміжне вікно, у якому користувач повинен ввести нові вирази для *y=y(t)* і *x=x(t)* та кінців відрізка, а також вибрати колір графіка. Введені користувачем вирази потрібно перевірити на правильність і якщо все гаразд, замінити попередні. Якщо така зміна відбулась, параметр `Chng` повинен набути значення True.

За методом `Info` у його параметр типу `TMemo` передаються дані про залежність – вирази *y=y(t)* та *x=x(t),* вирази кінців відрізка задання, мінімальні та максимальні значення аргументу та залежної змінної.

Клас потрібно розмістити у модулі `NGFuncs`.

Аналогічно до попереднього, розробити клас `TFf` для задання залежності у полярних координатах:

```
TFf=class(TFn)
    constructor Create(SFy,SA,SB: String; GColor: TColor);
    function PointX(Arg:extended):extended;override;
    function PointY(Arg:extended):extended;override;
    function GetType:FType;override;
    procedure Info(M: TMemo); override;
    procedure Change(var Chng:Boolean); override;
End;
```

Клас потрібно розмістити у модулі `NGFuncs`.

Завдання 7.

Вдосконалити програму для побудови графіка залежності. У вікно програми потрібно додати компонент для вибору залежності, що буде вводитись, та кнопку “Створити залежність”, призначену для створення нової залежності за вибраною користувачем залежністю. У процедурі обробника події кнопки “Створити залежність” передбачити вилучення попередньої залежності.

РОЗДІЛ 3
ІНФОРМАЦІЙНЕ МОДЕЛЮВАННЯ У ВИВЧЕННІ БАЗ ДАНИХ ТА ОСНОВ ШТУЧНОГО ІНТЕЛЕКТУ

3.1. Інформаційне моделювання у вивченні баз даних та СУБД

3.1.1. Сучасні тенденції у використанні баз даних та моделей даних. Як відомо, кількість всеможливих інформаційних матеріалів і повідомлень, що циркулюють в інформаційному просторі, зростає небаченими темпами. Відповідно виникає потреба у надійному зберіганні цих даних, а також у ефективному маніпулюванні ними. Всім цим потребам відповідають бази даних, оскільки для них накопичено багато напрацювань щодо підтримки цілісності даних, швидкого доступу до них, розділення прав доступу до даних між користувачами різного рівня.

Поряд з широким використанням баз даних у традиційних сферах, наприклад банківській, управлінській, можна спостерігати і застосування баз даних у тих галузях, де вони раніше не використовувалися.

Так, вже досить давно спостерігаються тенденції щодо переходу від зберігання даних у окремих файлах до зберігання їх у базах даних. Наведемо кілька прикладів. На початку розвитку мережі Інтернет та служб WWW дані в основному зберігалися в html-документах та окремих файлах інших типів (наприклад, графічних). Зараз такий підхід виправданий тільки для невеликих, любительських сайтів, де даних мало і оновлюються вони досить рідко. У всіх інших випадках дані сайту зберігаються у базі даних.

В операційній системі Windows більшість параметрів її роботи зберігаються у спеціальній базі даних, основаній на ієрархічній моделі – “Реєстр” (Register).

Іншим прикладом переходу до використання баз даних для більш зручного маніпулювання даними є сучасні підходи до роботи з музичними

матеріалами. Раніше музичні програвачі (такі як дуже популярний свого часу WinAmp і багато його аналогів) орієнтувались в основному на програвання музичних файлів із папки, вказаної користувачем. Тобто користувач повинен був сам створити папку для зберігання музичних файлів, скопіювати у цю папку відповідні музичні файли, а потім вказати на музичному програвачеві цю папку для того, щоб почати програвати ці файли. Такий спосіб використання музичних даних був дуже незручним, наприклад, якщо користувач хотів відтворити всі музичні дані конкретного виконавця або музичні дані у вибраному жанрі, то примітивний механізм списків програвання (playlists) не дуже зручний для цього. В сучасних музичних програвачах (наприклад, iTunes, Windows Media Player, Real Player, програвачі багатьох сучасних мобільних телефонів та смартфонів та інші) для зберігання музичних файлів створюються спеціальні бази даних, використання яких дає змогу легко сортувати та групувати музичні дані відповідно до спеціальних даних (музичних тегів), таких як прізвище композитора, прізвище виконавця, альбом, жанр і інші, що записані у музичних файлах.

Схожа тенденція спостерігається і щодо зберігання та використання фотографій та інших графічних файлів. При роботі сучасних фотопереглядачів, таких як, наприклад, iPhoto, Picasa фотографії зберігаються у базах даних, що дозволяє легко відібрати фотографії за певну дату, або зроблені за допомогою певної моделі фотокамери, або ті, на яких є зображення певної людини, а якщо у фотокамері передбачено можливість геотеггінгу (запису у файл фотографії даних про географічні координати зйомки), то і відібрати фотографії, зроблені у певному районі земної кулі.

Найбільш масштабним прикладом переходу до зберігання і маніпулювання даними за допомогою баз даних є надання властивостей бази даних файловій системі. Першим кроком у цьому напрямі була BeOS – операційна система для настільних персональних комп’ютерів,

розроблена компанією Be. Важливою особливістю цієї операційної системи була 64-бітна файлова система BeFS з можливістю журналювання. Особливістю цієї файлової системи є і покращений підхід до ідентифікації файлів даних. Крім традиційної прив'язки розширення файлу до програми, за якою опрацьовується файл, в BeOS використовується ідентифікація, заснована на так званих MIME-типах, достатньо стандартизований спосіб ідентифікації файлів і зв'язку їх з програмами. Використання цих розширених файлових атрибутів, що індексуються, наближають функціональність цієї файлової системи до реляційних баз даних, тобто баз даних, основаних на реляційній моделі [319], [200].

У сучасних операційних системах, таких як MAC OS X, Windows XP SP3 і старших вміст дисків автоматично індексується у фоновому режимі. В процесі цієї індексації у спеціальну базу даних заносяться як імена файлів, так і зміст більшості з них. Це дозволяє проводити миттєвий пошук за іменами папок та файлів, їх змістом, за поштою, адресною книгою, календарем тощо. У операційній системі MAC OS X результат пошуку можна, наприклад, розмістити у "розумній" папці (Smart Folder), і вона буде автоматично оновлюватися при додаванні або вилученні будь-яких даних на комп'ютері. "Розумна" папка містить файли, що згруповані за певним критерієм, а не ті, що фізично знаходяться поруч. Таким чином, один і той самий файл може бути зафіксований в кількох розумних папках і, крім того, фізично знаходитися в іншому місці. Отже зникає потреба дублювати, переміщувати та оновлювати файли, це буде зроблено автоматично [346].

У корпорації Microsoft триває розробка файлової системи WinFS.

Суттю WinFS є так звана модель структурованих даних, тобто механізм, за яким постійно опрацьовуються (адмініструються і структуруються) певні сутності (у дослівному перекладі «предмети» – items). В Items використовуються описові елементи, що виходять за поняття файлу. Слід розуміти, що ці описові елементи не зберігаються у

файлі, а повністю належать і управляються WinFS. Отже, фізична структура файлів на рівні файлової системи NTFS не зазнає яких-небудь змін. У даній схемі як сутність можна реєструвати не лише файли, але і, наприклад, контакти, улюблені посилання в Інтернеті, листи, дати, бренди, дані про виробників тощо, тобто будь-які атрибути, встановлені за замовчуванням або призначені користувачем.

З точки зору користувачів, використання items знімає необхідність у визначенні фізичного місця розташування файлів. Замість цього у Windows дані організовуються, залежно від їх вмісту, у віртуальні папки. При пошуку даних призначений для користувача критерій типу «Всі фотографії з відпусток за останні два роки» (атрибути «тип файлу», «звідки» і «за який період часу») тепер замінює відомості про формат файлу, автора і розташування [350].

Таким чином, робота користувача з файловою системою наближається до роботи з базою даних.

Таке широке використання баз даних, заснованих на різних моделях даних, у найрізноманітніших галузях інформаційних технологій є основою для поглибленого вивчення питань, пов’язаних з базами даних, і насамперед питань щодо моделей даних, у курсі інформатики в педагогічному університеті.

Розглянемо питання, пов’язані з моделюванням даних в базах даних.

Модель даних – це деяка абстракція, в якій знаходять своє відображення найбільш важливі аспекти об’єктів визначеної предметної галузі, а другорядні – ігноруються. Модель даних можна розглядати як деяку цільову модель предметної галузі. У моделі даних розрізняють три головні складові:

1) структурна частина, за якою визначаються правила породження допустимих для даної системи управління базами даних (СУБД) видів структур даних;

2) управляюча частина, за якою визначаються можливі операції над такими структурами;
3) класи обмежень цілісності даних, які можуть бути реалізовані засобами цієї системи [34].

В структурній частині визначається, чим логічно є база даних, через класи обмежень цілісності визначаються засоби опису коректних станів бази даних, а через управляючу частину визначаються способи переходу між станами бази даних і способи отримання даних із бази даних [168]. Управляюча частина містить специфікацію однієї або кількох мов, призначених для створення запитів до бази даних.

Вперше поняття моделі даних було введено Едгаром Коддом для опису реляційного підходу до організації бази даних, але найбільш розповсюджена трактовка цього поняття належить Кристоферу Дейту [79].

Кожна СУБД будується на основі певної моделі даних, наприклад, в основі ієрархічних СУБД лежить ієрархічна модель даних, в основі мережевих СУБД – мережева модель даних, в основі реляційних СУБД – реляційна модель даних. Розглянемо більш детально ці моделі даних.

3.1.2. Ієрархічна модель даних. Ієрархічна модель даних подається як дерево, у вузлах якого розміщено дані, а зв’язки між даними подано гілками дерева. Вузли дерева розподілено за рівнями. Кожен вузол нижчого рівня зв’язаний тільки з одним вузлом вищого. Дерево має тільки одну вершину (яку називають коренем), не підпорядковану ніяким іншим вершинам. Кожен вузол має своє ім’я (ідентифікатор). Вузли дерева є записами, що складаються з певної кількості полів, кожне з яких характеризується іменем та типом даних, що там зберігаються. У дереві є записи різних типів, що мають різні назви і складаються з різних полів.

Ієрархічна база даних може складатися з кількох дерев, у найпростішому випадку з одного дерева.

Наприклад, застосуємо ієрархічну модель для подання даних про жителів міста (модель схематична, для кожного типу записів набір полів не визначено) (Рис. 3.1):

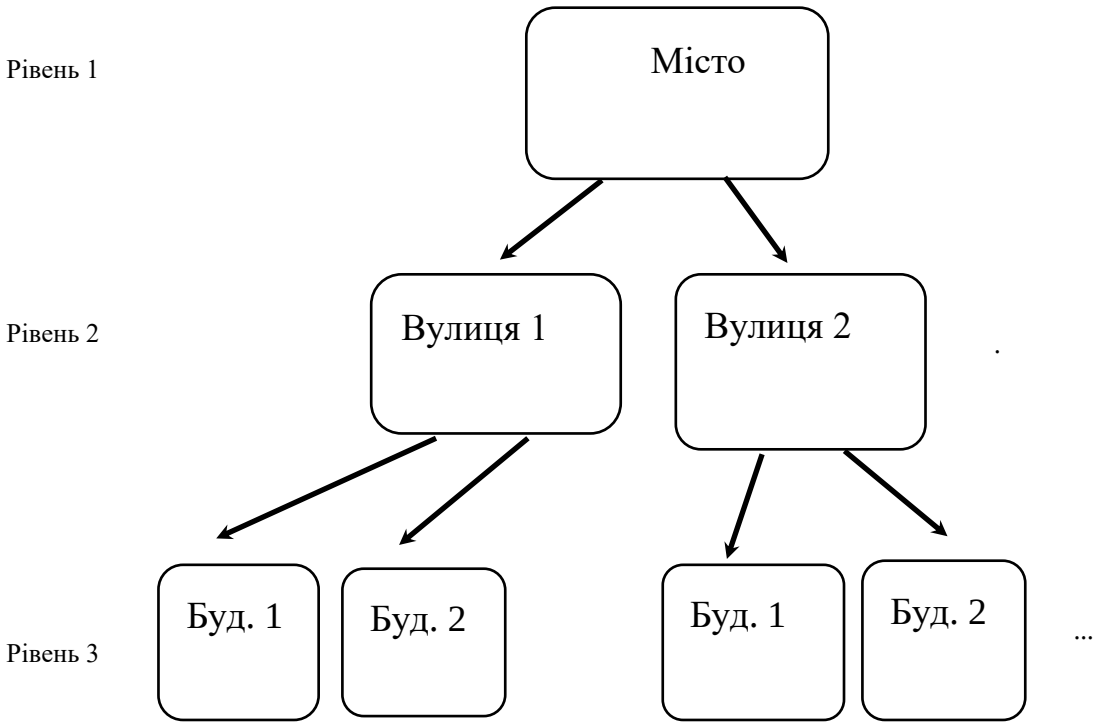

Рис. 3.1. Ієрархічна модель для подання даних про жителів міста.

Для ієрархічної бази даних визначається порядок обходу дерева: згори-вниз, зліва-направо.

На основі дерева (дерев) моделі будують екземпляри дерев з конкретними даними.

Прикладами маніпуляції у ієрархічній базі даних можуть бути:

1) знайти вказаний екземпляр;
2) перейти від одного екземпляра до іншого;
3) перейти від екземпляра до його нащадка (у наведеному прикладі перейти від вулиці до першого будинку на цій вулиці);
4) перейти від одного запису до іншого запису в порядку обходу ієрархії (наприклад, від будинку 1 до будинку 2);
5) вставити новий запис у вказану позицію;
6) вилучити поточний запис.

В ієрархічній моделі даних автоматично підтримується цілісність посилань між предками та нащадками за основним правилом: ніякий нащадок не може існувати без свого батька.

3.1.3. Мережева модель даних. Мережева модель щодо організації даних є розширенням ієрархічної моделі. Вона має ті самі основні складові: вузол, рівень, зв'язки, але якщо в ієрархічній моделі нащадок повинен мати тільки одного батька, в мережевій моделі у нащадка може бути довільна кількість батьків, тобто можуть існувати зв'язки між вузлами будь-яких рівнів. У вузлах розміщено записи, що складаються з полів.

Мережева база даних складається з набору записів і набору зв'язків між цими записами. Зв'язок визначається для батька і нащадка і складається з одного екземпляра типу запису батька і впорядкованого набору екземплярів типу запису нащадка. Для прикладу застосуємо мережеву модель для зображення зв'язків викладачів ВНЗ з групами, в яких вони працюють (Рис. 3.2):

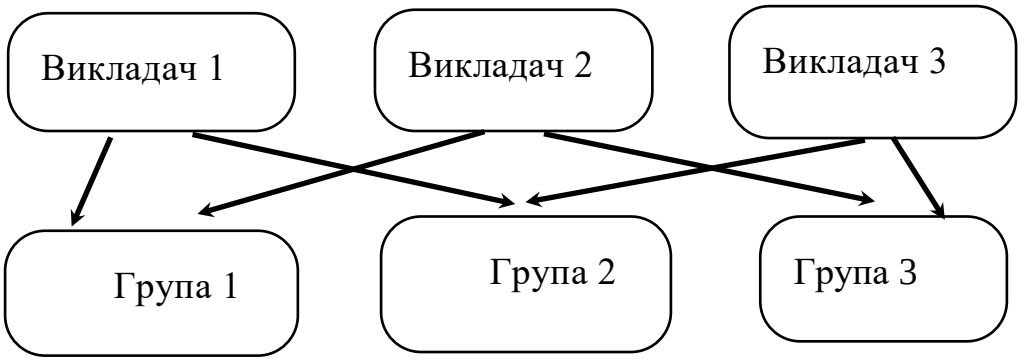

Рис. 3.2. Мережева модель для зображення зв'язків викладачів ВНЗ з групами, в яких вони працюють.

Розглянемо деякі основні типи операцій маніпулювання даними у мережевій базі даних:

- знайти конкретний запис у наборі однотипних записів (наприклад, викладача з прізвищем Петренко);

- перейти від батька до першого нащадка за певним зв’язком (наприклад до першої групи, у якій працює Петренко);
- перейти до наступного нащадка за певним зв’язком (наприклад до наступної групи, у якій працює Петренко);
- створити новий запис;
- модифікувати існуючий запис;
- включити у зв’язок;
- виключити із зв’язку;
- переставити в інший зв’язок.

3.1.4. Реляційна модель даних. Модель даних SQL. Основні ідеї реляційної моделі були запропоновані Едгаром Коддом у 1969 р. Подальший розвиток реляційної моделі пов’язаний з фірмою IBM. Там була розроблена мова маніпулювання даними SQL (Structured Query Language – мова структурованих запитів), яка базується на реляційній моделі даних. Вдосконаленням реляційної моделі у 90-х роках 20-го століття займалися Кристофен Дейт та Х’ю Дарвен [145].

При розробці реляційної моделі використовувалася математична теорія відношень. Відношенням R над множинами C_1, C_2, ..., $C_к$ називають підмножину декартового добутку цих множин. Множини C_1, C_2, ..., $C_к$ називають доменами. Кортеж – це набір з n значень, взятих з відповідних доменів. Оскільки термін „відношення” походить від англійського „relation”, ця модель і має назву “реляційна”.

Якщо подати відношення таблицею, то кортежі будуть її рядками. Стовпці такої таблиці називають атрибутами. Кожен атрибут містить значення з відповідного домену. Відношення і атрибути повинні бути іменованими.

Терміни, запропоновані Коддом, широко використовуються в теоретичних дослідженнях, пов’язаних з реляційною моделлю даних, але при розробці реляційних СУБД використовується не реляційна модель Кодда у “чистому” вигляді, а модель даних SQL, яка базується на

реляційній моделі, але не тотожна з нею. SQL-база даних подається як набір таблиць, кожна з яких в будь-який момент часу містить певну мультимножину (множину, що допускає повторення елементів) рядків, що відповідають заголовку таблиці. В цьому полягає перша відмінність моделі даних SQL від реляційної моделі даних. Другою суттєвою відмінністю є те, що для таблиці підтримується порядок стовпців, що відповідає порядку їх визначення. Таким чином, таблиця в SQL-моделі – це не зовсім відношення з реляційної моделі [218]. Оскільки вивчення СУБД у педагогічному ВНЗ має яскраво виражену прикладну спрямованість, доцільно реляційну модель подавати оглядово, а більш детально вивчати модель даних SQL, співставивши два набори термінів за допомогою наступної таблиці:

Таблиця 3.1.

Співставлення термінів реляційної моделі та SQL-моделі.

Терміни за Коддом	**Терміни СУБД**
Відношення	Таблиця
Кортеж	Запис
Атрибут	Поле
Домен	Тип значень поля

Надалі будемо користуватися термінологією, що застосовується в моделі даних SQL, і відповідно трактувати термін “реляційний”, як це і робиться в документації до реальних СУБД.

Таким чином, база даних в моделі даних SQL – це сукупність іменованих таблиць. Кожен рядок таблиці називається записом і містить дані про один об’єкт предметної галузі, наприклад, кожен запис таблиці “Студент” містить персональні дані про одного студента. Кожен запис складається з одного і того самого набору полів, що є характеристиками запису, наприклад у таблиці “Студент” полями можуть бути номер залікової книжки, прізвище, дата народження тощо. Кожне поле

характеризується ім’ям та типом даних. Ім’я поля зображується у заголовку відповідного стовпця таблиці.

Кожна таблиця повинна мати первинний ключ (Primary Key) – поле, або сукупність полів таблиці (мінімальну), що однозначно характеризують кожен запис таблиці, тобто всі значення ключа є гарантовано різними. Під мінімальністю первинного ключа треба розуміти наступне: якщо з множини полів первинного ключа вилучити хоча б одне поле, то у таблиці можуть з’явитися однакові значення ключа.

У таблиці може бути кілька варіантів вибору первинного ключа, всі ці можливі первинні ключі називають потенційними ключами (Candidate Key). Наприклад, у таблиці студентів потенційними ключами є “номер студентського квитка” та “прізвище, ім’я, по-батькові”, з них первинним ключем краще вибрати “номер студентського квитка”, як такий, що містить більш короткі значення.

Взагалі, всі дані у реляційній базі даних можна тримати в одній таблиці, але це дуже неефективно внаслідок значного дублювання даних. Наприклад, якщо тримати в одній таблиці як персональні дані студента, так і дані про отримані ним упродовж навчання оцінки, то для кожної нової оцінки потрібно буде продублювати персональні дані студента. Тому логічно для зберігання таких даних створити дві таблиці: одну з персональними даними студентів, іншу – з даними про отримані ними оцінки.

Для встановлення зв’язку між такими таблицями в реляційній моделі визначено поняття зовнішнього ключа. Зовнішній ключ (Foreign Key) – це сукупність полів таблиці, значення яких є одночасно і значеннями первинного або потенційного ключа іншої таблиці. Якщо таблиці “Студенти” та “Оцінки” мають наступну структуру (знак “+” вказує, що поле входить у ключ):

Таблиця 3.2.

Таблиця "Студенти"

Ключ	Поле
+	Номер студентського квитка
	Прізвище ім’я по-батькові
	Адреса
	Дата народження
	Група

Таблиця 3.3.

Таблиця "Оцінки"

Ключ	Поле
+	Номер студентського квитка
+	Предмет
+	Дата іспиту
	Оцінка

то поле “Номер студентського квитка” у таблиці “Оцінки” є зовнішнім ключем для зв’язування з таблицею “Студенти”.

Поле “Група” у таблиці “Студенти” є зовнішнім ключем для зв’язування з таблицею “Групи”, що може мати наступну структуру:

Таблиця 3.4.

Таблиця "Групи"

Ключ	Поле
+	Група
	Спеціальність
	Куратор

Підтримка цілісності в реляційній моделі містить такі складові:

1) структурна цілісність;
2) посилальна цілісність;
3) обмеження на реальні дані.

Поняття структурної цілісності перш за все пов’язане з первинним ключем, а саме з вимогою його обов’язкової наявності в кожній таблиці.

Як вже говорилося вище, це призводить до відсутності дублікатів первинного ключа в таблиці. Наприклад, якщо в таблиці “Студенти” первинним ключем є “Номер студентського квитка”, стає неможливим ввести один і той самий номер квитка для різних студентів.

Посилальна цілісність вже стосується пов’язаних таблиць, тому її ще називають обмеженням зовнішнього ключа. Суть посилальної цілісності полягає в наступному:

1) в підлеглій таблиці немає записів, для яких не існує відповідних записів в основній таблиці;
2) кожен запис в підлеглій таблиці має тільки один батьківський запис в основній таблиці.

Наприклад, посилальна цілісність для таблиць “Студенти” та “Оцінки” полягає в тому, що в таблиці “Оцінки” не може бути записів з таким номером студентського квитка, якого немає в таблиці “Студенти”.

Перед поясненням посилальної цілісності доцільно запропонувати студентам зміст таблиць “Студенти” та “Оцінки” з порушенням посилальної цілісності, наприклад такі:

Таблиця 3.5.

Таблиця «Студенти»

Номер квитка	**ПІБ**	**Адреса**	**Дата народження**	**Група**
1	Іваненко І.І.	вул. Шевченка,13, кв. 1	01.01.1990	11
2	Петренко П.П.	вул. Щорса,14, кв. 12	02.02.1990	11

Таблиця 3.6.

Таблиця «Оцінки»

Номер квитка	**Предмет**	**Дата іспиту**	**Оцінка**
1	Інформатика	14.01.2009	Добре
2	Інформатика	14.01.2009	Задовільно
3	Інформатика	14.01.2009	Добре

і запропонувати їм визначити, чи є помилка в даних, а якщо є, запропонувати вказати її.

Обмеження на реальні дані вимагають, щоб значення належали певному діапазону значень або задовольняли певне арифметичне співвідношення між значеннями констант та інших полів. Наприклад, значення поля “Дата народження” гарантовано не може відноситися до 19-го століття, значення поля “Група” на даному факультеті повинне бути у межах від 11 до 69 тощо.

До підтримки цілісності також можна віднести правила, за якими повинні вилучатися та оновлюватися дані у зв’язаних таблицях.

Для вилучення даних можливі такі варіанти:

1. RESTRICT: заборонено вилучати запис із батьківської таблиці, якщо є його нащадки в підлеглій таблиці:
2. CASCADE: при вилученні запису в батьківській таблиці вилучаються всі його нащадки в підлеглій таблиці;
3. SET NULL: при вилученні запису з батьківської таблиці у всіх записах-нащадках підлеглої таблиці зовнішнім ключам надається значення NULL (невизначене значення);
4. SET DEFAULT: аналогічно до попереднього, але замість невизначеного значення відбувається надання значення, встановленого за замовчуванням.

Для оновлення даних можливі такі варіанти:

1. RESTRICT: не можна змінювати первинний ключ у запису батьківської таблиці, якщо в підлеглій таблиці цей запис має нащадків;
2. CASCADE: при зміні первинного ключа в запису батьківської таблиці автоматично така ж зміна відбувається у всіх записах-нащадках підлеглої таблиці;

3. SET NULL: при зміні первинного ключа в запису батьківської таблиці автоматично у всіх записах-нащадках підлеглої таблиці зовнішній ключ замінюється значенням NULL;
4. SET DEFAULT: при зміні первинного ключа в запису батьківської таблиці автоматично у всіх записах-нащадках підлеглої таблиці зовнішній ключ замінюється значенням за замовчуванням.

Щодо маніпулювання реляційними даними, то оскільки реляційні дані являють собою множини, до них застосовні звичайні теоретико-множинні операції: об'єднання, переріз, віднімання, декартовий добуток.

Кодд розширив цей стандартний набір операцій і запропонував у якості засобу для маніпулювання реляційними базами даних спеціальний набір операцій, які було названо алгеброю Кодда. В алгебрі Кодда десять операцій: об'єднання, перетин, віднімання, декартовий добуток, перейменування, проекція, обмеження, об'єднання за умовою, ділення і надання значення [145].

При об'єднанні двох таблиць однакової структури створюється таблиця, що містить всі записи, які входять хоча б в одну з таблиць.

Результатом перетину двох таблиць з однаковою структурою є таблиця, що містить всі записи, які входять до обох таблиць.

Різницею двох таблиць з однаковою структурою є таблиця, яка містить ті записи першої таблиці, яких немає в другій.

Декартовим добутком двох таблиць, що не мають однакових полів, є таблиця, записи якої утворюються шляхом об'єднання записів першої та другої таблиць.

Операція перейменування призначена для зміни імен полів, це дає змогу виконувати попередні операції над таблицями, у яких не співпадають імена відповідних полів, але співпадають їх типи, а також виконувати операцію декартового добутку над таблицями, що мають поля з однаковими іменами.

Результатом обмеження таблиці за певною умовою є таблиця, що містить тільки ті записи, які задовольняють цю умову.

Результатом проекції таблиці на вказану підмножину її полів є таблиця, що містить тільки ті поля, що вказані у проекції.

При об’єднанні за умовою двох таблиць створюється таблиця, записи якої утворюються шляхом об’єднання записів першої і другої таблиць і задовольняють цю умову.

Операція надання значення призначена зберегти результат обчислення реляційного виразу в існуючій таблиці бази даних.

В реальних реляційних СУБД використання реляційної алгебри для маніпулювання реляційними даними практично не застосовується, а широкого застосування набули мови запитів, що базуються на моделі SQL. Реляційне числення основане на використанні предикатів, що мають задовольнити потрібні кортежі або домени. В мовах запитів, таких як SQL, QBE (Query By Example) реалізуються не тільки функції відповідної теорії, але і використовуються певні додаткові операції, наприклад, арифметичні операції, операції друкування тощо [218].

Реляційна модель і її нащадок модель SQL виявилися досить ефективними для подання даних для великої кількості задач з різних предметних галузей і тому покладені в основу функціонування більшості сучасних СУБД, які відповідно отримали загальну назву реляційних СУБД.

3.1.5. Моделі концептуального подання даних. Оволодіння студентами знаннями про реляційну (SQL) модель ще не є гарантією успішного розв’язування практичних задач, пов’язаних з базами даних, оскільки в таких задачах умова сформульована в термінах предметної галузі, а не реляційної моделі. Для розв’язування цієї проблеми необхідно познайомити студентів з моделями концептуального проектування баз даних. Розрізняють два головних підходи до моделювання даних при концептуальному проектуванні:

1. семантичні моделі;

2. об’єктні моделі [34].

В семантичних моделях головна увага приділяється структурі даних. Найбільш поширеною є модель “сутність – зв’язок” (ER-модель). Моделювання даних в цій моделі відображає логічну структуру даних.

В об’єктних моделях головна увага приділяється "поведінці" об’єктів даних і засобам маніпуляції даними. Головне поняття таких моделей – об’єкт, що характеризується станом і „поведінкою”.

Розглянемо більш детально ER-модель. Вперше поняття ER-моделі запровадив П.Чен. Підхід П.Чена дозволив перевести концептуальне проектування в практичну площину проектування бази даних [34]. В ER-моделі оперують сутностями, атрибутами та зв’язками.

Сутність – це абстракція реального об’єкту предметної галузі. Оскільки сутність відповідає цілому класу однотипних об’єктів предметної галузі, то передбачається, що існує багато екземплярів даної сутності. Атрибутом називають поіменовану характеристику сутності. Зв’язок – це засіб, за допомогою якого подаються відношення між сутностями.

Прикладами сутностей можуть бути сутності ВИКЛАДАЧ, СТУДЕНТ, ОЦІНКА. Атрибутами сутності ВИКЛАДАЧ можуть бути, наприклад, ТАБЕЛЬНИЙ НОМЕР, ПРІЗВИЩЕ, НАУКОВИЙ СТУПІНЬ, КАФЕДРА. Набір атрибутів, що однозначно характеризує кожен конкретний екземпляр сутності, називають ключовим.

Між сутностями встановлюються зв’язки, що відповідають реальним зв’язкам між відповідними об’єктами предметної галузі. Зв’язки бувають бінарні (між двома сутностями); тернарні (між трьома сутностями); N-арні (між N сутностями) та рекурсивні (зв’язок сутності самої з собою).

Найбільш поширеними є бінарні зв’язки, які поділяють на:

1. 1:1 (один до одного);

2. 1:M (один до багатьох);

3. N:M (багато до багатьох).

Наприклад, зв’язок “один до одного” встановлюють між сутностями “куратор” та “група”, оскільки у кожній студентській групі є тільки один куратор.

Зв’язок “один до багатьох” встановлюють між сутностями “студент” та “оцінка”, оскільки кожен студент упродовж навчання отримує багато оцінок.

Зв’язок “багато до багатьох” встановлюють між сутностями “студент” та “викладач”, оскільки один студент навчається у багатьох викладачів, і, в свою чергу, один викладач навчає багатьох студентів.

Взагалі, ER-модель досить складна і у достатньо повному вигляді її доцільно вивчати у курсі “Системи управління базами даних” на спеціальності “Інформатика” педагогічного університету [300].

Наступним етапом розв’язування задачі засобами СУБД є логічне проектування бази даних. Для реляційних СУБД логічне проектування полягає в тому, що на основі ER-моделі створюють реляційну схему, визначають кількість і структуру таблиць, створюють відповідні запити до БД, визначають типи звітних документів і відповідно формують звіти в СУБД, створюють форми для організації інтерфейсу з користувачем, в тому числі для перегляду, введення та редагування даних. Коректність логічної моделі перевіряють і, якщо необхідно, вдосконалюють за допомогою правил нормалізації, що дозволяє впевнитися в логічній цілісності даних у моделі і позбутися надмірності даних. Побудована таким чином концептуальна модель реалізується у конкретній СУБД.

3.1.6. Побудова баз даних в СУБД LibreOffice Base. У Чернігівському педагогічному університеті на кафедрі інформатики в якості навчальної СУБД обрано СУБД LibreOffice Base. Такий вибір зумовлений тим, що дана СУБД є складовою частиною пакету офісних програм LibreOffice, який є кросплатформенним (є реалізації для всіх популярних операційних систем: сімейства ОС Microsoft Windows, сімейства ОС Linux, ОС Mac OS X), а також вільно поширюваним. СУБД LibreOffice Base з цього пакету є

потужною реляційною СУБД, використання якої дає змогу створювати складні бази даних і використовувати мову SQL для формування запитів. В актуальній 3-й версії запити на вибірку можна створювати і за допомогою мови QBE, але для запитів на оновлення, додавання та вилучення даних потрібно обов'язково застосовувати мову SQL, причому реалізовано це таким чином, що мінімізується можливість необдуманого багаторазового запуску таких запитів. Такий спосіб реалізації запитів на модифікацію бази даних більш вдалий, ніж наприклад в СУБД MS ACCESS, де запити на модифікацію можуть запускатися недосвідченими студентами багатократно без будь-яких обмежень, і практика роботи з СУБД MS ACCESS показує, що це часто призводить до спотворення даних.

Для подальшого формування у студентів компетентностей щодо аналізу предметної галузі задачі, побудови інформаційної моделі (в даному випадку спочатку ER-моделі, а на її основі SQL-моделі), та дослідження отриманої моделі студентам пропонується цикл з трьох лабораторних робіт. У першій роботі студенти повинні на основі умови задачі у термінах предметної галузі розробити адекватну модель даних, визначивши сутності предметної галузі, їх властивості та зв'язки між ними; на основі цієї моделі даних створити відповідні таблиці, визначити ключі цих таблиць, встановити зв'язки між таблицями, створити форми для введення і редагування даних з визначенням обмежень на значення у полях. Заповнити таблиці даними.

У другій лабораторній роботі студенти повинні розробити запити до бази даних з попередньої лабораторної роботи. Умови запитів сформульовані у термінах предметної галузі. Запити на вибірку даних та обчислення над даними пропонується виконувати за допомогою мови QBE (у термінах СУБД LibreOffice Base – у дизайнері запитів), а запити на модифікацію даних повинні бути описані мовою SQL.

У третій лабораторній роботі студенти повинні розробити звіти (з використанням обчислювальних полів) до бази даних, розробленої у першій лабораторній роботі.

Завдання у лабораторних роботах містять 12 варіантів, таким чином кожен студент підгрупи виконує власний варіант.

Наведемо етапи розв'язування (побудова моделі даних, бази даних, форм, запитів, звітів) задачі, подібної до тих, що є в умовах лабораторних робіт.

Задача 3.1.

Створити базу даних FIRMA, в якій зберігатимуться дані про співробітників деякої фірми і зроблені їм виплати заробітної плати упродовж деякого періоду (не більше однієї виплати в день).

У базі даних мають зберігатися такі відомості про кожного співробітника:

- табельний номер;
- прізвище, ім'я, по-батькові;
- дата народження;
- домашня адреса;
- посада у фірмі;
- стаж роботи;
- дата отримання заробітної плати;
- отримана сума заробітної плати.

Розв'язування задачі почнемо з побудови ER-моделі даних, тобто моделі "Сутність-Зв'язок". З умови задачі можна виділити наступні сутності предметної галузі: "Співробітник фірми" та "Виплата". Ці сутності пов'язані відношенням "Один до багатьох", оскільки кожен співробітник фірми отримує багато виплат. Тепер визначимо атрибути кожної з сутностей. За умовою задачі сутність "Співробітник фірми" має атрибути "Табельний номер", "Прізвище ім'я по-батькові (ПІБ)", "Дата народження", "Адреса" (домашня адреса), "Посада" та "Стаж", а сутність "Виплата" має атрибути "Дата виплати" (коли зроблена виплата) та "Сума" (розмір виплати). Первинним ключем сутності "Співробітник фірми" є атрибут "Табельний номер", а сутності "Виплата" – "Дата виплати"

(оскільки за умовою задачі не можна робити більше однієї виплати за день).

Далі необхідно перетворити отриману ER-модель в реляційну (SQL) модель. Для кожної сутності створюється відповідна таблиця, полями якої стають атрибути сутності, а первинний ключ сутності стає первинним ключем таблиці. Отримаємо таблиці, які назвемо “Співробітники” та “Виплати”.

Для встановлення зв’язків “один до багатьох” між таблицями потрібно у кожну таблицю, що відповідає підлеглій сутності, додати набір полів, що є первинним ключем таблиці, відповідної основній сутності. Цей набір полів стає зовнішнім ключем. Оскільки сутність “Виплата” є підлеглою до сутності “Співробітник фірми”, у таблицю “Виплати” потрібно додати поле “Табельний номер” із таблиці “Співробітники”.

Таким чином отримано наступні таблиці (Табл.3.7, 3.8) (ключові поля позначено напівжирним шрифтом):

Ці таблиці пов’язані відношенням “один до багатьох” за полем “Табельний номер”.

Таблиця 3.7.

Таблиця «Співробітники»

Співробітники
Табельний номер
ПІБ
Дата народження
Адреса
Посада
Стаж

Таблиця 3.8.

Таблиця «Виплати»

Виплати
Табельний номер
Дата виплати
Сума

Далі потрібно перевірити утворену модель даних за допомогою правил нормалізації. Результатом процесу нормалізації є отримання такої моделі даних, в якій таблиці мають просту і регулярну структуру. В створеній моделі даних можуть бути аномалії – такі ситуації, що призводять до протиріч у базі даних, або ускладнюють опрацювання даних. Нормалізація дозволяє позбутися таких аномалій. Одним з видів аномалій є аномалія модифікації. Вона виникає, коли зміну значень якогось поля потрібно проводити у багатьох записах. Це може призвести до порушення цілісності бази даних (у разі неправильного введення нового значення поля в одному з записів). У розробленій моделі даних така аномалія виникає, коли потрібно змінити назву посади, оскільки таку зміну потрібно виконати для всіх співробітників, які обіймають цю посаду. Для ліквідації цієї аномалії створимо нову таблицю “Штатний розпис”, яка міститиме поля “Номер посади” та “Назва посади”. Тоді поле “Посада” у таблиці “Співробітники” відповідатиме полю “Номер посади” у таблиці “Штатний розпис” і повинно мати такий самий тип даних.

Таблиця 3.9.

Таблиця «Штатний розпис»

Штатний розпис
Номер посади
Назва посади

Для формування у студентів компетентностей щодо розробки запитів в конкретній СУБД студентам необхідно запропонувати такі завдання, які б максимально охоплювали всі види запитів: запити на вибірку даних (в тому числі з використанням параметрів), запити на обчислення, в тому числі і групові, запити на модифікацію даних (зміну існуючих даних, додавання нових, вилучення існуючих). Важливо, щоб студенти мали компетентності щодо створення запитів як мовою QBE, так і мовою SQL.

Наведемо кілька прикладів створення запитів до бази даних, про яку йшла мова у задачі 3.1, в СУБД LibreOffice Base.

Запити на вибірку даних і виконання обчислень

Для задання умов щодо даних числових, грошових типів і типу дата/час застосовуються оператори порівняння >, <, >=, <=, = (= не пишеться).

Для задання кон'юнкції ("і") застосовується оператор *And*, для заперечення ("не") – *Not*, для диз'юнкції ("або") – *Or*, складові диз'юнкції можуть також записуватись в різних рядках бланку запиту.

Константи типу дата/час обмежуються позначками #, наприклад #10/10/02#

Для задання умов щодо текстових полів можна застосовувати символи шаблонів * і ?, які мають загальноприйняте значення (відповідно довільна кількість довільних символів і довільний єдиний символ). Якщо потрібний повний збіг з умовою відбору, відповідний текст записується без символів шаблонів. При вказуванні шаблону слід записати оператор *Like* (подібний), а потім шаблон, взятий в одинарні лапки. Наприклад, умова *Like* '*бухгалтер' є критерієм відбору таких рядків, як, бухгалтер, головний бухгалтер, старший бухгалтер тощо.

Логічна константа EMPTY означає відсутність значення у полі.

В СУБД LibreOffice Base передбачено можливість формувати умови відбору даних в процесі виконання запиту. Така властивість запитів реалізується завдяки використанню параметрів. Щоб створити запит з параметром, слід у бланку запиту в рядку *Критерій* вказати не константу певного типу, з якою будуть порівнюватись дані, а ім’я змінної, перед яким поставити двокрапку. При виконанні такого запиту на екран виводиться вікно з назвою *Введення параметра*, в якому є поле для введення значення параметра. Після введення значення можна побачити відповідь на запит.

Можна вказати у бланку запиту кілька параметрів у різних полях. Тоді при виконанні запиту слід послідовно вказати значення всіх параметрів.

Для створення обчислюваного поля слід у бланку запиту в рядку *Поле* ввести вираз, за яким обчислюються значення у створюваному полі. Він може містити знаки математичних операцій, дужки, стандартні функції LibreOffice Base, імена полів таблиці, до якої створюється запит, взяті в подвійні лапки. Ім'я нового поля слід ввести в рядок бланку запиту *Псевдонім*.

Для обчислення підсумкових значень необхідно включити у бланк запиту рядок *Функція* (для цього слід натиснути кнопку *Функції* на панелі інструментів *Дизайн* вікна програми).

В СУБД LibreOffice Base передбачено виконання таких групових операцій над даними:

Group – операція призначена для групування даних у тому полі, в якому вона встановлена (відповідні записи виводяться підряд);

Сума (Sum) – знаходження суми значень у відповідному полі;

Середнє значення (Average) – знаходження середнього арифметичного (в Конструкторі запитів для виконання цієї операції слід ввести відповідний вираз;

Мінімум (Min) – знаходження мінімального значення у полі;

Максимум (Max) – знаходження максимального значення у полі;

Кількість (Count) – знаходження кількості записів.

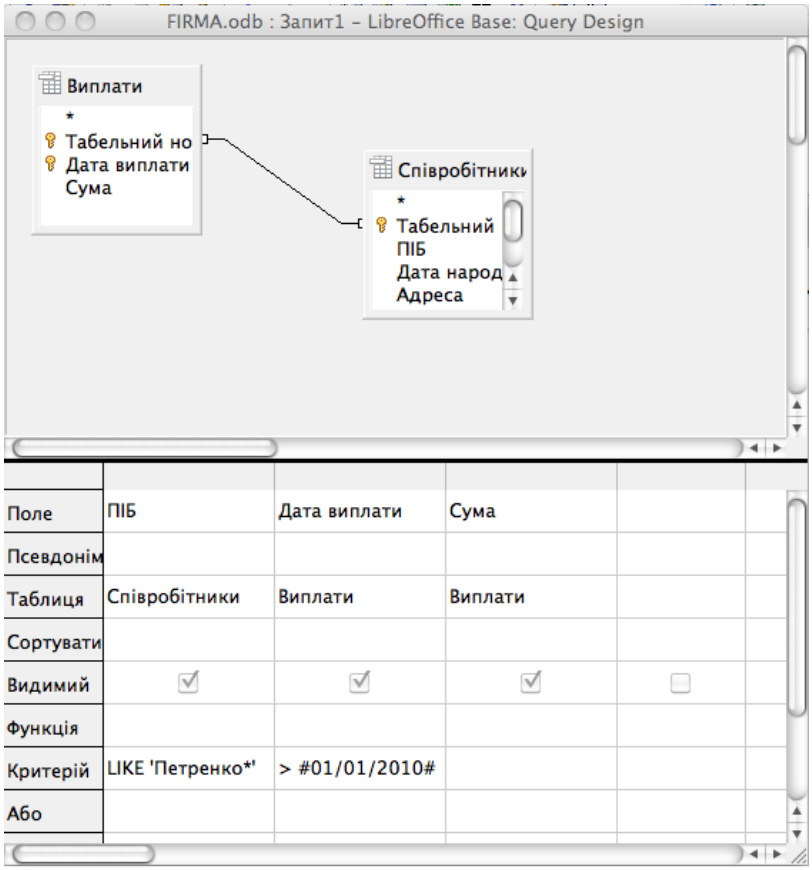

Рис. 3.3. Запит до Прикладу 1.

Приклад 1. За допомогою мови QBE створити запит для формування даних (ПІБ, Дата виплати, Сума) про розміри всіх виплат співробітнику Петренку, зроблених з початку поточного (2010) року.

Запит мовою SQL матиме наступний вигляд:

SELECT "Співробітники"."ПІБ", "Виплати"."Дата виплати", "Виплати"."Сума" FROM "Виплати", "Співробітники" WHERE "Виплати"."Табельний номер" = "Співробітники"."Табельний номер" AND "Співробітники"."ПІБ" LIKE 'Петренко%' AND "Виплати"."Дата виплати" > {D '2010-01-01' }

Запит мовою QBE матиме вигляд, як на Рис.3.3.

Приклад 2. Використовуючи дані таблиць “Співробітники” і “Виплати”, обчислити суми, відраховані у фонд соціального страхування (1%) з виплат усім співробітникам за 2009 рік. Вивести у запиті такі відомості: ПІБ, Сума, Податок.

Запит мовою SQL матиме наступний вигляд:

SELECT "Співробітники"."ПІБ", "Виплати"."Дата виплати", "Виплати"."Сума", "Сума" * 0.001 AS "Податок" FROM "Виплати", "Співробітники" WHERE "Виплати"."Табельний номер" = "Співробітники"."Табельний номер" AND "Виплати"."Дата виплати" >= {D '2009-01-01' } AND "Виплати"."Дата виплати" <= {D '2009-12-31' }

Запит мовою QBE матиме наступний вигляд:

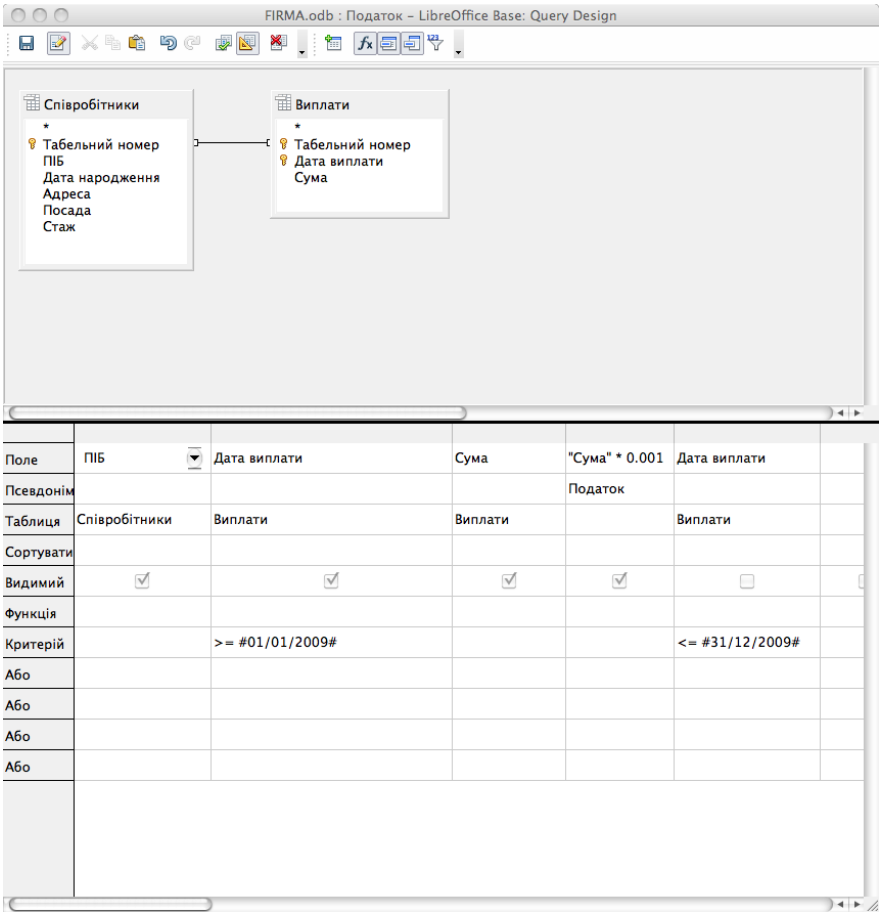

Рис. 3.4. Запит до Прикладу 2.

Приклад 3. Використовуючи дані таблиць “Співробітники” і “Виплати”, обчислити кількість виплат, зроблених кожному співробітнику упродовж другого півріччя 2009 року. У відповіді вивести поля ПІБ, Кількість виплат у 2 півр. 2009 р.

Запит мовою SQL матиме наступний вигляд:

SELECT "Співробітники"."ПІБ", COUNT("Виплати"."Сума") AS "Кількість виплат у 2 півр. 2009 р." FROM "Виплати",

"Співробітники" WHERE "Виплати"."Табельний номер" = "Співробітники"."Табельний номер" AND "Виплати"."Дата виплати" >= {D '2009-07-01' } AND "Виплати"."Дата виплати" <= {D '2009-12-31' } GROUP BY "Співробітники"."ПІБ"

Запит мовою QBE матиме наступний вигляд:

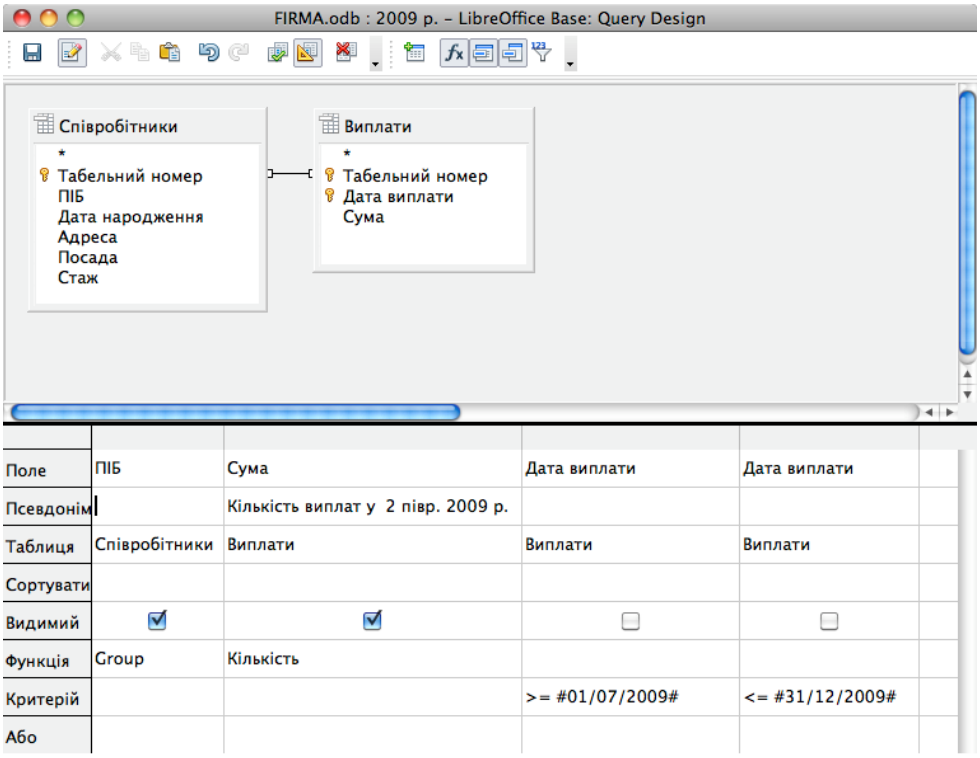

Рис. 3.5. Запит до Прикладу 4.

Запити на модифікацію даних

В СУБД LibreOffice Base передбачено можливість створювати запити, результатами яких є не нові таблиці, а зміни, внесені у вихідні таблиці бази даних. Такі запити створюються за допомогою засобів мови SQL. Після запуску запиту на виконання в LibreOffice Base не виводиться попередження про можливі зміни даних. Виконання запиту призводить до внесення незворотних змін у таблицю. Тому виконувати такі запити слід помірковано і зважено. Після виконання запитів на модифікацію їх результат переглядають, відкривши відповідну таблицю на закладці *Таблиці* вікна бази даних.

При створенні запитів на зміну таблиць слід перейти на закладку *Таблиці* вікна бази даних і вибрати команду меню *Сервіс – SQL*. У допоміжному вікні *Виконати інструкцію SQL* в поле *Виконувана команда* слід ввести текст запиту і натиснути кнопку *Виконати*. Повідомлення про результат виконання команди (успішне виконання або знайдені помилки) виводиться в нижній частині вікна в полі *Стан*. Після завершення роботи з запитом слід натиснути кнопку *Закрити*. Текст SQL-запитів у LibreOffice Base не зберігається. Тому для повторного запуску або збереження запиту в певному звіті текст запиту слід скопіювати до буфера обміну для подальшого використання.

Використовуючи запити на оновлення, можна змінювати значення окремих полів вихідної таблиці. Для полів, що підлягають зміні, можуть бути вказані певні умови.

Найпростіша структура запиту на оновлення:

UPDATE "Ім'я таблиці"
SET "Ім'я поля"="нове значення"
WHERE <умова>

<умова> – це вираз, в якому можуть використовуватись імена полів, константи, знаки математичних операцій і відношень, логічні оператори.

Текстові константи беруться в одинарні лапки, наприклад: *'текст'*

Константи типу „Дата" записуються у вигляді *{d 'рррр-мм-дд'},* наприклад:

{d '2010-12-01'} означає 1 грудня 2010 року.

За допомогою запитів на вилучення даних користувач може вилучати з таблиці записи, у яких значення певних полів відповідають вказаним умовам.

Найпростіша структура запиту на вилучення:

DELETE FROM “Ім’я таблиці”

WHERE <умова>

Запити на додавання дозволяють поповнювати таблицю даними з іншої таблиці або вказаними безпосередньо у тексті запиту.

Найпростіша структура запиту на додавання:

INSERT INTO “Ім’я таблиці”

VALUES(список значень полів)

Приклад 4. Замінити у таблиці “Виплати” всі входження дати 1 травня 2010 року на 11 травня 2010 року.

Виконання. У допоміжному вікні *Виконати інструкцію SQL* в поле *Виконувана команда* ввести такий текст запиту:

```
update "Виплати"
set "Дата виплати"={d '2010-05-10'}
where "Дата виплати"={d '2010-05-01’}
```

Приклад 5. Вилучити з таблиці “Виплати” відомості про виплати, зроблені директору.

Виконання. У допоміжному вікні *Виконати інструкцію SQL* в поле *Виконувана команда* ввести такий текст запиту:

```
delete from "Виплати"
        where "Табельний номер" in
        (select "Табельний номер" from "Співробітники"
        where "Посада" in
        (select "Номер посади" from "Штатний розпис"
        where "Назва посади"='Директор'))
```

Приклад 6. Додати до таблиці “Співробітники” всі анкетні відомості про нового співробітника: Табельний номер – 7777, ПІБ – Ткач О.О., Дата народження – 10.10.1980, Адреса – Проспект Перемоги, 33/22, Посада – 3, Стаж – 10.

Виконання. У допоміжному вікні *Виконати інструкцію SQL* в поле *Виконувана команда* ввести такий текст запиту:

```
insert into "Співробітники"
        values(7777,'Ткач О.О.',
        {d '1980-10-10'},'Просп Перемоги, 33/22',3,10)
```

Приклад 7. Збільшити на один рік стаж всіх співробітників, які працюють на посаді інженера і чий стаж більше одного року.

Виконання. У допоміжному вікні *Виконати інструкцію SQL* в поле *Виконувана команда* ввести такий текст запиту:

```
update "Співробітники"
        set "Стаж"="Стаж"+1
        where "Стаж">1 and "Посада" in
        (select "Номер посади" from "Штатний розпис"
        where "Назва посади"='Інженер')
```

Приклад 8. Подвоїти всі виплати директору фірми.

Виконання. У допоміжному вікні *Виконати інструкцію SQL* в поле *Виконувана команда* ввести такий текст запиту:

```
update "Виплати"
        set "Сума"="Сума"*2
        where "Табельний номер" in
```

(select "Табельний номер" from "Співробітники"

where "Посада" in

(select "Номер посади" from "Штатний розпис"

where "Назва посади"='Директор'))

Приклад 9. Вилучити з таблиці “Співробітники” всі відомості про інженерів, стаж роботи яких більше 20 років.

Виконання. У допоміжному вікні *Виконати інструкцію SQL* в поле *Виконувана команда* ввести такий текст запиту:

delete from "Співробітники"

where "Стаж">20 and "Посада" in

(select "Номер посади" from "Штатний розпис"

where "Посада" in

(select "Номер посади" from "Штатний розпис"

where "Назва посади"='Інженер'))

Приклад 10. Вилучити відомості про виплати бухгалтерам, стаж яких більше 10 років.

Виконання. У допоміжному вікні *Виконати інструкцію SQL* в поле *Виконувана команда* ввести такий текст запиту:

delete from "Виплати"

where "Табельний номер" in

(select "Табельний номер"

from "Співробітники"

where "Стаж">10 and

“Посада” in

(select "Номер посади" from "Штатний розпис"

where "Назва посади"='Бухгалтер'))

Приклад 11. Додати до таблиці “Співробітники” окремі анкетні відомості (Табельний номер, ПІБ, Посада) про нового співробітника.

Виконання. У допоміжному вікні *Виконати інструкцію SQL* в поле *Виконувана команда* ввести такий текст запиту:

insert into "Співробітники" ("Код","ПІБ","Посада")
values(8888,'Шевченко А.А.', 1)

3.2. Інформаційне моделювання і штучний інтелект

3.2.1. Загальні питання основ штучного інтелекту. Проблема навчання студентів інформаційного моделювання повинна вирішуватися комплексно при вивченні всіх дисциплін інформатичного напрямку, в тому числі і при вивченні дисципліни “Основи штучного інтелекту”, яка вивчається на спеціальності “Інформатика” у вищих педагогічних навчальних закладах. В даній дисципліні потрібно приділити якнайбільшу увагу таким темам, як моделі штучного інтелекту, які є саме інформаційними моделями.

Проблема створення штучного інтелекту цікавила дослідників вже досить давно. Спроби створити машини з розумною поведінкою багато в чому базуються на ідеях Норберта Вінера, творця кібернетики. Саме Норберт Вінер запропонував використовувати у розробці “розумних” машин ідею зворотного зв’язку. Ця ідея полягає у використанні сигналів, що надходять із зовнішнього світу, для зміни поведінки машини. Н.Вінер писав, що всі машини, які претендують на “розумність”, повинні “володіти можливістю переслідувати певні цілі і пристосовуватись, тобто навчатися” [26].

Ідеї Н.Вінера і його послідовників – Уоррена Маккалоха та Уотера Піттса були покладені в основу так званої “висхідної” моделі штучного

інтелекту – руху від простих аналогів нервової системи примітивних істот, що мають малу кількість нейронів, до найскладнішої нервової системи людини. Кінцева мета вбачалася в створенні адаптивної мережі, системи, що самоорганізується, або машини, що навчається [136]. Певних досягнень у цьому напрямку вдалось досягнути Френку Розенблату, який створив пристрій під назвою “Перцептрон” і його вдосконалений варіант “Марк-1”. “Марк-1” “зміг навчитися” розпізнавати деякі з літер, написаних на картках. Ці пристрої були найвищим досягненням “висхідної” моделі штучного інтелекту. Інші розробки в галузі штучного інтелекту пов’язані з “нисхідною” його моделлю. Ця модель пов’язана з розробкою комп’ютерних програм, за якими моделюється діяльність людини при розгляді задач, розв’язування яких потребує людського інтелекту, наприклад, для доведення теорем, розпізнавання образів тощо. Можна навести більш детальний список задач, пов’язаних із дослідженнями штучного інтелекту, що є актуальними сьогодні:

1. Доведення теорем.
2. Розпізнавання образів, наприклад, програми оптичного розпізнавання символів, програми розпізнавання голосу, розпізнавання жестів.
3. Комп’ютерні ігри, наприклад, шахові програми, що вже грають на рівні гросмейстерів.
4. Прийняття рішень.
5. Адаптивне програмування.
6. Складання музики, віршів.
7. Опрацювання даних, поданих природною мовою, наприклад, програми-перекладачі.
8. Нейронні мережі.
9. Символьні обчислення.

3.2.2. Експертні системи та моделі подання знань. В 60-ті роки увага дослідників питань штучного інтелекту сконцентрувалася на такому напрямі, як експертні системи. Експертна система є програмою, за

допомогою якої імітують діяльність людини-експерта у певній досить вузькій прикладній галузі. Цей напрям досліджень був вибраний внаслідок того, що виникли великі проблеми у створенні “думаючих” програм так би мовити загального призначення. Щодо експертних систем, один з їх розробників, Едвард Фейгенбаум, зауважив: “Ми з’ясували, що краще бути багатознаючим, ніж розумним” [136].

Експертним системам (ЕС) притаманні наступні ознаки [86]:

1. На основі ЕС за допомогою комп’ютера відтворюють деякі методики розв’язування проблем, що використовуються експертами.
2. За допомогою ЕС формуються певні висновки, базуючись на тих знаннях (фактах і правилах), на яких вона основана. Знання в ЕС зберігаються окремо від програмного коду. Компонент зберігання знань прийнято називати базою знань.
3. Під час розв’язування задач за допомогою ЕС основну роль відіграють евристичні та наближені методи, використання яких не завжди гарантує успіх.
4. ЕС застосовують в таких предметних галузях, робота в яких вимагає великого досвіду, накопиченого людиною-експертом. Вони мають яскраво виражену практичну спрямованість для застосування в науковій або комерційній сферах.
5. Використання ЕС повинно забезпечувати відшукання за прийнятний час розв’язку, що був би не гіршим за розв’язок, який може запропонувати експерт в цій предметній галузі. Вартість використання ЕС не повинна бути вищою за вартість роботи експертів.

В ЕС повинна бути передбачена можливість надати пояснення, чому запропоновано саме цей розв’язок, і доведення його обґрунтованості. Користувач повинен отримати всі відомості, необхідні для того, аби переконатись в обґрунтованості запропонованого розв'язку.

Експертна система складається з кількох частин, а саме:

1. база знань;

2. механізм виведення;

3. механізм пояснень;

4. інтерфейс користувача.

База знань містить знання (факти та правила) з певної предметної галузі, подані за однією з моделей знань. На основі правил за допомогою механізму виведення із вже наявних фактів отримують нові факти. Саме наявність правил відрізняє базу знань від бази даних, яка містить самі лише факти.

За допомогою механізму виведення знання зв'язуються в базі знань в єдине ціле, а потім із послідовності знань виводяться відповідні висновки.

Загальні поняття щодо експертних систем досить детально розглянуто в [216]. В цій роботі досліджено напрямки використання експертних систем у навчальному процесі та запропоновано досить детальну методику вивчення експертних систем, дібрано систему вправ для оволодіння компетентностями щодо експертних систем. Проте питання щодо моделей подання знань у експертних системах розглянуто не досить детально, хоч ці питання є досить важливими. Тому розглянемо тут моделі подання знань в експертних системах дещо детальніше. В системах з базами знань, в тому числі і експертних системах, подання знань є фундаментальним поняттям, а рішення про вибір моделі подання знань суттєво впливає на довільну їх складову. Можна навіть сказати, що моделлю подання знань визначаються можливості використання бази знань [192].

3.2.3. Продукційні правила. Першою моделлю подання знань розглянемо модель *продукційних правил*. В продукційних правилах описують знання у формі ЯКЩО-ТО. В загальному вигляді правило записується так "ЯКЩО A_1, A_2, ..., A_n, ТО В". Це означає, що якщо твердження від A_1 до A_n істинні, то твердження В також істинне або ж потрібно виконати дію В. Як приклад можна навести таке правило:

ЯКЩО Х набрав потрібну кількість балів

Х має потрібну медичну довідку

ТО X поступив до вузу

В даному правилі дві умови (*n=2*). Якщо умов взагалі немає, отримується факт, наприклад “Іваненко І.І. – чоловік”.

Розглянемо правило

ЯКЩО X старший за Y

Y старший за Z

ТО X старший за Z

та факти

Іван старший за Петра

Іван старший за Григорія

Петро старший за Степана

Григорій старший за Семена

Застосувавши дане правило до наведених фактів, можна отримати два нових факти:

Іван старший за Степана

Іван старшийза Семена

В механізмах виведення в системі, що базується на моделі продукційних правил, можуть для отримання висновків використовуватися як прямі, так і зворотні виведення. В прямих виведеннях застосовують правила до існуючих фактів в надії досягти потрібну ціль. У зворотних виведеннях йдуть від потрібної цілі в надії знайти факти для її обґрунтування. В деяких ЕС модель продукційних правил розширена можливістю використовувати в фактах і правилах нечіткі відомості, яку називають фактором впевненості.

Мабуть першою ЕС, що основувалася на моделі продукційних правил, була ЕС DENDRAL, розроблена Ледербергом та Фейгенбаумом у

Стенфордському університеті. Ця ЕС призначалась для оцінювання структури невідомих хімічних з'єднань, виходячи з експериментальних результатів мас-спектроскопії, хімічної теорії та емпіричних знань хіміків.

Іншою відомою ЕС, що базується на моделі продукційних правил, є система MYCIN, створена Шортліфом за підтримки Фейгенбаума та Букхенена. Використання цієї системи дозволяло діагностувати менінгіти та інфекції крові, а також вибирати антибіотики для їх лікування. В цій ЕС було застосовано поняття фактору впевненості.

На основі ЕС MYCIN була створена перша оболонка експертних систем під назвою EMYCIN (Empty MYCIN, тобто порожній MYCIN). Механізм виведення тут відокремлений від бази знань і існує можливість наповнити базу знань даними з іншої предметної галузі, описавши їх у формі, прийнятій в MYCIN, створивши таким чином нову експертну систему.

3.2.4. Семантичні мережі. Потреби в автоматичному перекладі текстів з однієї мови на іншу привели до появи програм, що засновані на синтаксисі, наприклад програм-перекладачів. Як правило такі програми включали великий словник (словники), набір знань про елементи мови та аналізатор синтаксису. Але досить швидко дослідники розчарувались в таких програмах, оскільки виникали істотні проблеми з неоднозначностями, якими багаті природні мови. Наприклад фраза "Hi-End computer" перекладалася як "Привіт-кінець комп'ютер". Тому увага дослідників була спрямована на семантику.

Семантика в широкому сенсі слова – аналіз відношень між мовними виразами і світом, реальним і уявним, а також самі ці відношення і сукупність таких відношень. Відношення між мовними виразами і світом полягають в тому, що мовні вирази (слова, словосполучення, речення, тексти) означають те, що існує у світі – предмети, якості (або властивості), дії, способи здійснення дій, відношення, ситуації і їх послідовності. Термін "семантика" утворений від грецького кореня, пов'язаного з ідеєю

“позначення” (semantikos – “той, що позначає”). Відношення між виразами природної мови і дійсним або уявним світом досліджує лінгвістична семантика, що є розділом лінгвістики. Семантикою називається також один з розділів формальної логіки, де описуються відношення між виразами штучних формальних мов і їх інтерпретацією в певній моделі світу [189].

В 1968 р. була створена програма “SHRDLU” (ця послідовність літер є відомим типографським позначенням), в якій вперше поєднувалися можливості аналізу синтаксису, семантики, “здатності до міркувань”. За програмою була сформована база знань про певну вузьку предметну галузь – іграшковий світ, що складався з коробки, наповненої різнокольоровими кубами і пірамідами, поверхні столу, а також руки робота, що могла ставити предмети один на один або у коробку. Все це моделювалося в пам’яті комп’ютера і зображувалося на екрані у вигляді тривимірної картинки. Програма реагувала на команди англійською мовою, що вводились з клавіатури, виконуючи відповідні операції у змодельованому в ній віртуальному світі.

Також в кінці 60-х років вчений М. Росс Куілліан запропонував моделювати пам’ять людини у вигляді величезної павутини, яку він назвав семантичною мережею. У вузлах такої мережі подають сутності і поняття, а дуги є описами відношень між цими сутностями або поняттями, тобто семантична мережа подається графом. В комп’ютерній пам’яті вузли можуть відповідати певним записам, а зв’язки – це вказівники.

Семантичні мережі бувають однорідними та неоднорідними. Однорідні мережі мають тільки один тип відношень. У неоднорідних мережах кількість типів відношень більше одного. Класичними ілюстраціями даної моделі подання знань є саме такі мережі. Неоднорідні мережі є більш придатними для практичних цілей, але і складнішими для досліджень [225].

За арністю типовими є мережі з бінарними відношеннями (що зв'язують рівно два поняття). Бінарні відношення дійсно є простими і зручно виглядають на графі у вигляді стрілки між двома поняттями. Крім того вони відіграють виняткову роль в математиці. На практиці, проте, можуть знадобитися відношення, що зв'язують більше двох об'єктів, – N-арні. При цьому виникає складність – як відобразити такий зв'язок на графі, щоб не заплутатися [225].

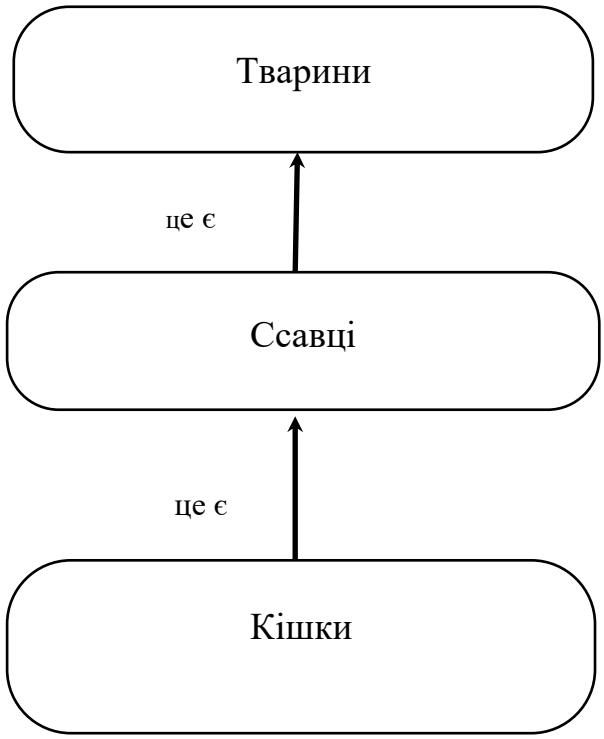

Рис. 3.6. Приклад семантичної мережі.

Найбільш важливими в семантичних мережах вважають зв’язки виду “це є” (“IS A”), за допомогою яких можна зв’язати поняття в ієрархію, в якій вузли нижчих рівнів наслідують властивості вищих. Але існують і інші види зв’язків, наприклад зв’язок “різновид” (“A KIND OF”), зв’язок “є частиною” (“HAS PART”) та інші.

Виведення в семантичних мережах визначаються через відношення між множиною дуг, що мають спільні вузли. Так, наприклад, за зображеною нижче семантичною мережею (Рис. 3.6) можна з’ясувати, що кішки – це ссавці, а відповідно наслідують всі властивості ссавців. Також можна зробити і висновок, що кішки є тваринами.

Останнім часом певної популярності набула нова концепція розвитку всесвітньої павутини під назвою “семантична павутина”.

Семантична павутина – це надбудова над існуючою Всесвітньою павутиною, яка покликана зробити повідомлення, що розміщені в мережі, більш “зрозумілими для комп'ютерів”. Відомо, що майже всі повідомлення в Інтернеті подані в текстовій формі. Не секрет також, що прогрес в галузі опрацювання людської мови (Natural Language Processing, NLP) йде дуже повільно. Тому основна ідея семантичної павутини – кожен ресурс, описаний природною мовою доповнити, описом, який легко може бути використаний у відповідних комп’ютерних програмах для оцінювання, класифікації, актуалізації таких ресурсів [226].

3.2.5. Модель дошки оголошень. Вже досить давно дослідників цікавили проблеми розпізнавання голосу. Так телефонна фірма “Белл” висунула ідею позбавити людей необхідності набирати номер телефону, проговорюючи його в телефонну слухавку. Питаннями розпізнавання голосу цікавилися і в міністерстві оборони США, де було запропоновано фінансування таких розробок. В результаті була розроблена програма “HEARSEY-II”. Ця програма складалася з кількох підпрограм, за допомогою кожної з яких розв’язували певну частинну проблему загальної задачі, наприклад за однією підпрограмою знаходили сигнали на голосових детекторах, за другою аналізували фонетику і фонеміку, тобто подання звуків мови і правила опису компонентів мовлення даної мови. За третьою підпрограмою аналізували правила інтерпретації наголосів і інтонацій. За синтаксичною підпрограмою аналізувалися норми правильної побудови фраз, а за семантичною – значення слів. За кожною з підпрограм фіксувалися результати їх роботи у спеціально виділеній частині пам’яті системи – “дошці оголошень”. За іншими підпрограмами, звернувшись до цієї дошки, можна вибрати найбільш правдоподібні з цих результатів. Наприклад, якщо за фонетичною підпрограмою було зафіксовано, що проаналізовані звукові дані – певна комбінація фонем, то

за синтаксичною підпрограмою оцінювалося, чи можуть з'явитися відповідні фонемам слова в даному місці речення [136].

Таким чином модель дошки оголошень застосовується у випадку, коли потрібно розв'язати проблему, що складається з кількох окремих підпроблем. Кожній такій підпроблемі відповідає специфічне джерело знань, причому всі джерела знань через спільну ділянку пам'яті, що називається дошкою оголошень, управляються так, що всі знання використовуються скоординовано, як єдине ціле. Кожне джерело знань саме по собі будується як продукційна система [192].

Джерела знань в моделі дошки оголошень об'єднуються в ієрархічну систему, відповідно виведення в такій моделі відбувається послідовно від нижчих рівнів ієрархії до вищих. На кожному рівні виведення висувається гіпотеза, яка записується на дошку оголошень. На більш високому рівні ця гіпотеза перевіряється, і якщо вона підтверджується, то її аналіз піднімається ще вище за ієрархією. Якщо гіпотеза не підтверджується, то здійснюється вихід на нижчий рівень для висування нової гіпотези.

3.2.6. Фреймова модель. В 1974 р. Марвін Мінський висловив припущення, що людський розум інтерпретує кожен новий об'єкт, зокрема мовний, за допомогою особливих структур пам'яті, які він назвав фреймами. На розробку фреймової моделі здійснили вплив результати досліджень, за якими було зроблено висновки, що люди зберігають і використовують свій досвід у вигляді загальних концепцій, які при необхідності конкретизуються. Фрейм – комплексний пакет знань, що зберігається в пам'яті комп'ютера і в якому описується певний об'єкт або поняття [136].

Фрейм – загальний каркас знань, містить іменовані відділення, які називаються слотами. Слоти містять дані про характеристики того об'єкта, який описується в фреймі. Слід зауважити, що в слотах можуть міститися прості дані, наприклад вік, стать, зріст, а також і процедури, за якими можна знаходити значення слоту, наприклад значення слоту "розмір

одягу” може обчислюватися за певною формулою на основі даних із таких слотів, як “зріст”, “маса”, “стать” і інші. Нарешті слот може містити інший фрейм, і також фрейм може бути складовою більш крупного фрейму. Таким чином можна утворити певну мережу (ієрархію) пов’язаних між собою фреймів. Від слотів, що подаються у вигляді процедур, може передаватися управління один одному шляхом надсилання повідомлень. Існують поняття фрейм-клас, що містить загальні характеристики цілого класу об’єктів, наприклад фрейм “людина” і фрейм – представник цього класу, наприклад фрейм “Іваненко Іван Іванович”, який наслідує характеристики фрейму “людина”. Таким чином для фреймів існує поняття наслідування. Фреймова система дуже близька до концепції об’єктно-орієнтованого програмування. Оскільки студенти вивчають об’єктно-орієнтоване програмування раніше, ніж основи штучного інтелекту, така аналогія дозволить їм краще з’ясувати фреймові моделі.

3.2.7. Штучні нейронні мережі. Останнім часом можна спостерігати зростання інтересу до так званих штучних нейронних мереж.

За [314] штучні нейронні мережі (ШНМ) — математичні моделі, а також їх програмна та апаратна реалізація, побудовані за принципом функціонування біологічних нейронних мереж — мереж нервових клітин живого організму. Архітектура і принцип дії штучних нейронних мереж базується на аналогії з мозком живих істот.

Штучні нейронні мережі виникли в рамках досліджень з питань штучного інтелекту. Історія ШНМ починається з 1943 року, коли У.Маккалок та У.Піттс запропонували модель біологічного нейрона, сформулювавши основні положення щодо роботи людського мозку.

Загальновідомо, що людський мозок має значні переваги у порівнянні з комп'ютером у багатьох задачах, наприклад у задачах розпізнавання образів.

Виділяють кілька принципових відмінностей людського мозку щодо опрацювання даних у порівнянні зі звичайним комп'ютером [182]:

1. Здатність до навчання на прикладах.
2. Здатність до узагальнень. На прикладах конкретних екземплярів якогось об'єкта, наприклад фото різних літаків, у людському мозку формується узагальнений образ цього об'єкта, тобто літака. Далі спостерігаючи будь-який об'єкт, людина може визначити, літак це, чи ні.
3. Паралельність опрацювання даних (наприклад зображення людиною аналізується не за пікселями, а відразу повністю).
4. Висока надійність мозку. З віком люди втрачають значну частину нервових клітин, проте залишаються у здоровому глузді і міцній пам'яті.
5. Асоціативність людської пам'яті, тобто здатність знаходити потрібні дані за малою їх частиною.

Всі ці переваги спонукали до досліджень, які б дозволяли з'ясувати, як працює людський мозок, і спроб змоделювати таку роботу.

Було з'ясовано, що мозок складається з нейронів. В тіло біологічного нейрона вхідні сигнали поступають через синапси у вигляді електричних імпульсів, які там накопичуються. Коли рівень електричного заряду у біологічному нейроні досягне певної величини, цей заряд передається у вихід біологічного нейрона, який називають аксоном.

У.Маккалок та У.Піттс запропонували модель біологічного нейрона, як перемикача, що отримує дані від інших нейронів. Отримані дані помножуються на значення, співставлені з синапсами, і підсумовуються. Якщо отримане значення перевищує задану межу, нейрон активізується, в протилежному випадку він залишається неактивним. У 60-х роках було доведено, що за такою моделлю можна виконувати складні операції розпізнавання образів.

Сформулюємо більш строго поняття штучного нейрона.

Штучним нейроном називається математичний об'єкт, що має n входів і 1 вихід, через які передаються числові дані.

Входи пов'язують штучний нейрон з іншими штучними нейронами або зовнішнім джерелом даних і називаються зв'язками. Вхідні дані позначатимо x_i. З кожним зв'язком пов'язане число, що називається ваговим коефіцієнтом зв'язка. Позначимо ці числа w_i.

Опрацювання даних відбувається за два етапи.

На першому етапі обчислюється зважена сума $S=w_0x_0+...+w_nx_n$.

На другому етапі обчислюється значення y як деяка функція від зваженої суми:

$y=f(S)$. y називають функцією активації.

Функції активації бувають різних видів. Історично першою була ступіньчата функція $f(S)$, за якою повертається нуль, коли $S<0$, і 1 у протилежному випадку. Тобто за аналогією з біологічним нейроном – коли загальний імпульс перевищить критичне значення, генерується імпульс зі значенням 1, в протилежному випадку нейрон залишається у спокої.

Існують і інші функції активації, наприклад логістична функція (сигмоїд):

$$f(S)=\frac{1}{1+e^{-\alpha S}}$$

При зменшенні α графік сигмоїда стає більш пологим, при збільшенні α графік сигмоїда за зовнішнім виглядом стає схожим на графік ступіньчатої функції.

З таких штучних нейронів складають штучну нейронну мережу. Штучні нейронні мережі можуть мати різні структури. Однією з найбільш поширених структур є багатошарова. Шар – це сукупність штучних нейронів, які не з'єднані між собою, але з'єднані з штучними нейронами попереднього та наступного шарів. Зовнішні вхідні сигнали подають на входи нульового шару мережі, а виходами мережі є вихідні сигнали останнього шару. Багатошарові ШНМ називають перцептронами.

Серед всіх багатошарових ШНМ виділяють:

1.Мережі прямого розповсюдження. В цих мережах передають сигнали тільки від попереднього шару до наступного. Як правило, кожен вихідний сигнал *n*-го шару передається на вхід всіх нейронів *(n+1)* шару.

2.Мережі із зворотними зв'язками. В таких мережах дані також передаються з наступних шарів до попередніх.

Найпростішою ШНМ є мережа, що складається з одного штучного нейрона з n входами, одним виходом і пороговою функцією активації (елементарний перцептрон). На входи подаються значення x_i. Для кожного з входів визначено ваговий коефіцієнт w_i, а також визначено поріг. На виході такої ШНМ буде 1, коли сума $S=w_0x_0+...+w_nx_n$ перевищить поріг і -1 в протилежному випадку. Таку мережу можна використовувати для розв'язування задачі класифікації двох класів: якщо на виході 1, то поданий на вхід вектор належить до першого класу, в протилежному випадку – до другого.

Збільшуючи кількість нейронів у шарі, та кількість шарів, можна розв'язувати, наприклад, досить складні задачі розпізнавання образів.

Розглянемо тепер ШНМ, в який функція активації є неперервною. В цьому випадку виходом нейронної мережі є неперервна функція її входів. Таким чином таку ШНМ можна використовувати для апроксимації функцій. За узагальненою теоремою Стоуна за допомогою ШНМ з довільною функцією активації можна апроксимувати довільну неперервну функцію.

Для того, щоб за допомогою ШНМ розв'язувати задачі, наприклад розглянуті вище задачі розпізнавання образів або апроксимації, потрібно правильно дібрати вагові коефіцієнти нейронів (w_i). Для цього ШНМ потрібно "навчити". Існують алгоритми "навчання" 3-х видів.

1. Алгоритми "навчання з учителем". За цим алгоритмом за допомогою ШНМ розв'язують ряд прикладів, кожен з яких складається з двох частин:

- значення, що подають на вхід;
- значення, яке повинно бути на виході.

В процесі "навчання" вагові коефіцієнти модифікують таким чином, щоб за вхідними даними отримувати вихідні значення, максимально наближені до бажаних.

2. Алгоритми "навчання з заохоченням". В прикладах, що пропонуються розв'язати за допомогою ШНМ, не вказують бажане вихідне значення, але після проходження кожного прикладу виставляється оцінка, як було виконане завдання, добре чи погано.
3. Алгоритми "навчання без вчителя". За допомогою ШНМ розв'язують набір прикладів (без бажаного значення на виході) і в процесі їх опрацювання в ШНМ відбуваються певні процеси самоорганізації, що призводять до модифікації вагових коефіцієнтів так, що за допомогою ШНМ стає можливим розв'язати певну задачу.

Одним з найбільш популярних алгоритмів "навчання з учителем" є алгоритм зворотного розповсюдження. За цим алгоритмом спочатку вагові коефіцієнти вибирають випадковим чином. Є певний набір "навчальних" прикладів, що складається з пар *X*,*D* (вектор *X* на вході та бажане значення *D* на виході). Через *Y* позначено реальний вихід мережі. "Навчання" полягає в тому, щоб дібрати вагові коефіцієнти *W* так, щоб мінімізувати певну цільову функцію, в якості якої розглядають суму квадратів помилок ШНМ, тобто різниць між *Y* та *D*. Мінімізувавши цю функцію, отримують розв'язок за методом найменших квадратів. Більш детально метод найменших квадратів розглянуто в 4.3.1.

Для елементарного перцептрона можна запропонувати такий алгоритм "навчання з учителем":

1. Вибирають вагові коефіцієнти w_i випадковим чином.
2. На вхід подають значення першої частини x_i поточного тренувального прикладу і обчислюють *y*.

3. Порівнюють вихід (*y*) з бажаним значенням (*d*). Якщо отримане значення співпадає з бажаним, то вибирають наступний навчаючий приклад і переходять по пункту 2, в протилежному випадку переходять до пункту 4.
4. Обчислюють нові значення вагових коефіцієнтів:

   (нові w_i*)=(попередні* w_i*)* + $d*x_i$

і переходять до пункту 2, вибравши наступний навчаючий приклад.

Використання ШНМ дозволяє розв'язувати задачі з багатьох предметних галузей, наприклад:

1.Розпізнавання образів.

2.Прогнозування на фондовому ринку.

3.Визначення ризиків надання кредиту.

4.Управління роботом.

і інші.

Для формування у студентів компетентностей щодо створення найпростішої штучної нейронної мережі (елементарного перцептрона), доцільно розглянути з ними наступний приклад.

Приклад 1. Напишемо програму мовою Object Pascal, за якою можна проводити розпізнавання образів, а саме відрізняти прямокутники від відрізків.

Прямокутники і відрізки будуть задаватися зафарбованими точками на рисунку розміром *8x8*. Для задання такого рисунка знадобиться масив з *64* цілих чисел. Одиниці в цьому масиві визначатимуть зафарбовані точки, нулі – незафарбовані.

За типом *TVector* описують рисунки з прямокутниками чи відрізками.

За типом описують простий перцептрон з набором вагових коефіцієнтів w_i та пороговим значенням *Porig*.

Змінні *xk1, xk2, xk3, xk4, xk5, xk6* містять дані про рисунки прямокутників для "тренування" ШНМ.

Змінні *xp1, xp2, xp3, xp4, xp5, xp6* містять дані про рисунки відрізків для "тренування" ШНМ.

Змінні *xz1, xz2, xz3, xz4* містять дані про два квадрати та два відрізки, відмінні від тих, які використовувались у "тренуванні".

Функцією активації є функція *Ask*, вхідним параметром якої є масив, через який задають рисунок, і за якою повертається *1*, якщо рисунок є прямокутником і *-1*, якщо рисунок є відрізком.

За процедурою *Init_perceptron* ваговим коефіцієнтам надаються випадкові значення та ініціалізується порогове значення *Porig*.

За процедурою *Teach* проводиться "навчання" ШНМ. Параметрами цієї процедури є масив, що відповідає рисунку прямокутника чи відрізка і бажане значення на виході ШНМ *d* (*1* відповідає прямокутнику, *-1* – відрізку). За цією процедурою, якщо значення функції активації не співпадає з *d*, проводиться модифікація вагових коефіцієнтів w_i.

За програмою спочатку відбувається "навчання" ШНМ шляхом виклику процедури *Teach* з параметрами, що відповідають описаним в програмі прямокутникам (*xk1, xk2, xk3, xk4, xk5, xk6*) та відрізкам (*xp1, xp2, xp3, xp4, xp5, xp6*), а потім класифікація рисунків *xz1, xz2, xz3, xz4*, відмінних від тих, що використовувались у "тренуванні".

```
program Project1;
{$APPTYPE CONSOLE}
uses
  SysUtils;
const n=64;
type tvector=array[1..n] of integer;
const xk1:tvector=(1, 1, 1, 1, 1, 1, 1, 1,
                   1, 0, 0, 0, 0, 0, 0, 1,
                   1, 0, 0, 0, 0, 0, 0, 1,
                   1, 0, 0, 0, 0, 0, 0, 1,
                   1, 0, 0, 0, 0, 0, 0, 1,
                   1, 0, 0, 0, 0, 0, 0, 1,
                   1, 0, 0, 0, 0, 0, 0, 1,
```

```
                        1, 1, 1, 1, 1, 1, 1, 1
                        );
const xk2:tvector=(0, 1, 1, 1, 1, 1, 1, 1,
                        0, 1, 0, 0, 0, 0, 0, 1,
                        0, 1, 0, 0, 0, 0, 0, 1,
                        0, 1, 0, 0, 0, 0, 0, 1,
                        0, 1, 0, 0, 0, 0, 0, 1,
                        0, 1, 0, 0, 0, 0, 0, 1,
                        0, 1, 0, 0, 0, 0, 0, 1,
                        0, 1, 1, 1, 1, 1, 1, 1
                        );
const xk3:tvector=(0, 0, 0, 0, 0, 0, 0, 0,
                        1, 1, 1, 1, 1, 1, 1, 1,
                        1, 0, 0, 0, 0, 0, 0, 1,
                        1, 0, 0, 0, 0, 0, 0, 1,
                        1, 0, 0, 0, 0, 0, 0, 1,
                        1, 0, 0, 0, 0, 0, 0, 1,
                        1, 0, 0, 0, 0, 0, 0, 1,
                        1, 1, 1, 1, 1, 1, 1, 1
                        );
const xk4:tvector=(0, 0, 0, 0, 0, 0, 0, 0,
                        0, 0, 0, 0, 0, 0, 0, 0,
                        0, 0, 0, 0, 0, 0, 0, 0,
                        1, 1, 1, 1, 1, 1, 1, 1,
                        1, 0, 0, 0, 0, 0, 0, 1,
                        1, 0, 0, 0, 0, 0, 0, 1,
                        1, 0, 0, 0, 0, 0, 0, 1,
                        1, 1, 1, 1, 1, 1, 1, 1
                        );
const xk5:tvector=(0, 0, 1, 1, 1, 1, 1, 1,
                        0, 0, 1, 0, 0, 0, 0, 1,
                        0, 0, 1, 0, 0, 0, 0, 1,
                        0, 0, 1, 0, 0, 0, 0, 1,
                        0, 0, 1, 0, 0, 0, 0, 1,
                        0, 0, 1, 0, 0, 0, 0, 1,
                        0, 0, 1, 1, 1, 1, 1, 1,
```

```
                        0, 0, 0, 0, 0, 0, 0, 0
                        );
const xk6:tvector=(1, 1, 1, 1, 1, 1, 1, 1,
                        1, 0, 0, 0, 0, 0, 0, 1,
                        1, 0, 0, 0, 0, 0, 0, 1,
                        1, 0, 0, 0, 0, 0, 0, 1,
                        1, 0, 0, 0, 0, 0, 0, 1,
                        1, 0, 0, 0, 0, 0, 0, 1,
                        1, 0, 0, 0, 0, 0, 0, 1,
                        1, 1, 1, 1, 1, 1, 1, 1
                        );
const xp1:tvector=(1, 1, 1, 1, 1, 1, 1, 1,
                        0, 0, 0, 0, 0, 0, 0, 0,
                        0, 0, 0, 0, 0, 0, 0, 0,
                        0, 0, 0, 0, 0, 0, 0, 0,
                        0, 0, 0, 0, 0, 0, 0, 0,
                        0, 0, 0, 0, 0, 0, 0, 0,
                        0, 0, 0, 0, 0, 0, 0, 0,
                        0, 0, 0, 0, 0, 0, 0, 0
                        );
const xp2:tvector=(0, 0, 0, 0, 0, 0, 0, 1,
                        0, 0, 0, 0, 0, 0, 1, 0,
                        0, 0, 0, 0, 0, 1, 0, 0,
                        0, 0, 0, 0, 1, 0, 0, 0,
                        0, 0, 0, 1, 0, 0, 0, 0,
                        0, 0, 1, 0, 0, 0, 0, 0,
                        0, 1, 0, 0, 0, 0, 0, 0,
                        1, 0, 0, 0, 0, 0, 0, 0
                        );
const xp3:tvector=(0, 0, 0, 0, 0, 0, 0, 0,
                        0, 0, 0, 0, 0, 0, 0, 0,
                        0, 0, 0, 0, 0, 0, 0, 0,
                        0, 0, 0, 0, 0, 0, 0, 0,
                        1, 1, 1, 1, 1, 1, 1, 1,
                        0, 0, 0, 0, 0, 0, 0, 0,
                        0, 0, 0, 0, 0, 0, 0, 0,
```

```
                        0, 0, 0, 0, 0, 0, 0, 0
                        );
const xp4:tvector=(0, 1, 0, 0, 0, 0, 0, 0,

                        0, 1, 0, 0, 0, 0, 0, 0,
                        0, 1, 0, 0, 0, 0, 0, 0,
                        0, 1, 0, 0, 0, 0, 0, 0,
                        0, 1, 0, 0, 0, 0, 0, 0,
                        0, 1, 0, 0, 0, 0, 0, 0,
                        0, 1, 0, 0, 0, 0, 0, 0,
                        0, 1, 0, 0, 0, 0, 0, 0
                        );
const xp5:tvector=(1, 0, 0, 0, 0, 0, 0, 0,
                        0, 1, 0, 0, 0, 0, 0, 0,
                        0, 0, 1, 0, 0, 0, 0, 0,
                        0, 0, 0, 1, 0, 0, 0, 0,
                        0, 0, 0, 0, 1, 0, 0, 0,
                        0, 0, 0, 0, 0, 1, 0, 0,
                        0, 0, 0, 0, 0, 0, 1, 0,
                        0, 0, 0, 0, 0, 0, 0, 1
                        );
const xp6:tvector=(0, 0, 0, 0, 0, 1, 0, 0,
                        0, 0, 0, 0, 0, 1, 0, 0,
                        0, 0, 0, 0, 0, 1, 0, 0,
                        0, 0, 0, 0, 0, 1, 0, 0,
                        0, 0, 0, 0, 0, 1, 0, 0,
                        0, 0, 0, 0, 0, 1, 0, 0,
                        0, 0, 0, 0, 0, 1, 0, 0
                        );
const xz1:tvector=(0, 0, 1, 1, 1, 1, 1, 1,
                        0, 0, 1, 0, 0, 0, 0, 1,
                        0, 0, 1, 0, 0, 0, 0, 1,
                        0, 0, 1, 0, 0, 0, 0, 1,
                        0, 0, 1, 0, 0, 0, 0, 1,
                        0, 0, 1, 0, 0, 0, 0, 1,
```

```
                        0, 0, 1, 0, 0, 0, 0, 1,
                        0, 0, 1, 1, 1, 1, 1, 1
                        );
const xz2:tvector=(0, 0, 0, 0, 0, 0, 0, 0,
                        0, 0, 1, 1, 1, 1, 1, 0,
                        0, 0, 1, 0, 0, 0, 1, 0,
                        0, 0, 1, 0, 0, 0, 1, 0,
                        0, 0, 1, 0, 0, 0, 1, 0,
                        0, 0, 1, 0, 0, 0, 1, 0,
                        0, 0, 1, 1, 1, 1, 1, 0,
                        0, 0, 0, 0, 0, 0, 0, 0
                        );
const xz3:tvector=(0, 0, 0, 0, 0, 0, 0, 0,
                        0, 0, 0, 0, 0, 0, 0, 0,
                        0, 0, 0, 0, 0, 0, 0, 0,
                        0, 0, 0, 0, 0, 0, 0, 0,
                        0, 0, 0, 0, 0, 0, 0, 0,
                        1, 1, 1, 1, 1, 1, 1, 1,
                        0, 0, 0, 0, 0, 0, 0, 0
                        );
const xz4:tvector=(0, 0, 0, 0, 1, 0, 0, 0,
                        0, 0, 0, 1, 0, 0, 0, 0,
                        0, 0, 1, 0, 0, 0, 0, 0,
                        0, 1, 0, 0, 0, 0, 0, 0,
                        1, 0, 0, 0, 0, 0, 0, 0,
                        0, 0, 0, 0, 0, 0, 0, 0,
                        0, 0, 0, 0, 0, 0, 0, 0,
                        0, 0, 0, 0, 0, 0, 0, 0
                        );
type tperceptron=record
           w:array[1..n] of integer;
           porig:integer;
      end;
var perceptron:tperceptron;
function ask(x:tvector):integer;
```

```
var i,s:integer;
begin
  s:=0;
  for i:=1 to n do
    s:=s+x[i]*perceptron.w[i];
  if (s>perceptron.porig) then
    ask:=1
  else
    ask:=-1;
end;
procedure init_perceptron;
var i:integer;
begin
  randomize;
  for i:=1 to n do
    perceptron.w[i]:=random(10);
  perceptron.porig:=70;
end;
procedure teach(x:tvector;d:integer);
var i:integer;
begin
  if (d<>ask(x)) then
    for i:=1 to n do
       perceptron.w[i]:=perceptron.w[i]+d*x[i];
end;
begin
  init_perceptron;
  teach(xk1,1);
  teach(xk2,1);
  teach(xk3,1);
  teach(xk4,1);
  teach(xk5,1);
  teach(xk6,1);
  teach(xp1,-1);
  teach(xp2,-1);
  teach(xp3,-1);
```

```
  teach(xp4,-1);
  teach(xp5,-1);
  teach(xp6,-1);
  if (ask(xz1)=1) then writeln('Rectangle')
  else writeln('Line');
  if (ask(xz2)=1) then writeln('Rectangle')
  else writeln('Line');
  if (ask(xz3)=1) then writeln('Rectangle')
  else writeln('Line');
  if (ask(xz4)=1) then writeln('Rectangle')
  else writeln('Line');
  readln;
end.
```

3.2.8. Модель логіки предикатів. Логіка предикатів може використовуватися як одна з мов подання знань і одночасно з цим вона має важливе значення як універсальний засіб опису базової структури і подання знань. Визначаючи мову предикатів, потрібно визначити три речі: синтаксис, семантику і операції.

Для побудови слів у мові предикатів можна використовувати будь-які літери та деякі спеціальні символи, за винятком зарезервованих символів. Скінченні набори символів, що починаються з великої літери, будемо вважати змінними, а решту – константами. Набір з n констант або змінних, що взятий в круглі дужки і слідує за іменем (скінченний набір символів), будемо називати функцією, наприклад *f(X,Y,Z)*. Константи, змінні та функції називають термами. Набір з *n* термів, взятий в дужки і такий, що слідує за іменем (скінченний набір символів), називається атомарним предикатом. Наприклад атомарним предикатом є *син(X, петро)*. Предикат набуває одного з двох значень – істинно чи хибно в залежності від значень констант, змінних і функцій, що входять в нього. Із атомарних предикатів за допомогою логічних операцій можна складати логічні речення (формули).

Для вказування логічних операцій використовують зарезервовані символи. Логічне “і” позначається символом “/\”, логічне “або” символом “\/”, заперечення символом “~”, імплікація позначається символом “⇒”, а еквіваленція символом “⇔”. Також до зарезервованих символів відносяться квантор загальності “∀” та квантор існування ”∃”.

Сформулюємо правила, дотримання яких дозволяє будувати правильну логічну формулу.

1. Атомарний предикат є правильною логічною формулою.
2. Якщо *A, B* – правильні логічні формули, то *A/\B, A\/B, ~A, A⇒B, A⇔B* – також правильні логічні формули.
3. Якщо *A(X)* – правильна логічна формула, то вирази *(∀X)A(X), (∃X)A(X)* також є правильними логічними формулами.
4. Всі результати, отримані застосуванням скінченну кількість разів правил 1-3 також є правильними логічними формулами.

Щодо семантики мови предикатів, то слід сказати, що конструкції цієї мови самі по собі ніяк не пов’язані з тими предметними галузями, для моделювання яких їх використовують. Розробник повинен чітко вказати, яка предикатна форма і для опису якої сутності предметної галузі використовується. Наприклад предикат *батько(петро, іван)* може означати як те, що Петро батько Івана, так і те, що Іван батько Петра. Щоб не було таких двозначностей, потрібно:

1. Встановити відповідність між константами моделі, заснованої на логіці предикатів, і об’єктами предметної галузі.
2. Встановити відповідність між формулами моделі, заснованої на логіці предикатів, і функціональними відношеннями між об’єктами предметної галузі.
3. Встановити відповідність між атомарними предикатами моделі, заснованої на логіці предикатів, і концептуальними відношеннями предметної галузі.

У формули логіки предикатів входять як змінні, семантично обмежені кванторами існування та загальності (квантовані змінні), так і не обмежені кванторами змінні (неквантовані змінні). Квантовані змінні називаються зв'язаними, а неквантовані – вільними. Логічна формула, в якій всі змінні є зв'язаними, називається реченням. Дуже важливо, що реченню можна однозначно поставити у відповідність значення "істинно" або "хибно". Наприклад речення *(∀X)(∃Y)батько(X,Y)*, яке будемо тлумачити "Для кожної людини *X* існує батько *Y*" є завжди істинним. Якщо ж у формулі логіки предикатів використовуються вільні змінні, то про істинність або хибність такої формули можна сказати тільки після підстановки замість змінних конкретних значень. Тобто не можна сказати, чи є істинною формула *батько(X,У)*. А після підстановки формула *батько(іван, петро)* буде істинною, якщо це істинно у відповідній предметній галузі. Формулу, в яку зроблено підстановку, будемо називати висловлюванням.

Виведенням у моделі логіки предикатів будемо називати процедуру, за якою із заданої групи виразів виводиться новий вираз, якого не було серед заданих.

У якості повністю формалізованого методу виведення можна використовувати метод резолюції. Для використання методу резолюції потрібно початкові логічні формули перетворити у певну нормальну форму. Таке перетворення здійснюється за кілька стадій.

У першій стадії формули зводять до префіксної нормальної форми. В цій формі всі квантори виводяться за дужки і займають місце на початку логічної формули. Для цього використовують кілька правил, наприклад із формули *(∀X)A(X)∧(∀X)B(X)* можна отримати формулу *(∀X)(A(X)∧B(X))*.

У другій стадії за допомогою перетворень повністю виключають квантори з предикатних формул. Такі перетворення роблять за допомогою так званих сколемівських функцій. Отримані в результаті таких перетворень предикатні формули називають сколемівською нормальною формою.

На третій стадії переходять до кон’юнктивної нормальної форми. Кон’юнктивна нормальна форма утворюється з висловлювань, що об’єднані диз’юнкціями, а ці диз’юнкції об’єднані кон’юнкціями, наприклад:

$(A \vee B) \wedge (C \vee D) \wedge (K \vee M)$

Така уніфікація дуже важлива для комп’ютерного опрацювання.

Далі кон’юнктивну нормальну форму зводять до так званої клаузальної форми, в якій кожен диз’юнкт називають реченням.

Якщо всі речення є висловленнями, тобто не містять змінних, то принцип резолюції полягатиме в наступному: об’єднуючи речення клаузальної форми і подаючи їх у вигляді диз’юнктів, отримують нові речення. Якщо в результаті одного з таких об’єднань буде отримано порожнє речення, то це доводить хибність початкової формули.

Якщо у клаузальній формі деякі предикати містять змінні, то процес виведення дещо ускладнюється. Для можливості доведення виконують уніфікацію формул шляхом підстановки замість змінних відповідних констант.

Подання знань у вигляді моделі логіки предикатів використовується у мові логічного програмування з відповідною назвою ПРОЛОГ (ПРОграмування ЛОГічне), також використовуються такі назви цієї мови, як Пролог, Prolog.

Перша Пролог-програма була написана на початку 1970-х років у Франції в рамках проекту стосовно розуміння природної мови. Основний етап розвитку мови Пролог припадає на 1975-1979 рр., коли на кафедрі штучного інтелекту університету Единбургу Девід Уоррен (David H.D. Warren) і Фернандо Перейра (Fernando Pereira) відповідали за реалізацію цієї мови. Вони створили перший інтерпретатор Пролог. Ця версія стала першим стандартом мови Пролог.

У мові Пролог підтримується так звана декларативна парадигма програмування.

3.2.9. Мова Пролог та її основні поняття. Як було зазначено вище, Пролог базується на декларативній парадигмі програмування. Програма цією мовою – це опис певної бази знань, що складається з фактів та правил, записаних у термінах логіки предикатів першого порядку. За допомогою машини виведення, що базується на принципі резолюції, можна отримувати нові факти з цієї бази знань за допомогою запитань, поставлених користувачем. Ця декларативність Прологу корінним чином змінює мислення програміста і робить навчання програмування мовою Пролог захоплюючим заняттям, яке потребує певних інтелектуальних зусиль [20, 215, 327]. Слід зазначити, що для того, щоб сформувати певні компетентності у програмуванні цією мовою, студенти повинні досить глибоко змінити ті навички імперативного програмування, які вони отримали при вивченні процедурних мов, таких як Паскаль, подивитися на задачу з іншого боку і тим самим вдосконалити своє мислення. Наприклад, відсутність у Пролозі такого стандартного підходу до розв’язування багатьох задач, як цикли, змусить студентів глибше з'ясувати сутність рекурсії. Хоч рекурсія вивчається і в імперативних мовах, але там вона є допоміжним механізмом і не є доцільною у багатьох задачах, де зручнішим і ефективнішим є цикл. Ті ж задачі, де рекурсія є природною в імперативних мовах, виявляються достатньо складними для багатьох студентів.

Пролог найкраще придатний для опрацювання символьних виразів, особливо для задач, в яких фігурують певні об’єкти і відношення між ними. В більшості літературних джерел вивчення Прологу починається якраз з прикладу побудови дерева родинних відношень. Хоча за допомогою цієї мови можна описувати програми і для розв’язування задач, пов’язаних з опрацюванням числових даних.

Перед тим, як розглянути приклади програм, описаних мовою Пролог, зупинимось на деяких синтаксичних особливостях даної мови. Ці особливості характерні для більшості діалектів мови Пролог.

1. Відношення між об’єктами предметної галузі задаються за допомогою предикатів.
2. Аргументи відношень можуть бути конкретними об’єктами, тобто константами (які називають атомами), або абстрактними об’єктами, тобто змінними. Описи атомів починаються з маленької літери, а змінних – з великої або символу підкреслення. Якщо опис певного атома потрібно написати з великої літери, його беруть у лапки.
3. Пролог-програма складається з речень, кожне з яких завершується крапкою.
4. Логічне “і” позначається комою (,), а логічне “або” – крапкою з комою (;).
5. Серед усіх змінних особливе місце займає так звана анонімна змінна, яка записується у вигляді одного знаку підкреслення. Анонімну змінну використовують у тих випадках, коли ім’я змінної не важливе, наприклад предикат *батько(X,_)* означає, що у *X* є батько. Якщо у реченні кілька разів записано знак підкреслення, то кожен такий новий запис означає нову змінну, відмінну від попередньої, наприклад предикат *батько(_,_)* означає, що існують два об’єкти, один з яких є батьком іншого.
6. Лексичний діапазон імені змінної – одне речення. Це значить, що якщо, наприклад, ім’я *X* зустрічається в двох реченнях, то в кожному з них воно означає змінну, ніяк не пов’язану з іншою. Для констант ситуація зовсім інша: один і той самий атом завжди позначає один і той самий об’єкт в будь-якому реченні.

3.2.10. Факти і правила в мові Пролог. Розглянемо тепер приклад досить простої програми мовою Пролог. В ній дещо відійдемо від традиції і розглянемо не родинні зв’язки, а певний оркестр:

струнний_інструмент(скрипка).

струнний_інструмент(альт).

духовий_інструмент(труба).

грає(петро,альт).

грає(степан,труба).

грає(тетяна,скрипка).

грає(людмила,скрипка).

Тепер можна зробити, наприклад, такий запит до бази знань (сформулювати ціль):

грає(Х,скрипка)

Він означатиме, що потрібно визначити всіх, то грає на скрипці. В результаті співставлення зі змінною *Х* одне за одним відповідних імен з бази знань, за Пролог-програмою надрукуються ті з них, які задоволяняють сформульований запит. Тому у відповіді буде одержано:

Х=тетяна

Х=людмила

У розглянутій вище програмі база знань складалася тільки з фактів, але там можуть бути і правила, які є у кожній нетривіальній програмі.

Правила у Пролозі описуються наступним чином: описується новий предикат – заголовок правила, а потім через сукупність символів ":-" вказується його зв'язок з вже існуючими – тіло правила. Наприклад опишемо таке правило "струнна група оркестру – це всі ті, хто грає на струнних інструментах", тобто

для всіх Х і У

Х належить до струнної групи, якщо

Х грає на У і

У – струнний інструмент.

Мовою Пролог це буде записано так:

струнна_група(Х):-грає(Х,У), струнний_інструмент(У).

Якщо тепер до розширеної таким правилом бази знань сформулювати ціль

струнна_група(К)

отримаємо відповідь

К=петро

К=тетяна

К=людмила

Оволодіння навіть такими початковими знаннями щодо програмування мовою Пролог дозволяє розв'язувати такі логічні задачі, які важко розв'язати за допомогою імперативних мов програмування. Наведемо приклади таких задач.

Задача 3.2. Кожен із трьох студентів (Сергій, Борис, Віктор) народився в одному з трьох міст – Рівне, Миколаїв, Львів (всі студенти народилися в різних містах) і їдуть на велосипедах, виготовлених в кожному з цих міст, але жоден із них не їде на велосипеді, виготовленому в рідному місті. Сергій їде на велосипеді з Рівного, у Віктора велосипед з Миколаєва, Віктор родом зі Львова. Хто звідки родом і на якому велосипеді їде?

Розв'язування.

Визначимо предикати задачі:

місто(X) – *X* є містом

студент(X) – *X* є студентом

їде(X,У) – студент *X* їде на велосипеді, виготовленому в місті *У*

родом(X,У) – студент *X* народився в місті *У*

їдуть – предикат для визначення, хто з студентів на якому велосипеді їде

звідки_родом – предикат для визначення, хто зі студентів звідки родом

Тепер визначимо речення програми, які безпосередньо випливають з умови:

факти:

місто(рівне).

місто(львів).

місто(миколаїв).

студент(віктор).

студент(сергій).

студент(борис).

їде(сергій,рівне).

їде(віктор,миколаїв).

родом(віктор,львів).

До списку речень також необхідно додати правило, за яким вказується, що кожен студент народився у місті, в якому не виговлено велосипед, на якому він їде:

їде(X,У):-not родом(X,У).

Також до списку речень додамо правила, де описуються предикати *їдуть* та *звідки_родом*. В правилі *їдуть* використовуються змінні *Віктор*, *Сергій* та *Борис* для зберігання даних про міста, в яких виготовлено велосипеди, на яких їдуть студенти з відповідними іменами. В правилі *звідки_родом* використовуються змінні з такими самими іменами, але використовуються вони для зберігання даних про міста, в яких народилися студенти з відповідними іменами:

```
їдуть:-місто(Віктор), місто(Борис), місто(Сергій),
        Віктор<>Борис, Віктор<>Сергій, Сергій<>Борис,
        Їде(віктор, Віктор), їде(сергій, Сергій),
        write("велосипед Віктора з ", Віктор), nl,
        write("велосипед Бориса з ", Борис), nl,
        write("велосипед Сергія з ", Сергій), nl.
звідки_родом:- місто(Віктор), місто(Борис), місто(Сергій),
        Віктор<>Борис, Віктор<>Сергій, Сергій<>Борис,
        родом(віктор, Віктор),
        write("родом Віктор з ", Віктор), nl,
        write("родом Борис з ", Борис), nl,
        write("родом Сергій з ", Сергій), nl.
```

Для отримання відповіді, в яких містах виготовлені велосипеди студентів, достатньо сформулювати запит:

їдуть

Для отримання відповіді, з яких міст родом студенти, достатньо сформулювати запит:

звідки_родом

Задача 3.3. На заводі працюють три товариші: слюсар, токар та зварник. Їх прізвища: Борисенко, Іваненко, Сергієнко. У слюсаря нема ні братів, ні сестер. Він наймолодший від усіх друзів. Сергієнко одружений на сестрі Борисенка, старший від токаря. Потрібно назвати прізвища слюсаря, токаря та зварника.

Розв'язування.

Опишемо наступні предикати:

прізвище(X) – у робітника прізвище *X*

має_сестру(X) – робітник *X* має сестру

старший_від_токаря(X) – робітник *X* старший, ніж токар

розв'язування – предикат, виклик якого призводить до друку розв'язку

Тепер запишемо факти, які безпосередньо випливають з умови:

прізвище(борисенко).

прізвище(сергієнко).

прізвище(іваненко").

має_сестру(борисенко).

старший_від_токаря(сергієнко).

Додамо до цього списку речень правило *розв'язування*, за яким будуть надруковані прізвища працівників. В цьому правилі використовуються змінні *Слюсар*, *Токар* та *Зварник*, які призначені для зберігання прізвищ слюсаря, токаря та зварника відповідно:

розв'язування:- прізище(Слюсар),

прізвище(Токар),
прізвище(Зварник),
Токар<>Слюсар,
Зварник<>Токар,
Зварник<>Слюсар,
not(має_сестру(Слюсар)),
not(старший_від_токаря(Слюсар)),
not(старший_від_токаря(Токар)),
write("Слюсар – ", Слюсар), nl,
write("Токар – ", Токар), nl,
write("Зварник – ", Зварник), nl,
fail.

Для друку відповіді достатньо зробити запит

розв'язування

Для формування вмінь і навичок написання простих програм (без використання розгалужень, рекурсії та інших прийомів) мовою Пролог можна запропонувати студентам самостійно описати базу знань про родинні відносини.

3.2.11. Реалізація розгалужень в описах програм мовою Пролог. Відтинання. Розглянемо, як можна змоделювати мовою Пролог таку широко використовувану в імперативних мовах програмування конструкцію, як розгалуження.

Для цього розглянемо просту задачу про знаходження більшого з двох чисел і порівняємо, як реалізується розв’язування цієї задачі мовою імперативного програмування (Паскаль) та мовою декларативного програмування (Пролог).

Відповідний фрагмент програми, описаний мовою Паскаль, виглядатиме так:

procedure Max2(a, b: Integer; var m: Integer);

```
begin
  if a>=b then
    m:=a
  else
    m:=b;
end;
```

Для реалізації цієї процедури мовою Пролог потрібно описати два правила. Перше відповідатиме ситуації, коли перше число більше другого, друге правило відповідатиме протилежній ситуації:

max2(A,B,M):-A>=B, M=A.

max2(A,B,M):-A<B, M=B.

За вказівкою *max2(5,7,M)* буде отримано відповідь

M=7

Розглядаючи ці правила, можна помітити, що вони є взаємно виключаючими, тобто якщо виконується перше, друге обов'язково не виконується. Якщо не виконується перше правило, друге обов'язково виконується. Отже як тільки виконується одне з правил, нема сенсу перевіряти інші. Залишилося описати це за наведеними правилами. В мові Пролог це можна реалізувати за допомогою відтинання. Відтинання позначається знаком "!" і показує, що не треба повертатися з точки, де використане відтинання, для перегляду інших правил з таким самим заголовком. Отримаємо:

max2(A,B,M):-A>=B, M=A, !.

max2(A,B,M):-M=B.

В цьому прикладі при виконанні умови *A>=B* за програмою здійсниться відтинання, а це означає, що друге правило не буде розглядатися.

Для вдосконалення у студентів вмінь і навичок описувати розгалуження і використовувати відтинання мовою Пролог можна

запропонувати їм написати програму для знаходження більшого з трьох чисел.

3.2.12. Циклічні дії та рекурсія у мові Пролог. В багатьох програмах потрібно певну дію (дії) повторити кілька разів. В імперативних мовах програмування для цього використовують в основному цикли і тільки в певних, досить обмежених класах задач, рекурсію.

У мові Пролог операторів циклу немає, але виконання повторень у певних ситуаціях можна організувати, використовуючи таке поняття, як відкат. Відкат організовується через вбудований предикат *fail (*невдача), за яким завжди повертається значення “хибно”. Розглянемо, як за допомогою відкату організувати повторення.

Нехай є база знань про фрукти:

фрукт(яблуко).

фрукт(груша).

фрукт(персик).

фрукт(слива).

фрукт(вишня).

Треба надрукувати назви всіх фруктів, описаних в базі знань. Для друкування використаємо предикати write – друк, та nl – переведення курсору на новий рядок (як в діалекті Прологу під назвою Turbo Prolog).

Розширимо базу знань правилом друкування:

друк_назв_фруктів:-фрукт(X), write(X), nl, fail.

Якщо тепер зробити запит

друк_назв_фруктів

отримаємо на екрані стовпчик з назвами перелічених вище фруктів. При цьому спочатку змінна *X* набуває значення назви першого фрукта – яблуко, це значення друкується на екрані і курсор переходить на новий рядок, за предикатом *fail* повертається значення “хибно”, а отже забезпечується невиконання цього правила, тому змінній *X* надається наступне значення і дії

повторюються. Таким чином буде надруковано назви всіх фруктів, що є в базі даних, наведеній вище.

У такий спосіб можна друкувати тільки ті дані з бази знань, що задовольняють певні умови. Опишемо базу знань про заробітні плати співробітників і опишемо в ній правило друкування прізвищ і зарплат співробітників, у яких заробітна плата більша 1500 грн.

зарплата(петренко, 1200).

зарплата(сидоренко, 2300).

зарплата(іваненко, 1800).

зарплата(степаненко, 1100).

зарплата(давиденко, 1900).

зарплата(сергієнко, 2200).

зарплата(григоренко, 1200).

друк_зарплати_більшої_1500:-зарплата(X, У), У>1500, write (X,У), nl, fail.

У відповідь на запит

друк_зарплати_більшої_1500

отримаємо

сидоренко 2300

іваненко 1800

давиденко 1900

сергієнко 2200

Розглянемо тепер використання рекурсії у мові Пролог. Рекурсія є одним з фундаментальних прийомів програмування мовою Пролог. Без рекурсії в ньому неможливо розв’язувати більш-менш складні задачі.

Рекурсивним називають правило, яке містить саме себе в якості однієї із своїх компонент. Використання рекурсії дозволяє виконувати певний набір дій кілька разів.

Для правильної організації рекурсії потрібно виконання кількох умов. По-перше для того, щоб рекурсія відбулася, потрібно в описі рекурсивного правила викликати це саме правило. По-друге, щоб рекурсія завершилася за потрібну кількість кроків, слід забезпечити правило виходу з рекурсії.

Покажемо реалізацію рекурсії на прикладі задачі обчислення факторіала. Як відомо, факторіал числа n позначається n! і обчислюється за такими формулами:

0!=1

*n>0, n!=1*2*...*n*

За таким означенням факторіала природним засобом його обчислення є цикл. Але у Пролозі немає циклічних операторів, як у мовах імперативного програмування, тому, щоб скористатися рекурсією, використаємо інше, рекурсивне означення факторіала, а саме:

0!=1

*n>0, n!=(n-1)!*n*

Перше правило в ньому є правилом виходу, а завдяки другому забезпечується рекурсія. Мовою Пролог опис такого фрагмента програми виглядатиме так:

факторіал(0,1).

*факторіал(N,F):-N1=N-1, факторіал(N1,F1), F=F1*N.*

Поставивши тепер запит

факторіал(5, X)

отримаємо

X=120

Наступним прикладом використання рекурсії розглянемо задачу знаходження найбільшого спільного дільника (НОД). Позначимо початкові числа Х та У, результат – Д. Для знаходження НОД скористаємося алгоритмом Евкліда, який складається з наступних правил:

1. Якщо Х дорівнює У, то Д дорівнює Х.

2. Якщо Х>У, то Д дорівнює НОД У та Х-У.

3. Якщо У>Х, то Д дорівнює НОД Х та У-Х.

Опис програми мовою Пролог виглядатиме так:

нод(Х, Х, Х).

нод(Х, У, Д):-Х>У, Х1=Х-У, нод(Х1, У, Д).

нод(Х, У, Д):-У>Х, У1=У-Х, нод(Х, У1, Д).

В якості ще одного прикладу використання рекурсії наведемо розв'язування задачі знаходження n-го члена послідовності чисел Фібоначчі. Нагадаємо, що два перших члени цієї послідовності дорівнюють 1, а потім кожен наступний член дорівнює сумі двох попередніх, тобто

Ф(1)=1,

Ф(2)=1,

n>2, Ф(n)=Ф(n-1)+Ф(n-2)

Як бачимо, послідовність чисел Фібоначчі визначається рекурсивно, тому можна скористатися означенням і написати наступну програму мовою Пролог:

фіб(1, 1).

фіб(2, 1).

фіб(Н, Ф):-Н1=Н-1, фіб(Н1, Ф1), Н2=Н-2, фіб(Н2, Ф2), Ф=Ф1+Ф2.

Проте можна помітити, що дана програма не є ефективною, оскільки одні й ті самі обчислення виконуються два рази, перший раз при рекурсивному виклику *фіб(Н1, Ф1)*, а другий раз – при виклику *фіб(Н2, Ф2)*. Цих повторних обчислень можна позбутися, скориставшись іншим алгоритмом, ідея якого полягає в тому, що потрібно визначати *n*-й член послідовності, починаючи з перших членів і просуватися вперед, обчислюючи члени послідовності один за іншим. Зупинитися потрібно, коли буде обчислено *n*-й член. Для цього визначимо предикат

фіб1(М, Н, Ф1, Ф2, Ф)

де *М* – номер поточного члена послідовності, *Н* – номер *n*-го члена послідовності, *Ф1* – (*m-1*)-й член послідовності, *Ф2* – *m*-й член послідовності, *Ф* – *n*-й член послідовності. Тоді програма набуде вигляду:

фіб(Н, Ф):-фіб1(2, Н, 1, 1, Ф).

фіб1(М, Н, Ф1, Ф2, Ф2):-М>=Н.

фіб1(М, Н, Ф1, Ф2, Ф2):-М<Н, М1=М+1, Ф3=Ф1+Ф2,

фіб1(М1, Н, Ф2, Ф3, Ф).

В цій програмі в першому реченні вказується, що перші два числа послідовності Фібоначчі дорівнюють 1. У другому реченні вказується, що якщо *n*-те число досягнуто, то відповіддю буде *Ф2*. У третьому реченні вказується, що якщо *n*-те число ще не досягнуто, потрібно додати два попередніх члена послідовності і організувати рекурсію.

Для формування вмінь і навичок використання рекурсії в описах програм мовою Пролог можна запропонувати студентам розв'язати наступні задачі:

- Знаходження суми натуральних чисел від 1 до *n*.
- Друкування чисел від 1 до 10.

3.2.13. Робота зі списками **у мові Пролог.** Список у мові Пролог – це довільний скінчений впорядкований набір елементів, взятий в квадратні дужки, наприклад [яблуко, груша, вишня]. Існує поняття порожнього списку, який записується []. Довжиною списку називають кількість елементів у ньому. Довжина порожнього списку дорівнює нулю.

Для опрацювання списків широко застосовується підхід, коли список розділяють на "голову" та "хвіст" ("голова" – перший елемент списку, "хвіст" – решта елементів). "Голову" опрацьовують у потрібний спосіб, а потім таке опрацювання рекурсивно повторюють для "хвоста". Поділ списку на голову і хвіст записують наступним чином:

[Г|Х]

де змінна *Г* позначає "голову" списку, а *Х* – його "хвіст".

Опишемо деякі операції, які можна виконувати над списком.

Друкування списку.

Щоб надрукувати список, скористаємося поділом списку на “голову” та “хвіст”, надрукуємо “голову” і рекурсивно повторимо друкування для “хвоста” списку. Отримаємо такі речення:

друк([]).

друк([Г|Х]):-write(Г), nl, друк(Х).

В першому реченні вказується, що порожній список друкувати не потрібно, це правило є правилом виходу з рекурсії; за вказівкою з другого речення рекурсивно друкується непорожній список.

Якщо поставити запит

друк([3, 5, 4, 12, 7])

отримаємо

3

5

4

12

7

Зауважимо, що для того, щоб надрукувати список у зворотному порядку, достатньо переписати друге речення наступним чином:

друк([Г|Х]):-друк(Х), write(Г), nl.

Обчислення довжини списку.

Ця задача розв’язується аналогічно до попередньої:

довжина([], 0).

довжина([_|Х], Н):-довжина(Х,Н1), Н=Н1+1.

У відповідь на запит *довжина([3,5,8])* буде виведено результат 3.

Перевірка належності елемента до списку.

Таку перевірку також опишемо рекурсивно:

належить(Г,[Г|Х]).

належить(К,[Г|Х]):-належить(К,Х).

В першому реченні вказується, що якщо шуканий елемент співпадає з головою списку, то він списку належить; це правило також служить правилом виходу з рекурсії. В другому реченні вказується, що якщо шуканий елемент не співпадає з головою списку, то його потрібно шукати у хвості списку.

Приєднання нового елемента до списку.

Найбільш простий спосіб приєднати новий елемент до списку – це поставити його в голову цього списку. Це легко зробити за допомогою наступного речення, в якому К – новий елемент, С – список, в який додається цей елемент:

додати(К, С, [К|С]).

Щоб перевірити роботу такого правила додавання, об'єднаємо додавання нового елемента із друкуванням списку:

друк([]).

друк([Г|Х]):-write(Г," "), друк(Х).

додати(К, С, [К|С]).

і сформулюємо наступну ціль:

С=[2,4,5,8,7], додати(8,С,С1), друк(С1)

Отримаємо відповідь

8 2 4 5 8 7

Якщо елемент, що додається, в списку вже є, він продублюється (у наведеному вище прикладі продублювалося число 8). Якщо ж дублювання елементів у списку недопустиме, правило додавання потрібно змінити, перевіряючи спочатку, чи є в списку елемент, що додається (за допомогою правила "належить"). Якщо він є в списку, то від додавання потрібно відмовитися за допомогою відтинання. Отримаємо:

належить(Г,[Г|Х]).

належить(К,[Г|Х]):-належить(К,Х).

додати(К, С, С):-належить(К,С), !.

додати(К, С, [К|С]).

Об’єднання двох списків.

Об’єднання двох спиків в один загальний можна виконати за допомогою двох наступних правил:

об’єднати([], С, С).

об’єднати([Г1|Х1],С, [Г1|Х3]):-об’єднати(Х1,С,Х3).

В першому правилі вказується, що результатом приєднання порожнього списку до списку С буде цей самий список С. Це є правилом виходу з рекурсії. В другому правилі вказується, що для того, щоб приєднати непорожній список, що має “голову” *Г1,* до списку *С*, потрібно цю “голову” зробити “головою” об’єднаного списку, а потім рекурсивно приєднати “хвіст” першого списку до списку С.

Перевірити роботу цих правил можна, об’єднавши їх з правилами друкування списку та сформулювавши, наприклад, наступну ціль:

С1=[1,2,3], С2=[4,5,6], об’єднати(С1, С2, С3), друк(С3)

Отримаємо відповідь:

1 2 3 4 5 6

Вилучення елемента зі списку.

Вилучення елемента зі списку реалізується за наступними правилами:

вилучити(Г, [Г|Х], Х).

вилучити(Г, [Г1|Х1], [Г1|Х2]):-вилучити(Г, Х1, Х2).

В першому правилі вказується, що якщо елемент, що вилучається, є “головою” списку, то результатом вилучення буде “хвіст” цього списку. Це є правилом виходу з рекурсії. В другому правилі вказується, що якщо елемент, який вилучається, не є “головою” списку, треба рекурсивно його вилучити з “хвоста” списку.

Перевірити роботу за цими правилами можна, об’єднавши їх з правилами друкування списку та сформулювавши, наприклад, наступну ціль:

С=[1,2,3,7,4], вилучити(3, С, С1), друк(С1)

Отримаємо відповідь:

1 2 7 4

Сортування списку.

Списки можна сортувати за зростанням або за спаданням, причому існують різні методи сортування. При вивченні імперативних мов програмування студенти знайомилися, наприклад, із сортуванням за методом “бульбашки”. Нагадаємо, в чому полягає цей метод. Для конкретизації розглянемо сортування за зростанням. В цьому випадку метод бульбашки полягає в пошуку двох сусідніх елементів, лівий з яких більший за правий. Якщо така пара знайдена, то елементи в ній міняються місцями і пошук продовжується. Якщо таку пару знайти не вдається, список елементів вважається відсортованим.

Опишемо сортування списку за зростанням за методом “бульбашки” мовою Пролог.

бульбсорт(С, УС):-перестановка(С, С1), !, бульбсорт(С1, УС).

бульбсорт(УС, УС).

С – початковий список, *УС* – відсортований. В першому правилі вказується, що якщо у списку вдалося зробити перестановку, в результаті якої зі списку *С* утворився список *С1*, то друге правило вже не розглядається, а виконується рекурсивний виклик *бульбсорт(С1, УС)* для сортування списку *С1*. Якщо ж перестановки виконати не вдалося, здійснюється перехід до другого правила, за яким вказується, що список вже відсортований.

Тепер залишилося написати правила для перестановки:

перестановка([Г1, Г2|Х],[Г2, Г1|Х]):-Г1>Г2.

перестановка([Г|Х], [Г|Х1]):-перестановка(Х, Х1).

В першому правилі вказується, що якщо два елементи, які знаходяться в “голові” списку, потребують перестановки, то їх потрібно переставити. Це є правилом виходу з рекурсії. В протилежному випадку перестановка рекурсивно викликається для “хвоста” списку.

Об’єднаємо всі 4 правила і отримаємо програму сортування за методом “бульбашки”:

бульбсорт(С, УС):-перестановка(С, С1), !, бульбсорт(С1, УС).

бульбсорт(УС, УС).

перестановка([Г1, Г2|Х],[Г2, Г1|Х]):-Г1>Г2.

перестановка([Г|Х], [Г|Х1]):-перестановка(Х, Х1).

Якщо до цих правил додати правила друкування списку і поставити, наприклад, ціль:

бульбсорт([2, 5, 3, 4, 7, 1, 9], С), друк(С)

отримаємо:

1 2 3 4 5 7 9

Оволодівши таким потужним інструментарієм Прологу, як опрацювання списків, студенти зможуть розв'язувати більш складні задачі. В якості такої досить складної задачі можна запропонувати задачу про 8 ферзів [20].

Задача 3.4. Задача полягає у тому, щоб так розставити 8 ферзів на шаховій дошці, щоб жоден з них не погрожував іншому.

Як відомо, ферзь б'є вздовж горизонталі, вертикалі та обох діагоналях. Тому, щоб забезпечити безпеку всіх ферзів, потрібно, щоб всі вони були розміщені на різних вертикалях, горизонталях та діагоналях. Таким чином позиція кожного ферзя може характеризуватися 4-ма числами: числом, що характеризує вертикаль (*X*), числом, що характеризує горизонталь (*Y*), числом, що характеризує діагональ, паралельну до головної діагоналі шахової дошки (*V*) та числом, що характеризує діагональ, паралельну до побічної діагоналі шахової дошки (*U*).

Значення *X* належить списку *Dx=[1,2,3,4,5,6,7,8]*.

Значення *Y* належить списку *Dy=[1,2,3,4,5,6,7,8]*.

З рис. 3.7 можна помітити, що для кожної клітинки діагоналі, паралельної до головної діагоналі, інваріантом є значення *V=X+Y*, таким чином значення *V* належить списку *Dv=[2,3,4,5,6,7,8,9,10,11,12,13,14,15,16]*.

Також з цього рисунка можна помітити, що інваріантом для кожної клітинки діагоналі, паралельної побічній діагоналі, інваріантом є значення *U=X-Y*, таким чином значення *U* належить списку *Du=[-7,-6,-5,-4,-3,-2,-1,0,1,2,3,4,5,6,7]*.

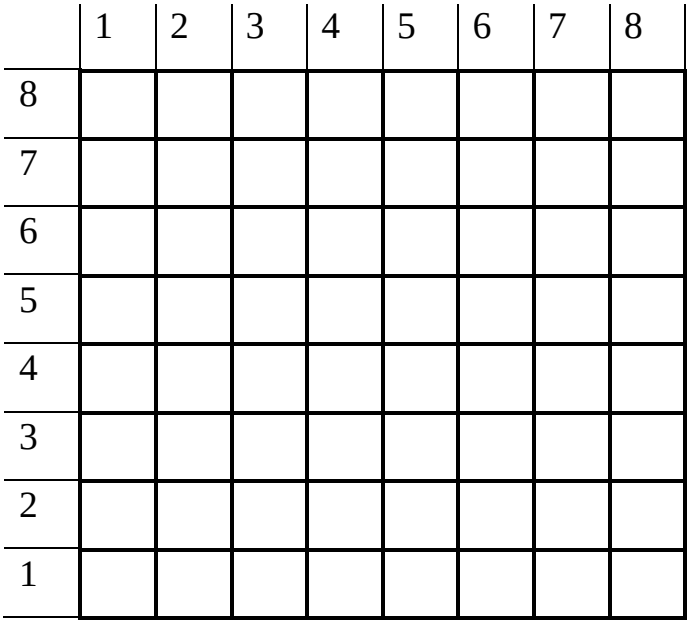

Рис. 3.7. Ілюстрація до задачі 3.4

З вище сказаного слідує, що задачу можна переформулювати наступним чином: вибрати 8 четвірок чисел *(X, Y, U,V)* таких, щоб кожне з цих чисел було з відповідного списку *(Dx, Dy, Du, Dv)* таким чином, щоб кожне число вибиралося з відповідного списку не більше одного разу.

Розв'язування полягає у наступному: вибрати першу четвірку чисел, якою характеризується позиція першого ферзя, вилучити кожне з цих чисел із відповідного списку, потім повторити цю послідовність дій ще 7 разів. Оскільки на кожній горизонталі може розміщуватися лише один ферзь, відповіддю може бути список із восьми чисел *Ly*, в якому перше число – це номер горизонталі для першої вертикалі, друге число – номер горизонталі для другої вертикалі і т.д.

Програма мовою Пролог матиме наступний вигляд:

речення для вилучення елементу зі списку:

вилучити(H,[H|T],T).

вилучити(H,[H1|T1],[H1|T2]):-вилучитиl(H,T1,T2).

речення для друку списку:

друк([]).

друк([H|T]):-write(H," "),друк(T).

речення для рекурсивного вилучення четвірок чисел з відповідних списків:

розв'язування([],[],_,_,_).

розв'язування([Y|Ly],[X|Dx1],Dy,Du,Dv):-

вилучити(Y,Dy,Dy1),U=X-Y,

вилучити(U,Du,Du1),V=X+Y,

вилучити(V,Dv,Dv1),

розв'язування(Ly,Dx1,Dy1,Du1,Dv1).

речення, за яким викликається розв'язування для сформованих 4-х списків:

розв'язок(Ly):-розв'язування(Ly,

[1,2,3,4,5,6,7,8],

[1,2,3,4,5,6,7,8],

[-7,-6,-5,-4,-3,-2,-1,0,1,2,3,4,5,6,7],

[2,3,4,5,6,7,8,9,10,11,12,13,14,15,16]).

Якщо тепер сформулювати запит

розв'язок(Ly),друк(Ly)

отримаємо відповідь задачі.

РОЗДІЛ 4
ІНФОРМАЦІЙНЕ МОДЕЛЮВАННЯ У ВИВЧЕННІ ПРИРОДНИЧО-МАТЕМАТЧНИХ ДИСЦИПЛІН

4.1. Використання інформаційного моделювання при вивченні стохастики

4.1.1. Стохастика: практикоорієнтований підхід. Розгляд стохастичних моделей даних є важливою частиною навчання інформаційного моделювання у педагогічному вузі. З цими моделями студенти вже зустрічалися у курсі "Теорія ймовірностей і математична статистика". На нашу думку, найбільш педагогічно виваженим і строго науковим вбачається підхід до вивчення цього курсу у педагогічному університеті за підручником М.І.Жалдака, Н.М.Кузьміної, Г.О.Михаліна "Теорія ймовірностей і математична статистика" [110].

Випадкові явища, та необхідність оцінювання їх ймовірності трапляються як при вивченні явищ фізичних, хімічних або механічних, так і при вивченні явищ біологічних і навіть соціальних [42]. При розв'язуванні багатьох таких практичних задач потрібно використовувати стохастичні моделі даних. Тому, зважаючи на важливість і практикоорієнтованість тем, пов'язаних з стохастичними моделями даних, доцільно додатково розглянути їх у курсі "Математичне і комп'ютерне моделювання", спираючись на сформовані у студентів знання після вивчення курсу "Теорія ймовірностей та математична статистика". Оскільки головний акцент тут потрібно поставити саме на моделях даних і моделюванні, доцільно розглядати тільки ті теми з стохастики, що стосуються дискретних та неперервних стохастичних моделей даних. Такий підхід надає студентам можливість засвоїти всі етапи розв'язування задачі з використанням моделювання, а саме аналіз умови, виявлення об'єктів і зв'язків між ними, побудова відповідної умові задачі математичної моделі, вибір адекватного інструменту (ПЗ) для аналізу цієї

моделі (пошуку розв’язку), розв’язування з використанням обраного інструменту, аналіз отриманого результату у термінах задачі (предметної галузі).

Подання матеріалу базується на розгляді і розв’язуванні практикоорієнтованих задач з використанням ППЗ “Gran1”.

Перейдемо безпосередньо до розгляду стохастичних моделей даних.

Відповіді на більшість питань, що виникають в процесі діяльності людини, потребують збирання певних даних, пов’язаних з підрахунком чи вимірюванням кількісних характеристик якихось об’єктів, наприклад:

Чи правда, що великі дози вітаміну С зменшують частоту застудних захворювань?

Чи правда, що кількість бракованих лампочок в партії з 1-го млн. штук менша, ніж 3%?

Яка кількість жителів республіки підтримує політику президента?

Такі дані називають статистичними, а науку, де вивчаються питання про збирання, організацію і опрацювання статистичних даних, називають статистикою.

Розглянемо таку задачу.

Задача **4.1.** Новорічні гірлянди містять по 6 лампочок. Організація планує купити 10000 гірлянд. Для перевірки якості гірлянд вибрана партія з 50 штук і визначено кількість дефектних лампочок у кожній гірлянді:

0	1	0	0	1	3	1	0	0	1
0	0	1	0	0	0	1	3	0	1
1	0	2	0	0	0	0	2	0	0
0	0	1	1	0	1	0	1	0	0
2	0	0	0	2	0	0	0	1	2

Необхідно визначити, скільки гірлянд придатні до використання і скільки лампочок всього зіпсовано серед вибраних 50 гірлянд. Зробити

висновок про можливу кількість придатних гірлянд та розбитих лампочок у всій партії.

При розв’язуванні задач з статистики можна виділити наступні етапи:

– збирання даних;
– організація і опрацювання даних (знаходження необхідних числових характеристик, побудова певних графічних залежностей, перевірка статистичних гіпотез тощо).

4.1.2 Основні означення. Сукупність об’єктів, відібраних для дослідження, будемо називати *вибіркою*. В наведеній задачі для дослідження партії гірлянд було відібрано сукупність з 50 гірлянд, кожній з яких відповідає певне число зіпсованих лампочок в межах від 1 до 6. Сукупність цих чисел і утворюють вибірку в наведеній задачі.

В багатьох задачах загальна сукупність об’єктів, які потрібно дослідити, буває досить великою. Таку сукупність досліджуваних об'єктів називають генеральною. Генеральною сукупністю в розглядуваній задачі є 10000 гірлянд.

Проте слід зауважити, що існує також значна кількість задач, в яких генеральну сукупність визначити неможливо, наприклад якщо потрібно визначити співвідношення випадання герба чи цифри при підкиданні монети 1000 разів, то вибірка представлятиме собою 1000 підкидань, а генеральна сукупність не визначена.

Навіть якщо генеральна сукупність відома, все одно в більшості випадків досліджують не її, а вибірку, оскільки, повне дослідження генеральної сукупності є практично неможливим або неекономним (у випадку великих сукупностей або досліджень, пов’язаних з фізичним пошкодженням об’єктів, що досліджуються). Наприклад дослідження часу безвідмовної роботи пристроїв, якості упаковок приводить до пошкодження досліджуваних об’єктів, проводити повний перепис населення країни кожного року є економічно невигідним, і т. ін.

Надалі величини, що досліджуються, позначатимемо великими літерами X, Y, Z тощо. Спостережені значення величини, наприклад X, будемо позначати відповідними малими літерами: $x_1, x_2, ..., x_n$. *Ці значення називатимемо варіантами.*

Важливо відмітити, що покупець в задачі про гірлянди не може відповісти на поставлені питання точно (для цього потрібно було б перевірити всі 10000 гірлянд), але можна вважати, що 50 вибраних гірлянд є достатньо репрезентативною вибіркою всієї партії, а тому відповіді, отримані для цих 50 гірлянд, з певною мірою впевненості можна узагальнити на всю партію (більш детально ці питання будуть розглянуті у наступному параграфі).

Об'ємом вибірки називатимемо кількість елементів в ній.

В наведеній задачі об'єм вибірки – 50.

4.1.3. Початкова організація даних. Оскільки дані вже зібрані, за умовою задачі, наступним етапом розв'язування є організація даних. Вибірка буде значно наочнішою, якщо всі її елементи впорядковані, наприклад за зростанням, що надалі завжди припускається.

Впорядкована вибірка називається *варіаційним рядом*.

В наведеній задачі варіаційний ряд матиме вигляд 0, 0,..., 0, 1, 1, ..., 1, 2, 2, ..., 2, 3, 3, ..., 3, де 0 повторюється 30 разів, 1 – 13 разів, 2 – 5 разів, 3 – 2 рази. Оскільки одне і те саме значення варіант вибірки може зустрічатися кілька разів, доцільно подати вибірку у вигляді таблиці, де один стовпчик – різні значення варіант (позначені x_i), другий – абсолютні частоти їх появи (позначені n_i), тобто кількість відповідних варіант. Таку таблицю називають *частотною (варіаційною) таблицею* або *рядом розподілу* абсолютних частот на множині можливих значень досліджуваної величини. Доцільно розширити цю таблицю третім стовпчиком, в який записувати накопичену (кумулятивну) абсолютну частоту (сума абсолютних частот елементів до даного включно). В розглядуваній задачі частотна таблиця матиме вигляд:

x_i	n_i	$\sum n_i$
0	30	30
1	13	43
2	5	48
3	2	50

Якщо множина всіх значень спостережуваної величини X скінченна, причому складається з фіксованого набору значень x_1, x_2, ..., $x_к$, то відповідний розподіл частот називатимемо точковим або дискретним. В розглядуваній задачі кількість дефектних лампочок може набувати значення 0, 1, 2, 3, 4, 5, 6, таких значень скінченна кількість, тому тут маємо справу з дискретним розподілом абсолютних частот.

Зрозуміло, що навіть найпростіше опрацювання значного масиву даних вручну (наприклад створення частотної таблиці) є громіздким рутинним процесом, тому доцільно для таких дій використовувати відповідні комп'ютерні програми (наприклад електронні таблиці, або програми, спеціально призначені для опрацювання статистичних даних). Серед вітчизняних програм можна запропонувати програму Gran1, використання якої надає можливість опрацьовувати різні математичні об'єкти, в тому числі і статистичні вибірки. Дані для цієї програми можуть бути введені як з клавіатури, так і з текстового файлу. Детальний опис можливостей використання цієї програми для роботи з статистичними даними наведено в параграфі 2.5. Для розв'язування задачі, сформульованої вище, послідовність дій з використанням програми Gran1 може бути такою:

1) у вікні "Список об'єктів" встановити тип залежності *"Стат. вибірка"*;
2) скористатися пунктом меню *"Об'єкт/Створити…"* або командою *"Створити"* контекстного меню вікна *"Список об'єктів"*;

3) ввести дані у допоміжне вікно *"Дані для статистичної вибірки"* (як на Рис. 4.1) і натиснути кнопку *"Далі--->"*. Зауважимо, що у полі *„Тип даних"* допоміжного вікна в цій задачі вибрано значення *„Частоти"*, оскільки вже є частотна таблиця. Якщо така таблиця ще не побудована, як це буває в більшості експериментальних задач, потрібно вибирати значення *„Варіанти";*

4) звернутися до послуги *"Операції/Статистика/Частотна таблиця"* і з'ясувати, в скількох гірляндах немає жодної дефектної лампочки (30) і в скількох гірляндах є 1, 2 або 3 дефектні лампочки (13+5+2=20)

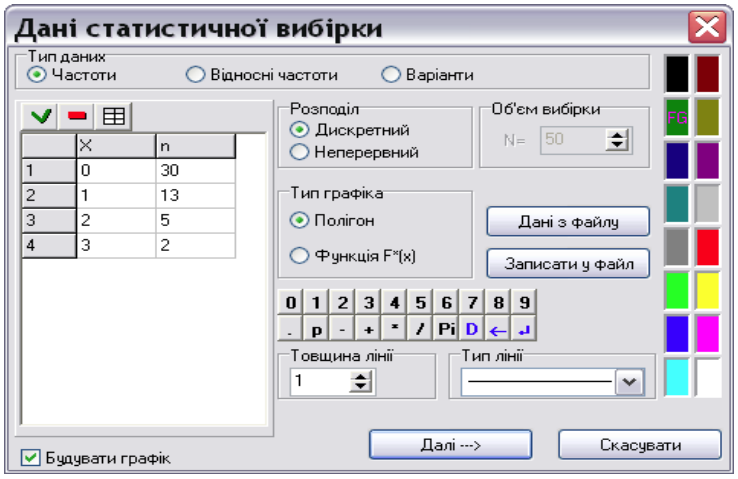

Рис. 4.1. Введені дані до Задачі 4.1.

Кількість дефектних лампочок у вибірці буде 13+5*2+2*3=29.

Поширюючи результати дослідження вибірки на всю генеральну сукупність, можна сказати, що у партії з 10000 гірлянд слід очікувати близько 30*10000/50=6000 гірлянд без дефектів та близько 20*10000/50=4000 гірлянд з дефектами.

Зауваження. *Всі створені за програмою Gran1 вибірки бажано записувати на диск за допомогою послуги "Файл/Записати", якщо вони будуть використовуватись при подальшому вивченні.*

Часто зручніше замість частоти n_i появи значення x_i розглядати відносну частоту $P_n^*(x_i) = \frac{n_i}{n}$, де n - об'єм вибірки.

Таблицю виду

Таблиця 4.1.

Ряд розподілу відносних частот

x_i	x_1	x_2	...	x_k
$P_n^*(x_i)$	$P_n^*(x_1)$	$P_n^*(x_2)$	...	$P_n^*(x_k)$

називають *рядом розподілу статистичних ймовірностей (відносних частот)* на множині можливих значень досліджуваної величини. Очевидно $P_n^*(x_i) \geq 0$ і $\sum_{i=1}^{k} P_n^*(x_i) = 1$.

Наведемо ряд відносних частот для розглянутої вище задачі:

x_i	0	1	2	3
$P_n^*(x_i)$	0.60	0.26	0.10	0.04

Нехай A – деяка підмножина множини $\{x_1, x_2, ..., x_к\}$. Якщо потрібно визначити відносну частоту попадання в множину A, то можна скористатися формулою $\sum_{x_i \in A} P_n^*(x_i)$. Надалі відносну частоту попадання в множину A позначатимемо $P_n^*(A)$.

Очевидно:

$0 \leq P_n^*(A) \leq 1$;

$P_n^*(\{x_1,..., x_k\}) = 1$;

$P_n^*(\varnothing) = 0$;

якщо $A \subset B$, *то* $P_n^*(A) \leq P_n^*(B)$;

якщо $A \cap B = \varnothing$, *то* $P_n^*(A \cup B) = P_n^*(A) + P_n^*(B)$;

Задача 4.2. Нехай при 100 підкиданнях кубика отримано такі відносні частоти появи граней:

x_i	1	2	3	4	5	6
$P_n^*(x_i)$	0.08	0.10	0.27	0.18	0.25	0.12

Якщо $A=\{2,4,6\}$, $B=\{4,5,6\}$, тоді

$P^*_{100}(A)=0.10+0.18+0.12=0.40,$

$P^*_{100}(B)=0.18+0.25+0.12=0.55,$

$P^*_{100}(\{1,2\})=0.08+0.10=0.18,$

$P^*_{100}(\{5,6\})=0.25+0.12=0.37,$

$P^*_{100}(\{3,4,5\})=0.27+0.18+0.25=0.70$

4.1.4. Способи збирання статистичних даних. Необхідно пояснити студентам важливість питання правильного формування вибірки і сформувати в них базові компетентності щодо отримання таких вибірок.

Для того, щоб за вибіркою можна було з достатньою мірою впевненості досить правильно судити про досліджувану величину, треба, щоб вибірка була репрезентативною. В більшості випадків для репрезентативності вибірки необхідно, щоб її об'єкти отримувалися випадковим чином, причому кожен об'єкт, який вибирається для дослідження, повинен мати однакову з іншими можливість потрапити до вибірки, а кількість таких об'єктів має бути достатньо великою.

Тому надалі будемо вважати, що всі вибірки, наведені в цьому параграфі, отримані випадковим чином.

Кожна конкретна задача має свої особливості стосовно формування вибірки. Наприклад, статистичне дослідження думки жителів міста стосовно певної проблеми центру важко вважати об'єктивним, якщо опитувалися випадковим чином жителі не всього міста, а тільки на одній вулиці (наприклад біля університету), або на різних вулицях, але в один проміжок часу.

Другою важливою вимогою є достатній об'єм вибірки. Взагалі кажучи, збільшення об'єму вибірки підвищує вірогідність результатів статистичного аналізу вибірки, але додає труднощів і робить дослідження більш витратним.

Доцільно навести студентам кілька способів отримання вибірок для різноманітних практичних задач.

Спосіб 1. Перегляд записів.

Проблема. Чи зменшилась кількість пригод на дорогах після прийняття нових правил дорожнього руху?

Для відповіді на це питання переглядають записи про дорожні пригоди до і після прийняття нових правил руху.

Спосіб 2. Проведення експерименту.

Проблема. Чи підвищує нова методика навчання рівень знань з математики?

Для відповіді на питання необхідно провести експеримент. Вибирають кілька класів, половина з яких навчається за новою методикою, а інша половина за традиційною. Регулярно проводяться контрольні роботи і аналізуються отримані оцінки.

Спосіб 3. Аналіз вибірки.

Проблема. Скільки голосів на виборах отримає певний кандидат?

Дані опитування певної кількості жителів республіки (вибірка) аналізуються і результати узагальнюються на все населення.

4.1.5. Види розподілу. Досліджуваний в задачі 4.1 розподіл частот дискретний. Але існує ще один вид розподілу — неперервний, про який йтиметься далі.

Якщо спостережувана величина X може набувати будь яких значень із деякого відрізка $[a, b]$, спостережених значень досить багато і вони досить густо розсіяні між a і b (наприклад у випадку, коли дані є результатами якихось вимірювань), тоді зберігання кожного окремого значення x_i втрачає смисл. В такому разі зручно розглянути деякий проміжок $[a, b)$, який містить всі спостережені значення (як правило беруть проміжок $[x_{min}-\varepsilon, x_{max}+\varepsilon)$), де ε — деяке досить мале число, поділити його на деяке число проміжків $[a_0, a_1)$,

$[a_1, a_2)$, ..., $[a_{m-1}, a_m)$ однакової довжини, при цьому $a_0 \leq x_{min}$, $a_m > x_{max}$, $a_i = a_{i-1} + h$, $h = (a_m - a_0)/m$, і подати результати спостережень таблицею виду

Таблиця 4.3.

Інтервальний розподіл відносних частот

$[a_{i-1}, a_i)$	$[a_0, a_1)$	$[a_1, a_2)$	...	$[a_{m-1}, a_m)$
$P_n^*([a_{i-1}, a_i))$	$P_n^*([a_0, a_1))$	$P_n^*([a_1, a_2))$	...	$P_n^*([a_{m-1}, a_m))$

де $P_n^*([a_{i-1}, a_i))$, $i=1, 2, \ldots, m$ – відносні частоти (статистичні ймовірності) попадання значень спостережуваної величини в інтервали $[a_{i-1}, a_i)$, тобто кількість значень x_k, що знаходяться в межах $[a_{i-1}, a_i)$, поділена на загальну кількість спостережених значень. Таку таблицю називають інтервальним (неперервним) розподілом статистичних ймовірностей (відносних частот) на множині значень досліджуваної величини. Середину кожного інтервалу називають міткою інтервалу.

Для визначення кількості інтервалів, на які слід поділити проміжок $[a_0, a_m)$ можна використати формулу Стерджеса: $m = 1 + 3.322 * lg(n)$, де n – об'єм вибірки.

Для ілюстрації наведених вище теоретичних положень щодо опрацювання неперервних розподілів, доцільно розв'язати з студентами наступну задачу.

Задача 4.3. Були зважені виловлені із ставка 50 коропів. Отримано такі результати:

0.62	0.83	0.78	0.77	0.88	0.79	0.84	1.07	0.81	0.88
1.10	1.01	1.03	0.87	1.13	0.94	0.94	0.68	0.97	0.71
0.80	0.73	0.86	0.92	0.65	0.89	0.97	0.92	0.85	1.11
1.08	1.14	1.00	1.05	0.95	1.01	0.91	0.75	0.68	1.19
0.73	0.85	0.66	0.87	0.81	0.77	0.83	1.05	1.01	0.80

Скільки коропів мають вагу, меншу за 1 кг? За 2 кг?

В даному прикладі множину можливих значень слід поділити на інтервали. При використанні програми **Gran1** потрібно створити новий об'єкт – "статистична вибірка", заповнити допоміжне вікно *“Дані статистичної вибірки”* як на Рис. 4.2 і „натиснути” *кнопку “Далі→”*.

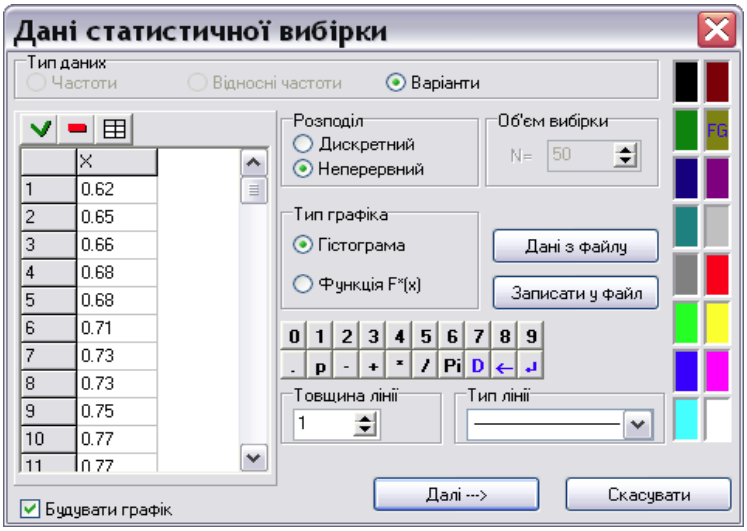

Рис. 4.2. Введення даних до Задачі 4.3.

У додатковому вікні потрібно вказати відрізок задання вибірки та кількість інтервалів. В наведеному випадку доцільно натиснути кнопку *“Відрізок за вибіркою”* для визначення відрізка задання вибірки і кнопку *“Застосувати формулу Стерджеса”* для визначення кількості інтервалів (Рис. 4.3).

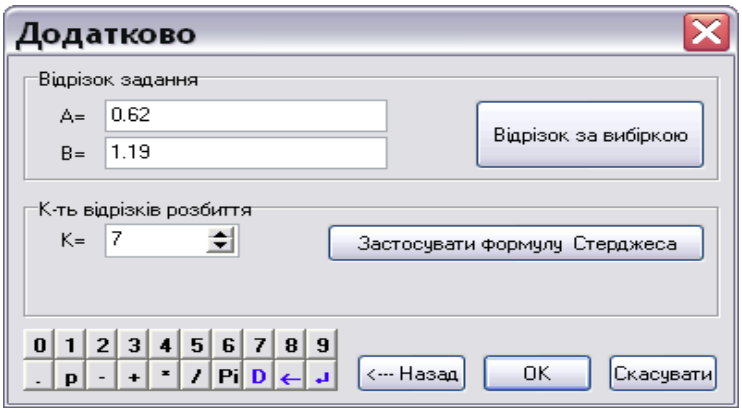

Рис. 4.3. Введення додаткових даних до Задачі 4.3.

Після цього за програмою буде поділено відрізок задання вибірки, вказаний користувачем, на визначену ним кількість частин і підраховано кількість елементів, що попадає у кожен інтервал.

За допомогою послуги *“Операції/Статистика/Частотна таблиця”*, можна отримати таблицю, в якій записано значення кінців інтервалів, на які поділено відрізок, і, відповідно, абсолютні частоти, накопичені абсолютні частоти, відносні частоти та накопичені відносні частоти потрапляння елементів вибірки в кожен інтервал. (Рис. 4.4)

Відрізок	n	Накопич. n	Pn*	Накопич. Pn*
0.62 - 0.7014	5	5	0.1	0.1
0.7014 - 0.7829	7	12	0.14	0.24
0.7829 - 0.8643	11	23	0.22	0.46
0.8643 - 0.9457	10	33	0.2	0.66
0.9457 - 1.027	7	40	0.14	0.8
1.027 - 1.109	6	46	0.12	0.92
1.109 - 1.19	4	50	0.08	1

Рис. 4.4. Частотна таблиця до Задачі 4.3.

Для відповіді на питання задачі достатньо за допомогою команди *“Об’єкт/Змінити”* звернутися до варіаційного ряду, з якого відразу видно, що коропів, легших за 1 кг, буде 36 (Рис. 4.5) і кожен із 50 зловлених коропів має масу, меншу ніж 2кг (Рис. 4.6).

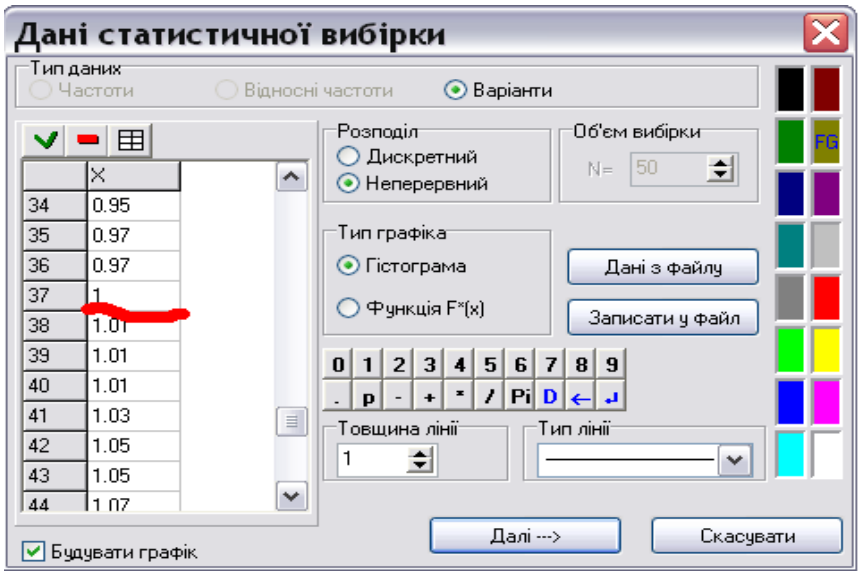

Рис. 4.5. Отримання 1-ї відповіді до Задачі 4.3.

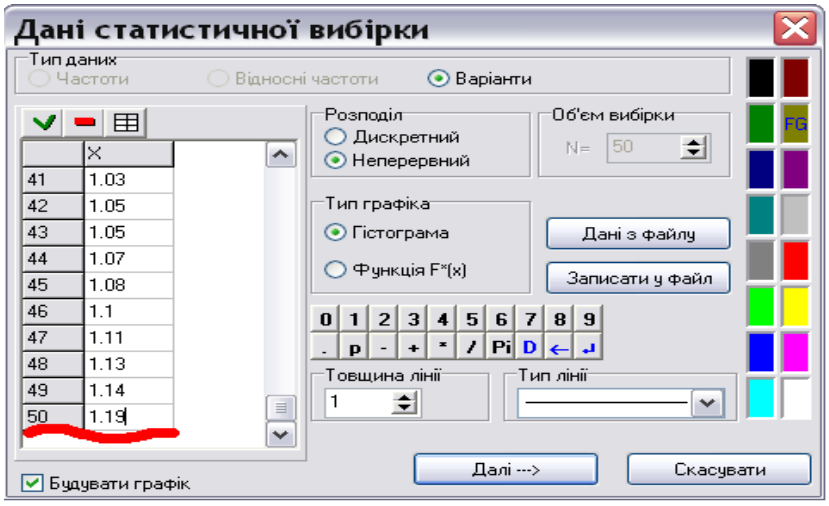

Рис. 4.6. Отримання 2-ї відповіді до Задачі 4.3.

Для підвищення рівня компетентностей студентів щодо наведених вище теоретичних положень доцільно запропонувати їм розв'язати наступні задачі.

Задача 4.4. Є 40 анкет, заповнених даними про десятьох учнів з кожного з 4-х класів. Навмання вибирають 30 анкет і записують номер класу. Необхідно скласти частотну таблицю випадання номерів класів на основі 30 анкет, що вибрано.

Зауваження. *У цій задачі розподіл частот дискретний (існує тільки 4 класи), що і потрібно зазначити при заповненні допоміжного вікна* ***"Дані статистичної вибірки"*** *при використанні програми* ***Gran1.***

Задача 4.5. Гральний кубик підкидають 50 разів. Скласти частотну таблицю випадання кожної із граней кубика.

Задача 4.6. Виміряти довжину долоні 30 учнів і скласти частотну таблицю довжин.

Задача 4.7. Виміряти зріст 30 учнів і скласти частотну таблицю зростів.

Задача 4.8. Виміряти вагу 40 учнів і скласти відповідну частотну таблицю, визначивши довжину інтервалу самостійно.

Зауваження. *У задачах 4.6-4.8 досліджувана величина може набувати довільних значень з певного діапазону, тому розподіл відносних частот – неперервний.*

Задача 4.9. Перевіряється якість фруктів перед продажем. Для цього з 1000 ящиків вибирається навмання 50 ящиків і підраховується кількість ушкоджених плодів в кожному ящику. Отримано такі дані:

0	3	3	6	1	2	2	4	2	1
2	4	0	2	4	1	3	2	4	3
2	4	0	4	0	2	1	4	1	0
3	1	2	5	2	3	5	0	2	4
2	1	3	1	3	4	3	1	3	2

Питання:

1) скільки у вибірці ящиків вищого ґатунку (0 або 1 ушкоджених плодів);

2) скільки у вибірці ящиків з ушкодженими плодами;

3) підрахувати кількість ящиків у вибірці у відповідності до ґатунків:

- вищого ґатунку: 0-1 ушкоджених плодів;
- стандартного ґатунку: 2-4 ушкоджених плодів;
- неякісні: 5 і більше ушкоджених плодів.

4) визначити наближену оцінку: скільки у всій партії із 1000 ящиків ящиків кожного ґатунку.

Задача 4.10. Коректор вибрав 40 сторінок з 400-сторінкового рукопису книги і перевірив ці сторінки на кількість помилок. Кількість помилок на кожній сторінці така:

0	1	1	3	0	0	1	1
3	2	0	2	1	0	0	2
0	0	0	2	1	0	2	0
0	1	0	1	0	2	3	0
1	0	0	2	1	0	1	0

Потрібно побудувати частотну таблицю для кількості помилок на сторінці.

Для досліджуваної книги розрахувати наближену кількість:

- сторінок без помилок;
- сторінок з трьома і більше помилками;
- сторінок з двома і менше помилками.

Задача 4.11. Контейнер містить 5000 піддонів з яйцями (у кожному піддоні 20 яєць). На перевірку взято 100 піддонів і підраховано кількість розбитих яєць. Отримана вибірка:

0	0	0	1	0	20	0	0	0	20
0	1	2	1	0	0	3	20	1	0
1	0	1	1	0	0	1	0	0	0
0	0	20	0	2	3	1	2	0	0
2	0	0	3	0	0	2	0	3	0
1	1	0	0	0	3	3	2	0	0
0	0	0	1	0	0	2	2	0	20
0	0	1	2	2	20	20	1	0	0
0	0	0	1	1	0	0	20	0	1
1	0	0	0	20	1	2	0	0	0

Потрібно:

- побудувати частотну таблицю для кількості розбитих яєць;
- розрахувати:
- кількість піддонів без розбитих яєць;
- кількість піддонів з усіма розбитими яйцями;
- кількість піддонів з більш ніж одним розбитим яйцем;
- прогнозовану загальну кількість цілих яєць у контейнері.

Задача 4.12. За комп’ютерною програмою згенеровано 100 псевдовипадкових чисел у проміжку від 0 до 1 (з трьома знаками після коми):

0.184 0.398 0.941 0.953 0.357 0.690 0.035 0.116
0.421 0.687 0.540 0.415 0.966 0.503 0.809 0.346
0.930 0.512 0.745 0.633 0.454 0.631 0.970 0.621
0.581 0.808 0.947 0.344 0.537 0.322 0.435 0.614
0.669 0.098 0.105 0.577 0.880 0.186 0.257 0.559
0.474 0.919 0.985 0.227 0.847 0.451 0.495 0.602
0.273 0.094 0.438 0.264 0.722 0.946 0.410 0.046
0.918 0.751 0.278 0.796 0.345 0.810 0.736 0.524
0.472 0.965 0.831 0.684 0.486 0.005 0.427 0.490
0.472 0.290 0.487 0.736 0.081 0.569 0.339 0.738
0.226 0.794 0.299 0.181 0.244 0.606 0.118 0.328
0.774 0.096 0.168 0.691 0.332 0.636 0.031 0.505
0.712 0.243 0.389 0.177

Потрібно:

– побудувати інтервальний розподіл відносних частот з довжиною відрізка поділу 0.1;
– зробити припущення про рівномірність розподілу псевдовипадкових чисел на відрізку *[0, 1]*.

Зауваження. *В задачі наведено готову статистичну вибірку, але можна описати програму однією з мов програмування для отримання текстового файлу із значеннями випадкових чисел. В цьому випадку можна розглянути вибірку більшого об'єму, наприклад 1000, для отримання більш обґрунтованого висновку про рівномірність розподілу псевдовипадкових чисел на відрізку [0, 1].*

Задача 4.13. Певну компанію цікавить термін придатності деяких компонентів приладів. 30 компонентів тестували до їх виходу з ладу і записували (у годинах) час безвідмовного функціонування цих компонентів. Отримали наступні дані:

1,2	21,0	34,7	13,2	3,6	14,7
31,0	17,1	22,1	16,4	21,2	15,2
11,3	6,8	2,7	31,2	9,0	6,8
23,7	16,3	30,0	28,6	19,0	29,0
4,4	44,3	18,5	5,3	5,5	10,0

Потрібно:

- побудувати інтервальний розподіл відносних частот на множині значень досліджуваної величини з довжиною інтервалу 10 год.;
- компанія стверджує, що не менше 90% компонентів вийде з ладу не раніше, ніж через 10 год. Наскільки вірогідне таке твердження?
- фірма замовила 150 компонентів; необхідно розрахувати кількість компонентів, не вийдуть з ладу протягом 20 годин.

4.1.6. Графічне подання статистичних даних. (Гістограми, полігони, емпіричні функції розподілу частот). Доцільно нагадати студентам, спираючись на їх попередні знання та досвід, що графічне подання даних часто виявляється зручнішим і більш наглядним за табличну форму. Для графічного зображення точкових (дискретних) розподілів використовують полігон відносних частот. Для його побудови на площині *xOy* необхідно нанести точки з координатами $(x_i, P_n^*(x_i))$ і потім з'єднати їх ламаною лінією. Для побудови полігона за допомогою програми Gran1 треба в процесі задання вибірки встановити тип графіка *"Полігон"*, а потім скористатися послугою *"Побудувати"* з меню *"Графік"*. На Рис. 4.7 зображено побудований за програмою полігон до задачі 4.1.

Інколи виникає потреба порівняти кілька вибірок, як, наприклад, в наступній задачі.

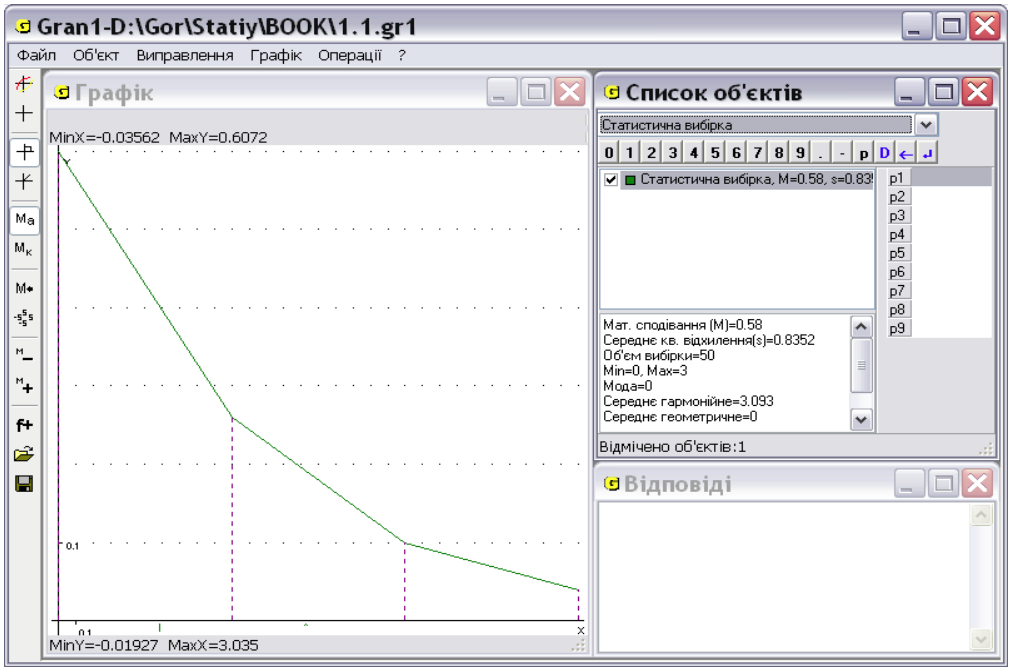

Рис.4.7. Полігон відносних частот до Задачі 4.1.

Задача 4.14. Є такі дані щодо продажу кольорових і чорно-білих телевізорів:

Рік	80	81	82	83	84	85	86	87
Чорно-білі	32	28	15	22	14	7	4	2
Кольорові	23	35	51	58	53	43	39	29

Необхідно визначити, які телевізори продавались краще після 1983 року.

Якщо на спільній системі координат побудувати полігони для обох вибірок цієї задачі, можна побачити, що для даних після 1983 р. полігон, що відповідає продажам кольорових телевізорів, знаходиться вище, ніж полігон, що відповідає продажам чорно-білих телевізорів, тому можна зробити висновок про кращий продаж після 1983 року кольорових телевізорів.

Для розв'язування цієї задачі за допомогою програми Gran1 спочатку визначаємо обидві вибірки, вказуючи для кожної, що розподіл частот є точковим (тобто дискретним), а тип графіка – полігон. Після цього впевнюємося, що обидві вибірки відмічені у вікні *"Список об'єктів"* і звертаємося до послуги *"Графік/Побудувати"*. (див. Рис.4.8).

Для графічного подання інтервальних розподілів використовують гістограму відносних частот. Гістограма – це графік кусково-сталої функції $f_n^*(x)$ (щільності розподілу статистичних ймовірностей), яка дорівнює нулеві за межами проміжка $[a_0,a_m)$ і $\frac{P_n^*([a_{i-1},a_i))}{h}=\frac{P_n^*([a_{i-1},a_i))}{a_i-a_{i-1}}$ на проміжку

$[a_{i-1},a_i)$, $i=1, 2,..., k$. тобто

$$f_n^*(x)=\begin{cases}\dfrac{P_n^*([a_{i-1},a_i))}{h}, коли\ x\in[a_{i-1},a_i), i\in\overline{1,m}\\ 0, коли\ x\notin[a_0,a_m)\end{cases}$$

Оскільки $\sum_{i=1}^{k}P_n^*([a_{i-1},a_i))=1$, то площа під гістограмою дорівнює 1, як сума площ виду $\frac{P_n^*([a_{i-1},a_i))}{h}(a_i-a_{i-1})=P_n^*([a_{i-1},a_i))$. (див. [92, 93, 97, 98, 100, 103, 109, 110, 330-334]).

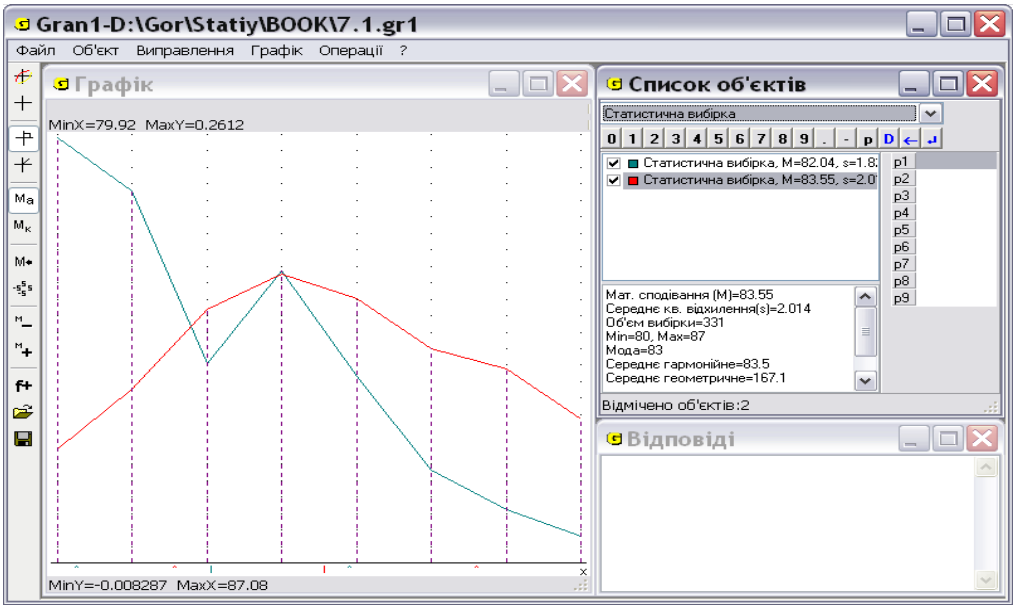

Рис. 4.8. Полігони відносних частот до Задачі 4.14.

Для побудови гістограми за допомогою програми Gran1 треба в процесі задання вибірки встановити тип графіка *"Гістограма"*, а потім скористатися послугою *"Побудувати"* з меню *"Графік"*. На Рис. 4.9 зображена побудована за програмою Gran1 гістограма до задачі 4.3 (про коропів).

Якщо замість відносних частот взяти накопичені відносні частоти, отримаємо графік *функції* $F_n^*(x)$ *розподілу відносних частот (статистичних ймовірностей).*

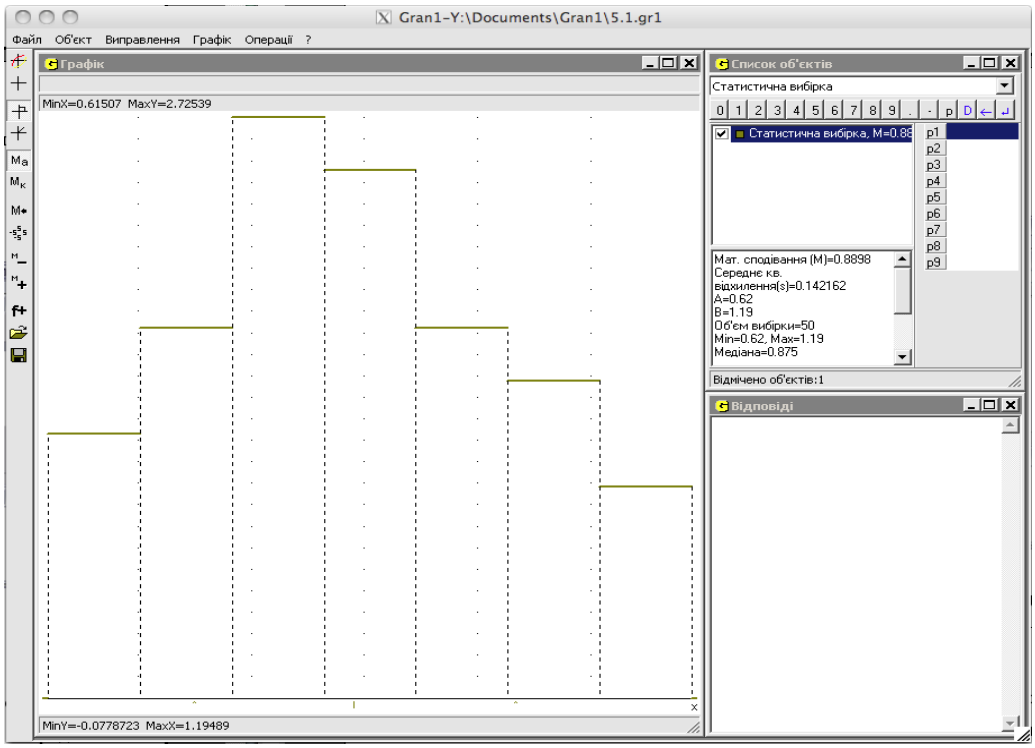

Рис. 4.9. Гістограма до Задачі 4.3.

У випадку інтервального розподілу частот (щільність розподілу відносних частот на кожному інтервалі вважається сталою), тобто коли досліджувана величина X є неперервною, вважають, що всередині інтервалів накопичені частоти зростають не стрибкоподібно, а плавно, і зображують кумуляту не кусочно-сталою, а кусочно-лінійною. Для дискретного розподілу так робити, зрозуміло, не можна, тому графік емпіричної функції дискретного розподілу частот буде кусочно-сталим (див. [110]).

Функцією розподілу відносних частот (статистичних ймовірностей) називають функцію $F_n^*(x)$, значення якої для кожного значення x дорівнює $P_n^*((-\infty, x))$. При дискретному розподілі відносних частот це значення дорівнює кількості варіант n_x, які менші за x, поділеній на загальну кількість варіант n, тобто

$$F_n^*(x) = \frac{n_x}{n} = \sum_{x_i < x} P_n^*(x_i)$$

Властивості функції розподілу:

1. $0 \le F_n^*(x) \le 1$;

2. $F_n^*(x)$ – неспадна функція;

3. $F_n^*(x) = 0$ для $x \le a$;

4. $F_n^*(x) = 1$ для $x > b$.

a – ліва межа відрізка, b – права межа відрізка, на якому знаходяться значення досліджуваної величини.

Для побудови функції розподілу відносних частот за допомогою програми Gran1 треба звернутися до послуги *"Об'єкт/Змінити"* і встановити тип графіка *"Функція F*(x)"*, а потім скористатися послугою *"Побудувати"* з меню *"Графік"*.

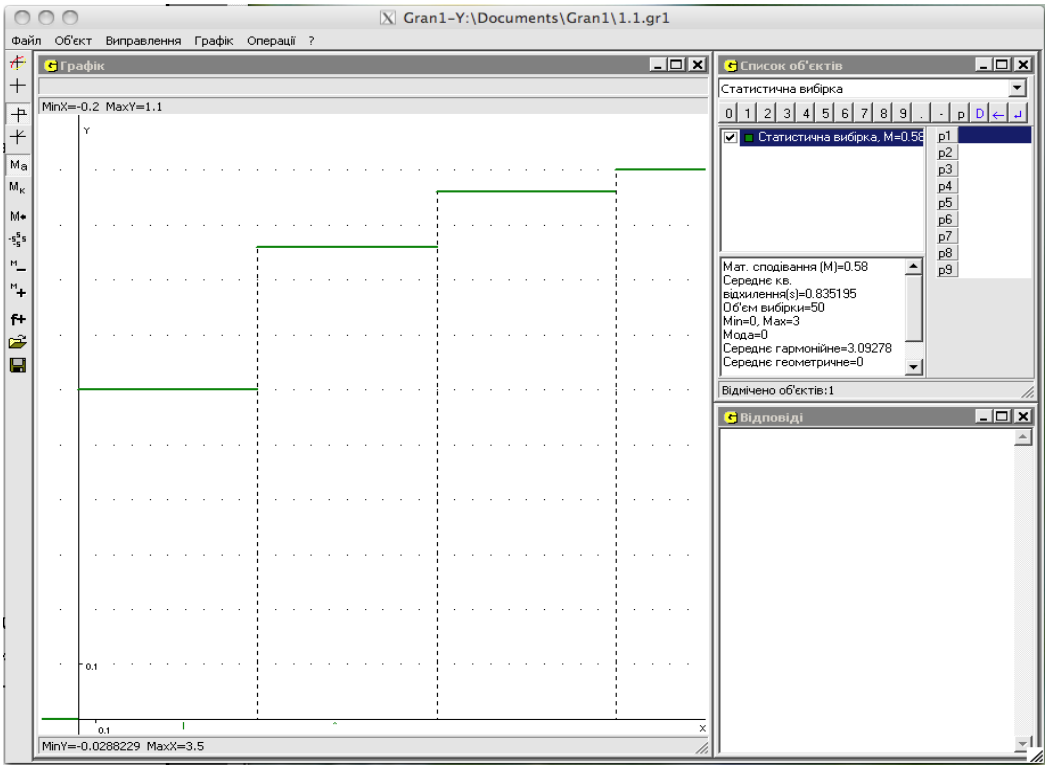

Рис. 4.10. Функція розподілу до задачі 4.1.

На Рис. 4.10 зображено функцію розподілу до задачі 4.1 (про гірлянди), в якій досліджуваний розподіл відносних частот є дискретним.

На Рис. 4.11 зображено функцію розподілу до задачі 4.3 (про коропи), в якій досліджуваний розподіл відносних частот є неперервним (інтервальним).

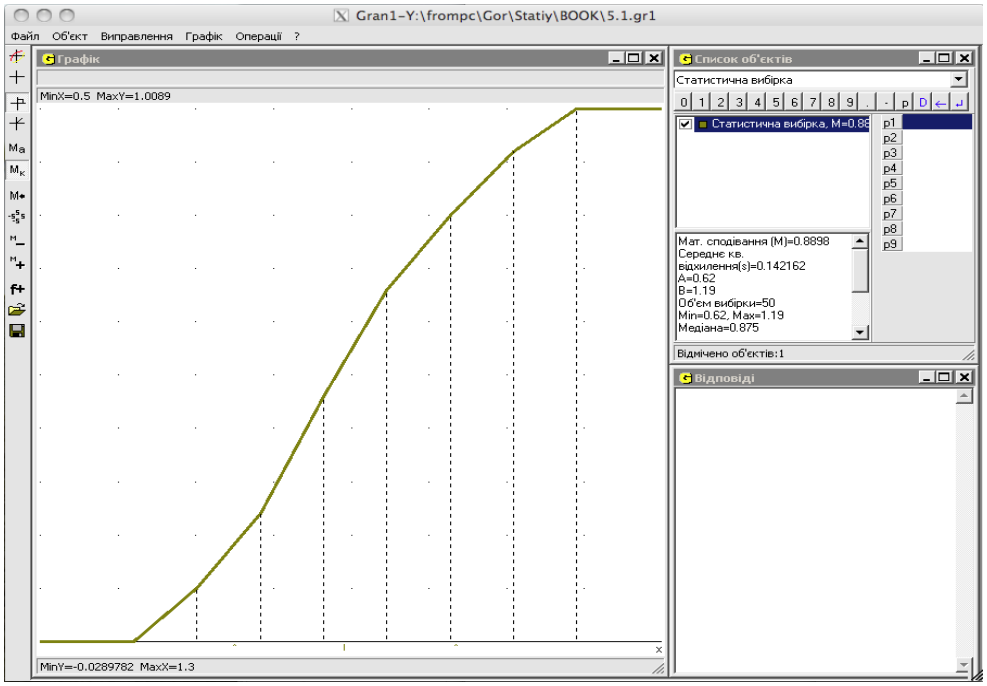

Рис. 4.11. Функція розподілу до задачі 4.3.

Для підвищення рівня компетентностей студентів щодо теоретичних положень про графічне подання статистичних даних доцільно запропонувати їм розв’язати наступні задачі.

Задача 4.15. Для порівняння двох добрив при вирощуванні соняхів вибрали двісті ділянок однакової площі і на перших ста з них застосували перший вид добрив, а на інших ста ділянках – другий вид добрив. Обчислили кількість соняшникової олії, отриманої з урожаю з кожної ділянки, і отримали наступні дані:

Мітка класу	600	800	1000	1200	1400	1600	1800
1-ше добриво	3	9	19	39	21	9	0
2-ге добриво	3	2	11	32	30	20	2

Необхідно:

За допомогою гістограм, побудованих в одній системі координат, відповісти, при використанні якого з добрив можна отримати більшу кількість олії з однієї ділянки.

За допомогою функцій розподілу частот, побудованих в одній системі координат, порівняти для кожного виду добрив процент ділянок, де кількість олії з урожаю більша за 1200 л.

Задача 4.16. Побудувати гістограми для наступних даних про курців:

Дані за 1985 р.

К-ть сигарет	1-5	6-10	11-15	16-20	21-25	26-30	31-35
Жінки	40	15	3	1	0	0	0
Чоловіки	12	16	25	18	8	5	2

Дані за 1992 р.

К-ть сигарет	1-5	6-10	11-15	16-20	21-25	26-30	31-35
Жінки	10	13	19	11	4	2	1
Чоловіки	17	21	16	12	5	1	1

Чи курили чоловіки більше, ніж жінки, у 1985 р.?

Чи це справджується для 1992 р.?

Які зміни за 7 років помітно в палінні жінок?

Які зміни за 7 років помітно в палінні чоловіків?

4.1.7. Міри центральної тенденції вибірок. Для розв’язування багатьох практичних задач потрібно визначити деякі числа, що відображають найбільш характерні риси розподілу статистичних ймовірностей на множині значень величини, що досліджується. Одне з таких чисел називають середнім або мірою центральної тенденції. Міра центральної тенденції найчастіше може бути виражена середнім арифметичним спостережених значень x_{icn} чи модою.

Середнє арифметичне спостережених значень x_{icn} отримується додаванням всіх елементів вибірки і діленням результату на кількість елементів. Будемо позначати середнє арифметичне через M_n*:

$$M_n^* = \frac{\sum_{i=1}^{n} x_{icn}}{\text{n}} = \sum_{i=1}^{m} x_i P_n^*(\{x_i\})$$

Де x_{icn}, $i=1, 2, \ldots, n$ — спостережені значення досліджуваної величини, які можуть повторюватися, x_i, $i=1, 2, \ldots, m$ — можливі значення досліджуваної величини, які не повторюються, $P_n^*(\{x_i\})$ $i=1, 2, \ldots, m$ — відносні частоти (статистичні ймовірності) появи можливих значень x_i у вибірці.

Мода – значення, яке частіше від інших зустрічається серед спостережених у даних.

На графіку мода – це x-координата найвищої точки на полігоні, або точки, в якій щільність розподілу сягає максимуму (при неперервному розподілі).

Інколи міру центральної тенденції характеризують середнім гармонійним:

$$Серед.гарм. = \frac{1}{\frac{1}{n}\sum_{i=1}^{n} \frac{1}{x_i}}$$

Така характеристика може бути більш зручною у випадку, наприклад, визначення середньої швидкості руху деякого тіла вздовж прямої, якщо відомо швидкості, з якими тіло проходило кожну одиницю шляху.

Згадані вище числові характеристики (середнє арифметичне, мода, медіана, середнє гармонійне), та деякі інші визначаються за програмою Gran1 автоматично після створення або редагування вибірки і відображуються в нижній частині вікна *“Список об’єктів”*.

Для формування практичних вмінь і навичок щодо знаходження середнього арифметичного і моди, а також щодо з'ясування такого важливого питання, як вибір однієї з цих характеристик для більш точного

репрезентування досліджуваних даних доцільно розв'язати зі студентами наступну задачу.

Задача 4.17. Для кожного із запропонованих наборів даних побудувати полігон та визначити середнє арифметичне і моду:

Кількість дефектів в кожному з 100 мотків пряжі

К-ть дефектів	0	1	2	3	4	5	6
Частота	22	46	18	8	3	2	1

Кількість розбитих банок в ящику (12 банок на ящик) в партії з 50 ящиків

К-ть банок	0	1	2	3	4-11	12
Частота	36	8	2	1	0	3

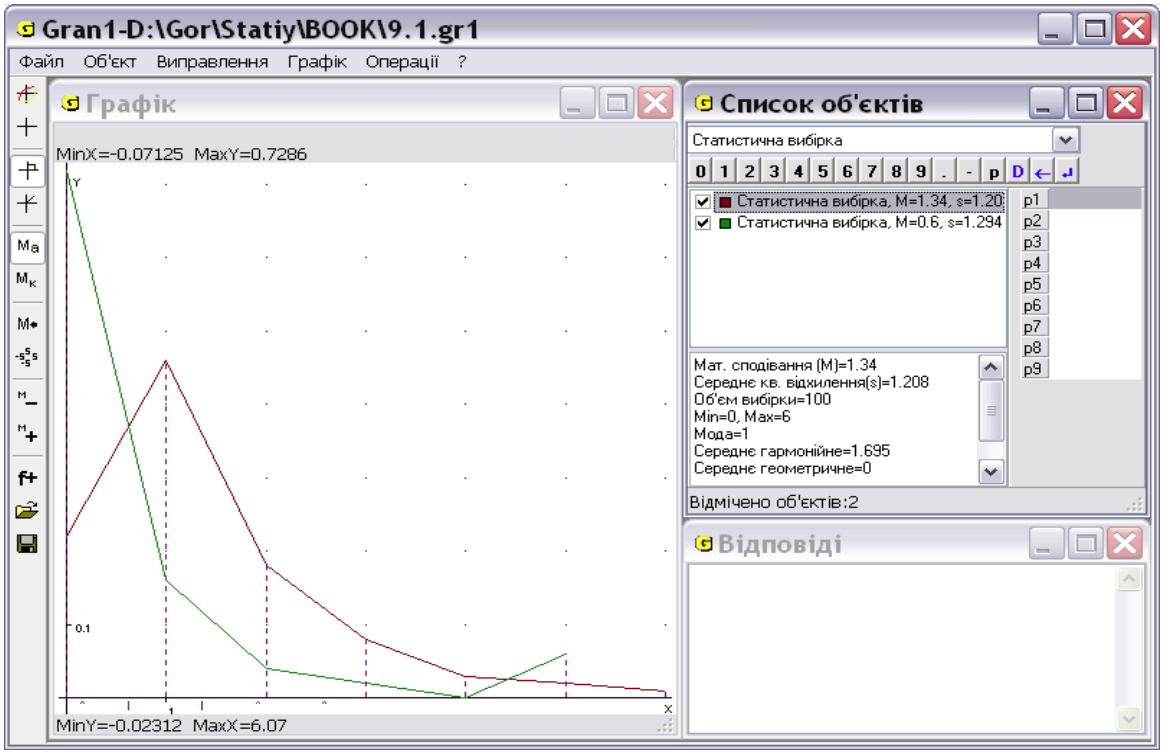

Рис. 4.12. Полігони відносних частот до Задачі 4.17.

Скористаємося програмою Gran1 і введемо обидві вибірки. За програмою обчислюються потрібні числові характеристики для кожної з

вибірок, а саме для 1-ї вибірки *M=1.34*, *Мода=1*; для 2-ї вибірки *M=1.02*, *Мода=0*. Значення середнього арифметичного не співпадають із значенням моди для наведених даних. Виникає питання, яка з цих числових характеристик більш точно репрезентує досліджувані дані? Тут важливо відмітити, що іноді неуважне використання різних мір центральної тенденції може ввести в оману, оскільки вони можуть досить суттєво різнитись між собою. В таких випадках графічне подання результатів аналізу вибірки є вирішальним для вибору міри центральної тенденції. Спостерігаючи полігони, можна помітити, що одна з частот значно перевищує інші (Рис. 4.12) і тому в такому випадку більш точною характеристикою є мода.

4.1.8. Розподіли частот, що часто зустрічаються. Поняття дисперсії. Спостереження за графічними поданнями вибірок при розв'язуванні багатьох практичних задач дають підстави розглядати кілька класів розподілів статистичних ймовірностей, що найчастіше зустрічаються.

1. Рівномірний розподіл (наприклад розподіл частот випадання кожної з шести граней однорідного грального кубика).

Доцільно запропонувати студентам файл з даними про випадання граней грального кубика, підкинутого 100 разів, побудувати полігон відносних частот, відмітити його характерну форму, яка наближається до прямої (Рис. 4.13).

2. Ванноподібний розподіл.

Задача 4.18. Метеорологи вимірюють процент неба, вкритого хмарами. Частіше буває, що небо або все в хмарах, або майже ясне. Частково хмарні дні зустрічаються рідше.

Дані спостережень до Задачі 4.18

Хмарність %	5	15	25	35	45	55	65	75	85	95
К-ть днів	70	35	26	22	20	21	19	26	38	65

За даними таблиці студенти будують гістограму і бачать її характерну форму (Рис. 4.14).

3. Нормальний розподіл.

Задача 4.19. Медична експертиза в армії подала дані про зріст 321 солдата:

Зріст (см)	165	170	175	180	185	190	195
К-ть	10	29	72	98	69	31	12

Студенти будують гістограму (Рис.4.15) і бачать її характерну форму, на основі чого вони можуть зробити висновок, що у цьому випадку дані добре групуються біля середнього арифметичного, причому середнє арифметичне (180.1) мало відрізняється від медіани (180).

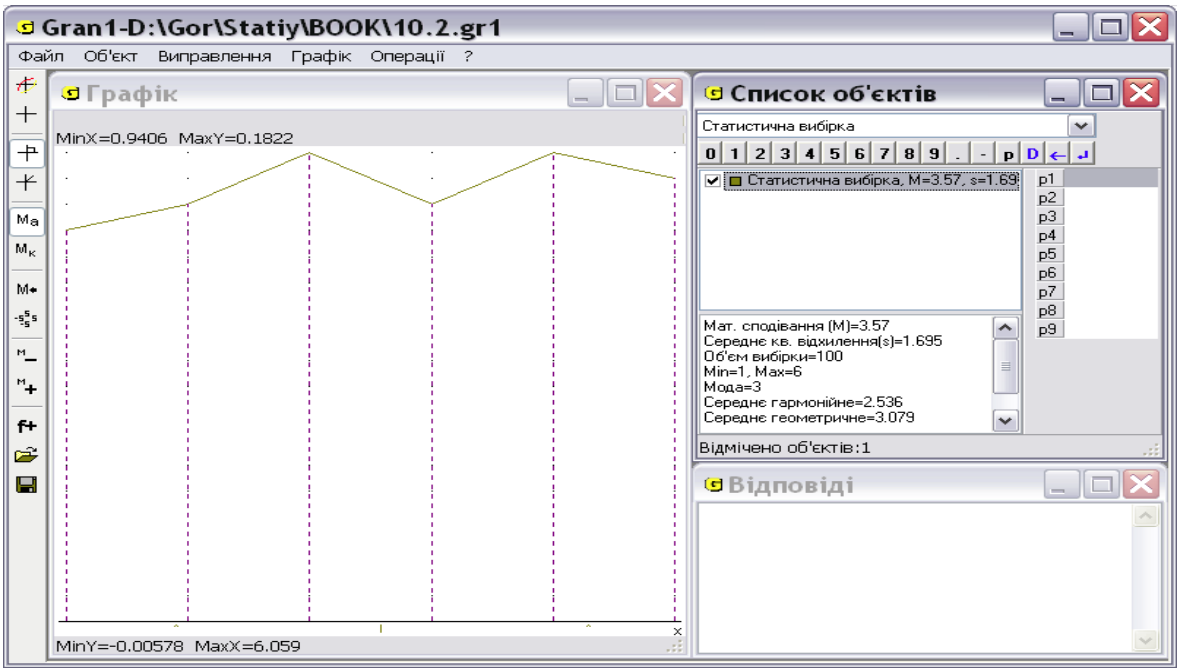

Рис. 4.13. Полігон до Задачі 4.17.

Доцільно наголосити студентам, що із умов задач 4.18 та 4.19 видно, що досліджувані величини неперервні, тому слід розглядати інтервальні розподіли відносних частот, а наведені у верхньому рядку числа є серединами відповідних інтервалів.

Нормальний розподіл відносних частот отримують, коли на значення якоїсь з величин досліджуваного об’єкта діє велика кількість факторів,

кожен з яких вносить досить малу частку в значення цієї величини. Слід відмітити, що нормальний розподіл дуже часто зустрічається в практичних задачах.

Розгляд всіх трьох прикладів, наведених в цьому параграфі, дозволяє студентам також зробити висновок про те, що сама лише величина центральної тенденції не може надійно репрезентувати розподіл частот на множині значень досліджуваної величини. Тому виникає потреба у характеристиці ступеня розсіювання частот (як щільно вони групуються біля середнього).

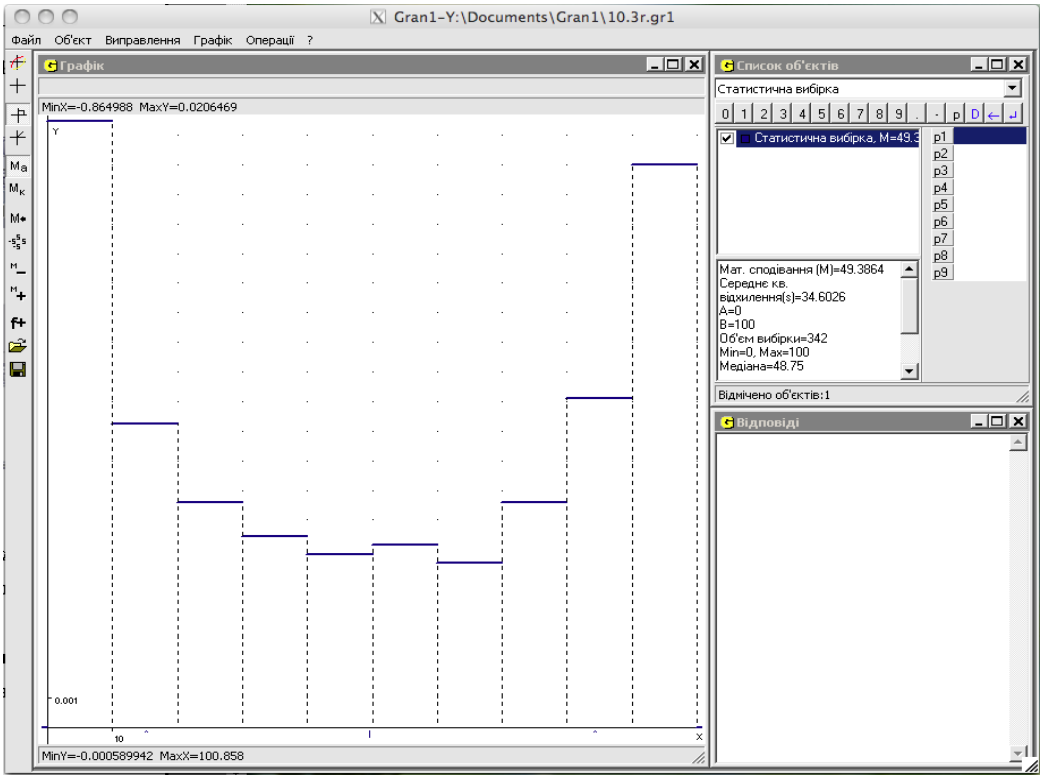

Рис. 4.14. Гістограма до Задачі 4.18.

Найбільш простою характеристикою розсіювання є *розмах вибірки* $R=x_{max}-x_{min}$, де x_{min} – мінімальна варіанта вибірки, x_{max} – максимальна варіанта вибірки.

Але більш суттєвими для визначення величини розсіювання будуть ті значення, які у вибірці зустрічаються найчастіше. Значення ж, частота появи яких мала, практично не впливають на величину розсіювання основної маси варіант, тому розмах вибірки може виявитися завищеною характеристикою розсіювання. Більш істотними характеристиками

розсіювання частот на множині значень досліджуваної величини є дисперсія та середнє квадратичне відхилення.

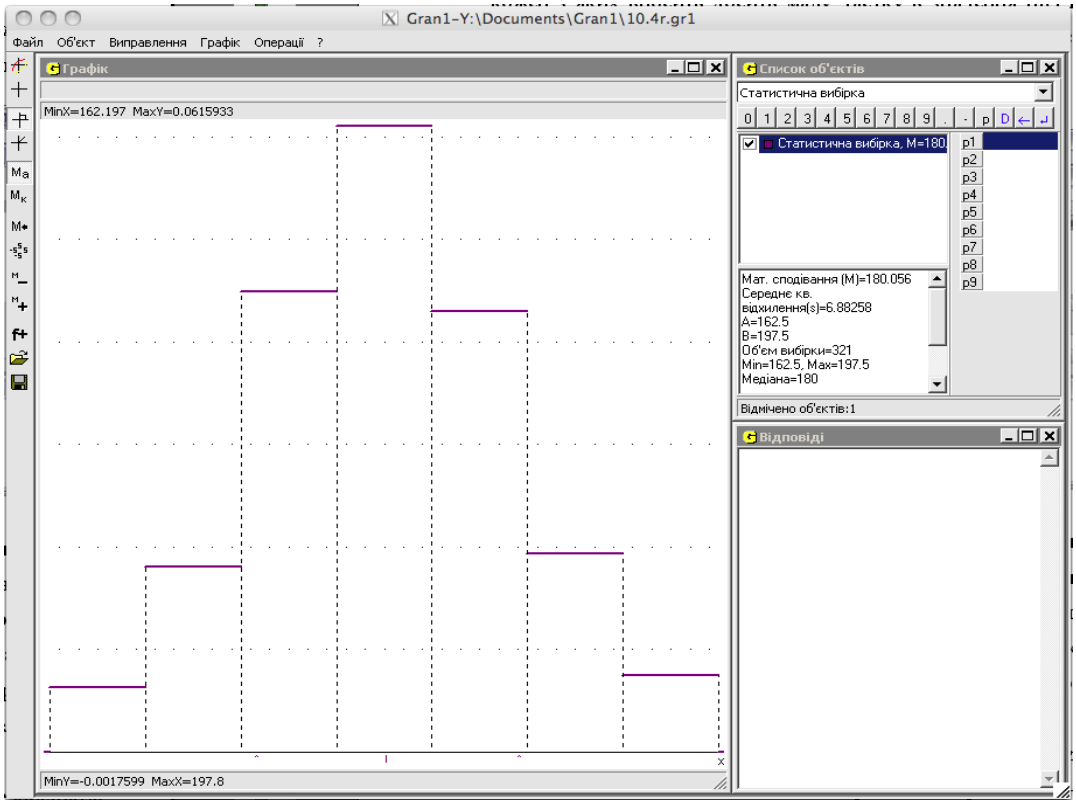

Рис. 4.15. Гістограма до Задачі 4.19.

Дисперсія розподілу частот (D_n^*) – це середнє арифметичне квадратів відхилення спостережених значень $x_{1\ \text{сп}}$, ..., $x_{n\ \text{сп}}$ досліджуваної величини X від середнього значення M_n^*:

$$D_n^* = \frac{\sum_{i=1}^{n}(x_{i\text{сп}} - M_n^*)^2}{n} = \sum_{i=1}^{m}(x_i - M_n^*)^2 \cdot P_n^*(\{x_i\}).$$

Зрозуміло, що чим менша дисперсія, тим щільніше групуються значення вибірки біля середнього арифметичного.

Середнім квадратичним відхиленням (σ_n^*) для вибірки називають квадратний корінь з дисперсії D_n^*:

$$\sigma_n^* = \sqrt{D_n^*}.$$

Про середнє арифметичне можна сказати, що воно є досить точною характеристикою розсіювання статистичних ймовірностей на множині значень досліджуваної величини (при достатньо великій кількості

спостережень), тому вибірковим середнім арифметичним можна користуватись для оцінювання невідомого "істинного" середнього. Якщо ж спробувати оцінювати невідому "істинну" дисперсію за вибірковою дисперсією D_n^*, то така оцінка даватиме занижене значення "істинної" дисперсії. Тому вибіркову дисперсію “виправляють”, для отримання більш точної оцінки "істинної" дисперсії: “виправлена” дисперсія позначається s^2 і обчислюється за формулою

$$s^2 = \frac{(n-1)}{n} D_n^* .$$

Відповідно “виправлене” середнє квадратичне відхилення обчислюються за формулою

$$s = \sqrt{s^2} .$$

У ППЗ Gran1 вибіркові середнє арифметичне та „виправлене” середнє квадратичне відхилення обчислюються автоматично.

Для для підвищення рівня компетентностей студентів щодо побудови інформаційних моделей доцільно запропонувати їм звернутися до літературних джерел типу [104], [110] та розв’язати наступні задачі.

Задача 4.20. Відомо дані щодо зарплат співробітників певної фірми.

Зарплата (грн..)	230	380	560	2000	3000
Кількість співробітників	20	7	4	2	1

Необхідно визначити, яка є “середня” зарплата співробітників.

Задача 4.21. В списку наведено округлений до одного см розмір ноги учнів трьох класів:

см	15	16	17	18	19	20	21	22	23	24	25	26	27	28
n	1	3	5	6	5	4	6	8	7	6	5	3	2	1

Необхідно визначити числові характеристики вибірки – моду, медіану, середнє арифметичне, дисперсію, середнє квадратичне відхилення.

Задача 4.22. За допомогою фотографування з повітря з’ясовано, що в районі на півночі Саскачеваня (Канада) існує 1564 колоній бобрів. За

допомогою біологічних досліджень було підраховано кількість бобрів в кожній з 30 колоній. Отримані такі результати:

5	2	4	7	6	5	3	7	7	8
4	6	8	5	5	8	7	2	6	4
7	7	6	8	7	5	7	4	5	6

Необхідно:

1) побудувати частотну таблицю і знайти середнє арифметичне і моду спостережених кількостей бобрів;
2) підрахувати кількість колоній з 4 і більше бобрами;
3) підрахувати загальну кількість бобрів в районі;
4) з'ясовано, що в середньому 4 бобри в колонії – умова виживання. На скільки повинна зменшитися популяція бобрів, щоб середня кількість стала 4?

Задача 4.23. Чи можливо визначити підручники з гуманітарних та точних дисциплін за довжиною слів, що використовуються автором? Потрібно провести наступний експеримент:

1) Вибрати довільну сторінку підручника з літератури.
2) Підрахувати кількість літер в кожному з 100 перших слів на сторінці.

Заповнити запропонований нижче список:

К-ть літер в слові	1	2	3	4	5	6	7	8	9	≥10
К-ть слів										

Повторити експеримент для підручника з математики (тільки слова, ігноруючи формули, числа та вирази).

Зробити висновок.

4.1.9. Перевірка статистичних гіпотез. Коли експериментатор перевіряє якесь твердження (наприклад твердження про те, що дані антропометричних вимірів мають нормальний розподіл ймовірностей) на основі отриманих статистичних даних, майже завжди з'ясовується, що

існує певна різниця між очікуваними і експериментальними даними. Принциповим є питання, чи ці розходження мають систематичний характер, чи вони мають виключно випадковий характер, викликаний другорядними, нерегульованими в досліді випадковими чинниками. Твердження про те, що спостережені відмінності визначаються випадковими чинниками, називається нульовою гіпотезою. Сформульована гіпотеза потребує перевірки. Твердження про правильність гіпотези може висловлюватися з тією чи іншою мірою впевненості (надійності). Якщо проводити низку дослідів, деякі з них можуть підтверджувати сформульовану гіпотезу, деякі – ні. Частку випадків, які не протирічать вибраній гіпотезі, будемо називати рівнем значущості (зауважимо, що в деяких посібниках рівнем значущості називають частку випадків, що протирічать сформульованій гіпотезі) і позначатимемо символом α. При виборі α слід пам'ятати, що значення α, близькі до нуля, вибирають, щоб не допустити прийняття хибної гіпотези, але в цьому випадку великий ризик відхилити правильну гіпотезу. Значення α, близькі до 1, вибирають, щоб не відхилити правильну гіпотезу, але так збільшується ризик прийняття гіпотези неправильної.

Одним з найбільш часто використовуваних на практиці критеріїв перевірки статистичних гіпотез є критерій Пірсона (χ^2).

Нехай в результаті n експериментів отримали вибірку і побудували інтервальний розподіл статистичних ймовірностей на множині значень випадкової величини X; k – кількість інтервалів. Тоді за цим критерієм з'ясовують, чи не можна щільність розподілу ймовірностей $f^*_{nX}(x)$ з точністю, достатньою для практичних застосувань, замінити деякою іншою функцією $f_X(x)$. Ця функція може бути задана певним математичним виразом або еталонним розподілом. Зрозуміло, що функція $f_X(x)$ повинна бути визначеною на деякому відрізку $[a_0, b_0]$, що включає в себе відрізок задання вибірки $[a, b]$, і інтеграл від $f_X(x)$ на відрізку $[a_0, b_0]$ повинен буди рівним 1.

Величину розходження (чи близькості) функцій $f^*_{nX}(x)$ і $f_X(x)$ можна охарактеризувати числом

$$\chi^2_{експ.} = \frac{1}{n}\sum_{i=1}^{k}\frac{(m_i - np_i)^2}{p_i},$$

де m_i – абсолютна частота попадання спостережених значень в проміжок $[a_{i-1}; a_i)$,

$$p_i = \int_{a_{i-1}}^{a_i} f_X(x)dx,\ i = 1,2,...k.$$

Далі обчислюють значення $\chi^2_{теор}$, яке залежить тільки від рівня значущості α та кількості ступенів вільності. Кількість ступенів вільності для нормального розподілу визначається за формулою $s=k-3$, в інших випадках доцільно скористатися формулою $s=k-1$. Якщо $\chi^2_{експ}<\chi^2_{теор}$, то вважають, що гіпотеза про випадковість розходження між експериментальними і гіпотетичними значеннями не суперечить статистичним даним. В протилежному випадку вважають, що розходження викликане систематичними причинами і гіпотеза не узгоджується із статистичними даними.

Аналогічно перевіряються гіпотези про дискретні розподіли частот.

Доцільно звернути увагу студентів на те, що використання засобів ІКТ (ППЗ Gran1) при вивченні математичної статистики, зокрема і методів статистичного моделювання, імітаційного моделювання і ін., дозволяє перекласти на комп’ютер громіздкі обчислення, зосередившись на сутності методів, що розглядаються, зокрема на сутності методів перевірки статистичних гіпотез.

Для демонстрації правил використання критерію Пірсона доцільно розглянути зі студентами розв'язування наступної задачі:

Задача 4.24. Використовуючи критерій Пірсона, при рівні значущості 0.98, перевірити, чи узгоджується гіпотеза про нормальний розподіл ймовірностей з розподілом статистичних ймовірностей при об’ємі вибірки 200:

x_i	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9	2.1	2.3
Частота	6	9	26	25	30	26	21	24	20	8	5

Щільність нормального розподілу ймовірностей визначається за формулою

$$f(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-a)^2}{2\sigma^2}}$$

Розв'язування задачі за стандартною схемою є дуже громіздким (обчислення теоретичних частот за наведеною формулою, знаходження теоретичного і експериментального значень χ^2), і ненаочним. Тому педагогічно доцільно запропонувати студентам скористатися ППЗ Gran1.

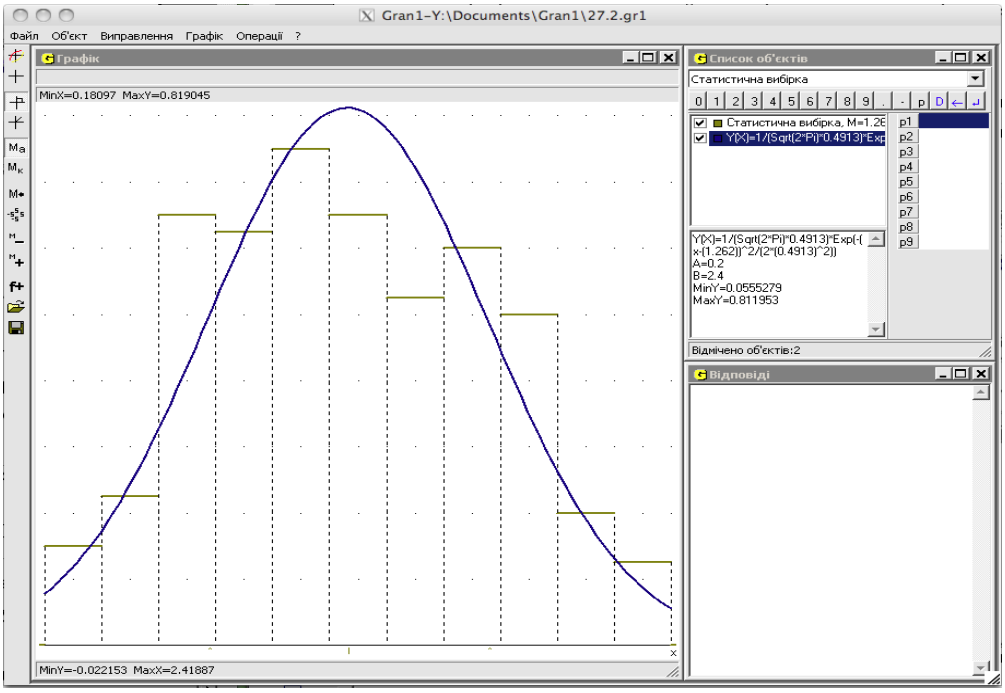

Рис. 4.16. Відповідь до Задачі 4.24.

Подальша послідовність дій студентів може бути така: встановити тип задання залежності між змінними *"Статистична вибірка"* і за допомогою послуги *"Об'єкт/Створити"* ввести заданий інтервальний розподіл статистичних ймовірностей, вказавши, що він є неперервним. За допомогою послуги *„Операції/Статистика/Щільність нормального розподілу за вибіркою"* ввести щільність нормального розподілу, що має такі самі

математичне сподівання та середнє квадратичне відхилення, як і визначені за вибіркою.

Звернемося до послуги *“Операції/Статистика/Критерій Пірсона”*.

Вказавши рівень значущості 0.98, а кількість ступенів вільності 8, отримаємо, що експериментальні дані не суперечать гіпотезі про нормальний розподіл ймовірностей (Рис. 4.16).

Задача 4.25. Перевірити правильність закону дигібридного схрещування, виходячи з наступних даних, отриманих при схрещуванні мух-дрозофіл, що мають нормальні крила і червоно-коричневі очі, з мухами, що мають зачаткові крила і яскраво-червоні очі:

Крилаті, червоно-коричневі	Крилаті, яскраво-червоні	Зачаткові, червоно-коричневі	Зачаткові, яскраво-червоні
50	20	22	4

Оскільки в задачі розглядаються якісні ознаки (колір та розмір крил), перед введенням даних в комп’ютер необхідно ці ознаки пронумерувати. Зручніше нумерувати ознаки послідовними натуральними числами, починаючи з одиниці. Виконавши це, отримаємо наступний набір даних:

1	2	3	4
50	20	22	4

Задача полягає в перевірці гіпотези, яку можна сформулювати таким чином: отримані експериментальні дані підтверджують закон дигібридного схрещування. Для перевірки гіпотези потрібний еталонний набір даних. Оскільки в законі дигібридного схрещування говориться про те, що ознаки розщеплюються в пропорції 9:3:3:1, еталонний набір даних може мати такий вигляд:

1	2	3	4
9	3	3	1

Подальші дії для розв’язування задачі з використанням програми Gran1 можна подати у такій послідовності:

1. Встановити тип залежності *“Статистична вибірка”* і за допомогою послуги *“Об’єкт/Створити”* ввести експериментальні дані, вказавши, що розподіл частот дискретний.
2. За допомогою цієї ж послуги ввести еталонні дані, вказавши, що розподіл частот дискретний.
3. За допомогою послуги *“Графік/Побудувати”* впевнитися, що хоча форма полігонів схожа, між ними є певні відмінності.
4. Вибрати послугу *“Операції/Статистика/Критерій Пірсона”* для перевірки гіпотези про те, що відмінності між експериментальними і еталонними даними викликані випадковими факторами і є несуттєвими. Вказати, що гіпотетичною (теоретичною) є друга вибірка, а рівень значущості встановити 0.98 (Рис.4.17).
5. Отримати відповідь (Значення $\chi^2_{експ}$, $\chi^2_{теор}$ та повідомлення про те, що гіпотеза підтверджується – Рис. 4.18).

Задача 4.26 Для задачі 4.12 (про генератор випадкових чисел) перевірити з рівнем значущості α=0.99 гіпотезу про рівномірний розподіл ймовірностей на відрізку [0, 1], розглядаючи як статистичний матеріал згенерований за допомогою комп’ютера набір чисел.

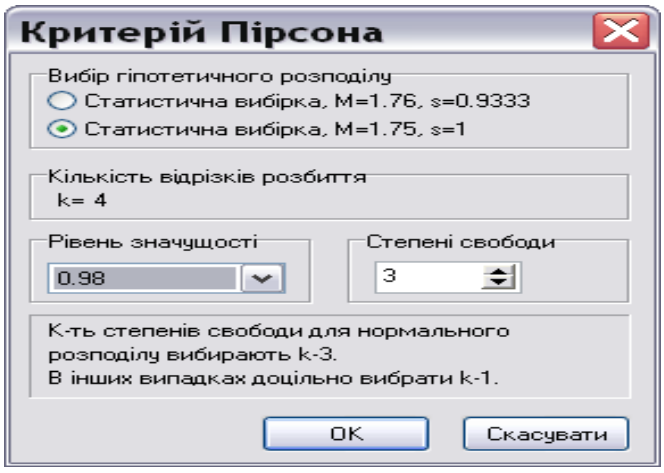

Рис. 4.17. Введення параметрів критерію Пірсона.

Зауваження. *Якщо потрібно дослідити рівномірність неперервного розподілу ймовірностей на множині значень випадкової величини, гістограму потрібно порівнювати з функцією y=1/(b-a), де a, b – кінці відрізка, на якому задано вибірку. В розглядуваному випадку це буде функція y=1.*

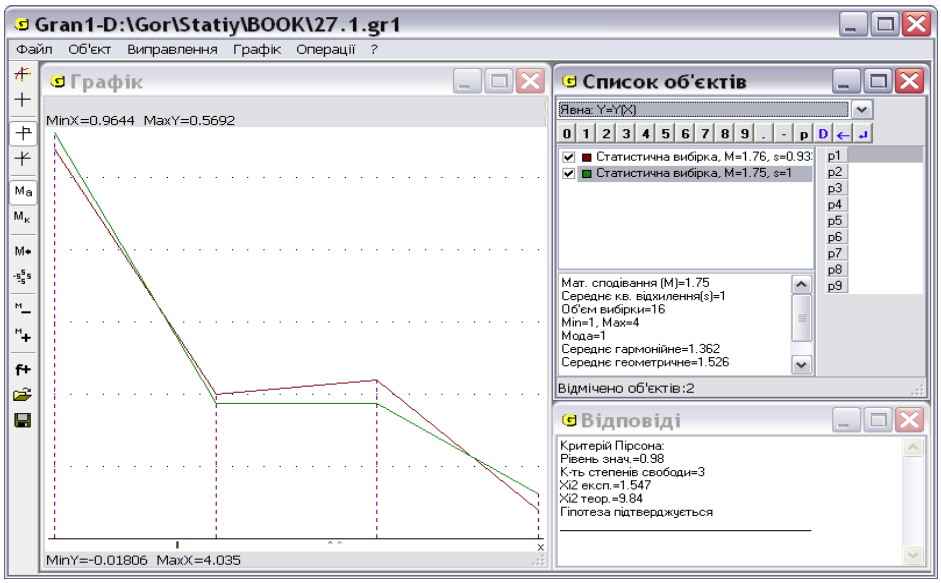

Рис. 4.18. Відповідь до Задачі 4.25.

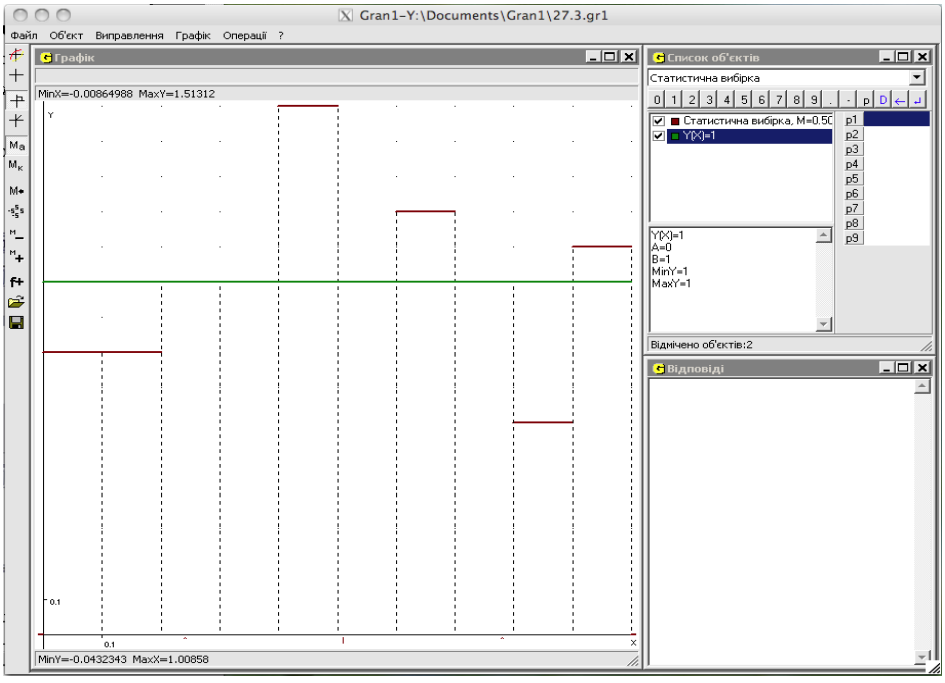

Рис. 4.19. Відповідь до Задачі 4.26.

Для розв’язування задачі потрібно:

- завантажити вибірку, записану раніше на диск;

- задати функцію *y=1*, причому відрізок задання функції повинен співпадати з відрізком задання вибірки, тобто з відрізком *[0; 1]*;
- скористатися послугою *“Операції/Статистика/Критерій Пірсона”*, встановивши рівень значущості $\alpha=0.99$.

Після виконання обчислень виводиться повідомлення про те, що гіпотеза підтверджується (Рис. 4.19).

Задача 4.27. Для перевірки справжності рукописів листів К.К.Снодграсса, які приписують М.Твену, Ч.С.Грінегар підрахував кількість слів довжиною в *k*=1, 2, …, 12 літер в творах М.Твена і в листах К.К.Снодграсса. З’ясувалося, що слова з 2, 3, 4 і більше літерами зустрічаються в творах М.Твена з відносними частотами

2	3	4	Решта
0,177	0,232	0,191	0,400

В 13175 словах рукописів К.К.Снодграсса слова тієї ж довжини зустрічаються з наступними частотами:

2	3	4	Решта
2685	2752	2302	5436

За допомогою критерію Пірсона перевірити гіпотезу про те, що листи К.К.Снодграсса написані М.Твеном.

В даному випадку мова йде про дискретний розподіл статистичних ймовірностей, тип даних для першої вибірки – відносні частоти. Оскільки в задачі не сказано, скільки слів з творів М.Твена було проаналізовано, можна вважати, що їх було приблизно стільки ж, скільки досліджено слів з рукописів К.К.Снодграсса, наприклад 13000. Це число потрібно вписати у полі “Об’єм вибірки”. Тип даних другої вибірки – абсолютні частоти. При введенні вибірок в програму можна вважати всі слова з кількістю літер від 5 до 12 як слова з кількістю літер 5. В задачі не вказано рівень значущості, з яким потрібно провести дослідження, тому його потрібно вибрати

самостійно. Вибравши різні значення α, студенти можуть з'ясувати, чи в даній задачі гіпотеза підтверджується чи ні при заданих значеннях α.

Послідовність дій для розв'язування може бути такою:

- ввести першу вибірку;
- ввести другу вибірку;
- відмітити вибірки і звернутися до команди *"Операції/Статистика/Критерій Пірсона"*, гіпотетичною вибіркою будемо вважати вибірку з даними про твори М.Твена.

Після виконання всіх операцій за програмою буде виведено повідомлення, що гіпотеза не підтверджується, тобто можна вважати, що М.Твен не писав листів К.К.Снодграсса (Рис. 4.20).

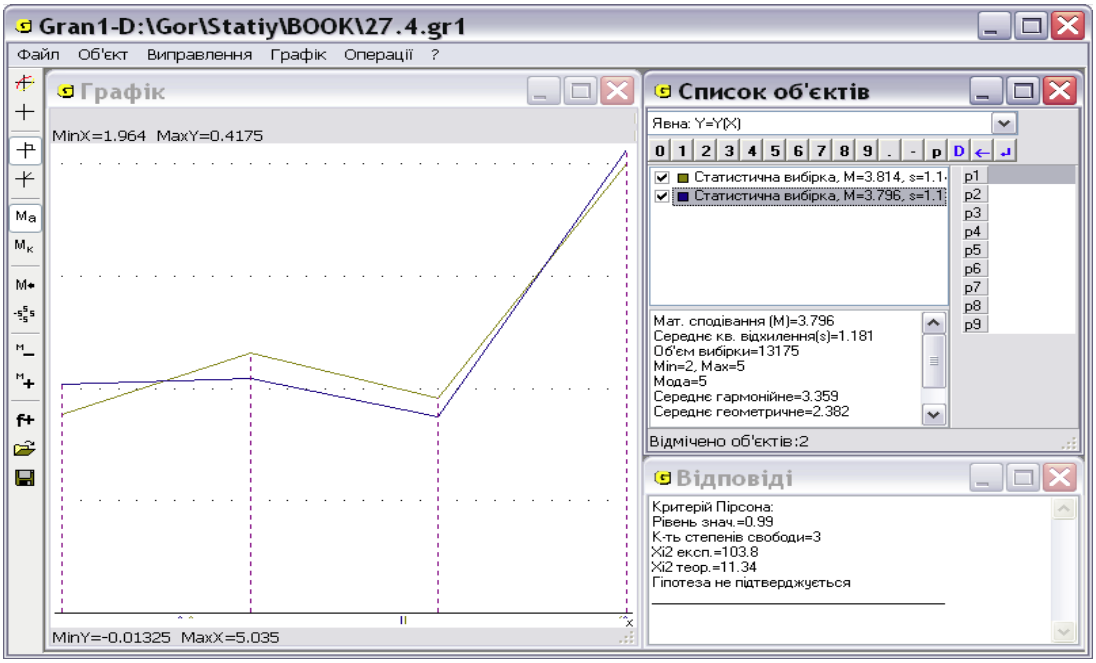

Рис. 4.20. Відповідь до Задачі 4.27.

Задача 4.28. Результати підрахунку частот появи цифр *0, 1, ..., 9* в *10002* знаках десяткового запису числа *(π-3)* наведено в списку:

0	1	2	3	4	5	6	7	8	9
968	1026	1021	974	1014	1046	1021	970	948	1014

За допомогою критерію Пірсона перевірити гіпотезу про рівну можливість появи в даному записі кожної з цифр 0, 1, …, 9.

Зауваження. *В цій задачі потрібно порівнювати дану вибірку з рівномірним розподілом ймовірностей на множині всіх цифр, який і будемо вважати теоретичним:*

0	1	2	3	4	5	6	7	8	9
1000	1000	1000	1000	1000	1000	1000	1000	1000	1000

Розподіл ймовірностей розглядається дискретний, тип даних – частоти.

В результаті розв'язування за допомогою програми Gran1 можна з'ясувати, що при рівнях значущості, близьких до 1, немає причин відхиляти гіпотезу.

Для підвищення рівня компетентностей у студентів щодо теоретичних положень з параграфу 4.1.12 доцільно запропонувати їм розв'язати наступні задачі.

Задача 4.29. Для аналізу впливу вітаміну С на профілактику застудних захворювань 200 чоловік випадковим чином було розподілені на 2 рівних групи. В одній групі дали вітамін С, іншій – схожу пігулку, але без вітаміну, причому всім пацієнтам було сказано, що їм дали вітамін С. Результати обстеження наведено у наступній таблиці:

	Менше застудних захворювань	Більше застудних захворювань	Без змін
Контрольна група	39	21	40
Група, людей, які приймали вітамін	51	20	29

За допомогою критерію Пірсона для рівня значущості $\alpha=0.95$ перевірити гіпотезу про незалежність кількості застудних захворювань від приймання вітаміну С.

Зауваження. *Розподіл ймовірностей – дискретний, тип даних – частоти. Щоб можна було скористатися програмою* ***Gran1****, надамо значенню "Менше застудних захворювань" номер 1, значенню "Більше застудних захворювань" номер 2, значенню "без змін" номер 3. Вибірку для контрольної групи будемо вважати гіпотетичною.*

Задача 4.30. За даними з задачі 6.4 про зріст учнів за допомогою критерію Пірсона для рівня значущості 0.99 перевірити гіпотезу про нормальний розподіл ймовірностей на множині значень зросту учнів.

Зауваження. При розв'язуванні задачі необхідно завантажити файл з вибіркою (звернувшись до послуги програми "Файл/Відкрити"), відмітити вибірку і звернутися до послуги "Операції/Статистика/Функція щільності норм. розподілу за вибіркою". За цією послугою записується до списку об'єктів функція щільності нормального розподілу ймовірностей. Далі необхідно скористатися послугою "Операції/Статистика/Критерій Пірсона".

Задача 4.31. При схрещуванні двох видів кукурудзи у другому поколінні виявлено чотири різних типи рослин. Проста менделевська модель передбачає появу типів з ймовірностями 9/16, 3/16, 3/16, і 1/16. В результаті аналізу 1301 рослин отримані частоти n_1=773, n_2=231, n_3=238, n_4=59. При якому рівні значущості α критерій χ^2 Пірсона підтверджує менделевську модель?

Задача 4.32. Серед 2020 сімей, що мають двох дітей, було зареєстровано 527 сімей, в яких обидві дитини – хлопчики, 476 сімей, в яких обидві дитини – дівчинки, в решті 1017 родинах діти різної статі. Чи можна з рівнем значущості α=0.95 вважати, що на множині значень кількості хлопчиків у родині з двома дітьми розподіл ймовірностей біноміальний?

Задача 4.33. На першому потоці з 300 студентів, що складали екзамен, оцінку „2" отримали 33 студенти, „3" – 43 студенти, „4" – 80 студентів, „5"

– 144 студенти. На другому потоці інші 300 студентів, що складали той самий екзамен у того самого викладача, отримали наступні результати: „2” – 39, „3” – 35, „4” – 72, „5” – 154. Чи можна з рівнем значущості $\alpha=0.95$ вважати обидва потоки статистично однаковими?

4.2. Моделювання і вивчення основ алгоритмізації

Саме інформатика є тією наукою, вивчення якої дозволяє найбільш послідовно і цілеспрямовано формувати знання щодо моделювання взагалі і інформаційного моделювання зокрема. При вивчення курсу “Основи алгоритмізації і алгоритмічні мови” слід приділити увагу формуванню компетентностей стосовно всіх етапів моделювання при розв’язуванні задач за допомогою комп’ютера.

1. Змістова постановка задачі. Спочатку потрібно усвідомити задачу, чітко сформулювати її. При цьому визначаються об'єкти, які розглядаються в задачі, їх взаємозв’язки.
2. Системний аналіз. Щоб задачу можна було описати кількісно і використати при її розв’язуванні обчислювальну техніку, потрібно зробити якісний і кількісний аналіз об'єктів і ситуацій, що розглядаються в задачі. При цьому складні об'єкти поділяють на елементи, визначають зв'язки цих елементів, їх властивості, кількісні і якісні значення властивостей, кількісні й логічні співвідношення між ними, що виражаються у вигляді рівнянь, нерівностей і т.п.
3. Наступним етапом є математична постановка задачі, у процесі якої здійснюється побудова математичної моделі об'єкта й визначення методів (алгоритмів) одержання розв’язку задачі. Слід зауважити, що на цьому етапі може виявитися, що раніше проведений системний аналіз привів до такого набору елементів, властивостей і співвідношень, для якого немає прийнятного методу розв’язування задачі. В результаті доводиться повертатися до етапу системного аналізу. Як правило, розглядувані в курсі основ програмування і

алгоритмізації задачі стандартизовані, системний аналіз розробляється з огляду на відому математичну модель і алгоритм її аналізу, проблема полягає лише у виборі відповідного методу. Тому крім таких задач треба розглядати і більш складні задачі, які мають нестандартне формулювання, фабулу, неочевидні методи розв'язування, так звані олімпіадні задачі. Це дозволить підвищити у студентів навички інформаційного моделювання, зацікавити їх даною дисципліною, підготувати до творчої педагогічної діяльності, сформувати відповідні компетентності щодо навчання основ програмування, проведення шкільних олімпіад.

4. Подальшим етапом є розробка програми розв'язування задачі за допомогою комп'ютера. У курсі основ програмування і алгоритмізації цей етап полягає у написанні програми тією мовою програмування, що вивчається, в певних інших випадках важливим є і вибір самої мови програмування, оскільки мов програмування існує багато, як універсальних, так і розрахованих на досить вузьке використання (офісні пакети, бази даних, веб-програмування, системи комп'ютерної математики - СКМ та ін.). Слід також зауважити, що розв'язування реальної задачі з певної предметної галузі за допомогою комп'ютера у більшості випадків, особливо на даному етапі розвитку інформаційних технологій, не зводиться до написання програми тією чи іншою мовою програмування, а може бути здійснене з використанням тієї чи іншої прикладної програми, хоча не виключається і використання вбудованої мови програмування, якщо така передбачена в прикладній програмі. Для складних об'єктів, що складаються з великої кількості елементів, з великим числом властивостей, може знадобитися складання бази даних і використання засобів роботи з нею, методів знаходження даних, потрібних для розрахунків. Для багатьох економічних задач або задач на складні розрахунки можна скористатися табличним процесором. Не слід

забувати і про спеціалізовані математичні пакети, за допомогою яких можна розв’язувати досить складні математичні задачі, проводити обчислення з надвисокою точністю, якщо це потрібно, візуалізувати складні математичні об’єкти, важливо лише правильно дібрати потрібний пакет програм. Щоб зробити такий вибір, студент повинен глибоко знати характеристики і можливості використання таких пакетів, чітко бачити сфери їх використання.

5. Тестування отриманої програми. Це досить важливий етап, призначений для виявлення помилок, що могли виникнути при розробці моделі, алгоритму та програми. На жаль, більшість студентів нехтує цим етапом, вважаючи, що, якщо програма “запрацювала”, то вона вже правильна. У найкращому випадку перевіряється правильність програми на одному з простих тестів. Тому дуже важливо навчити студентів правильно тестувати програми, добирати систему тестів таким чином, щоб вона охоплювала всі можливі нюанси щодо вхідних даних, наприклад граничні значення. Відомо, що всі типи величин у багатьох мовах програмування мають чітко визначений діапазон значень, тому ще на етапі розробки програми потрібно правильно вибрати типи величин, щоб вхідні, вихідні та проміжні дані не виходили за межі відповідних діапазонів, а на етапі тестування перевірити правильність вибору типів. До речі, в деяких із задач жоден із стандартних типів даних вибраної мови програмування може не задовольнити умови щодо діапазону. В цьому випадку доведеться конструювати власний тип даних, наприклад так звану “довгу арифметику”. Важливо постійно звертати увагу студентів на такі особливості, тому що більшість з них це випускає з виду, наприклад багато студентів не звертають увагу на те, що цілочисельний тип даних не придатний для обчислення факторіалів навіть не дуже великих чисел. Також можуть виникнути проблеми

переповнення стеку у рекурсивних алгоритмах, і такі проблеми теж повинні бути визначені за допомогою тестування.

6. Останнім етапом є експлуатація моделі, отримання результатів на реальних даних та трактування отриманих результатів у термінах предметної галузі задачі.

Важливо також звертати увагу студентів при розробці алгоритмів на швидкодію цих алгоритмів. Досить часто виявляється, що очевидний алгоритм розв'язування задачі виконується за неприйнятно довгий проміжок часу. Потрібно навчити студенів визначати швидкодію алгоритму на основі аналізу його циклів, знайомити студентів із швидкими алгоритмами для певних класів задач (швидкими алгоритмами сортування, пошуку), а також із шляхами оптимізації алгоритмів щодо їх швидкодії.

Як приклад розв'язування олімпіадної задачі, наведемо задачу „Терези", запропоновану на одній з обласних олімпіад з інформатики на Чернігівщині [63]. В математичній літературі ця задача має назву задачі Баше-Мендєлєєва [80].

Задача 4.34. Леонардо Пізанський, Лука Пачіолі та ін. розглядали задачу про найкращу систему важків. Нею цікавився і Д.І.Мендєлєєв, коли був директором Головної палати мір та вагів. Французький математик Клод Баше опублікував цю задачу у „Збірнику цікавих задач", виданому у 1612 р. Ця задача існує у двох варіантах. Спочатку розглянемо більш простий, а саме, при якій системі важків, маючи їх по одному, можна зважити всі можливі вантажі до деякого найбільшого, якщо класти важки тільки на одну шальку терезів? Маються на увазі вантажі з цілочисельною вагою.

Зрозуміло, що максимальний вантаж, який може бути зважений даною системою важків, дорівнює їх сумі. Таким чином треба дібрати систему важків такою, щоб їх кількість була мінімальною і за допомогою неї можна було б зважити всі цілі вантажі від 1 до максимального. Дібрати таку систему важків можна, наприклад, наступними міркуваннями. Нехай за

допомогою системи важків можна зважити будь-який вантаж від 1 до M. Тоді наступний важок у системі повинен бути M+1. Нехай M=1, тоді наступний важок буде масою 1+1=2. Важками з масами 1 та 2 можна зважити вантажі з масою від 1 до 3, тоді наступний важок буде 3+1=4. Таким чином ми отримаємо систему важків 1, 2, 4, 8, ..., тобто степенів двійки. Зважування вантажу за допомогою такої системи важків зводиться до запису маси цього вантажу у двійковій системі числення. Оскільки будь яке десяткове число можна подати у вигляді двійкового числа, то доведено, що така система важків дозволить зважити будь-яке число від 1 до суми мас всіх важків. Оптимальність такої системи випливає з процесу її побудови. Для добору системи важків для зважування конкретного вантажу досить записати його масу у двійковому поданні, що є досить легкою задачею.

Більш складний варіант задачі можна сформулювати наступним чином: при якій системі важків, маючи їх по одному, можна зважити всі можливі вантажі до деякого найбільшого, якщо можна класти важки на обидві шальки терезів?

Покласти важок на ту саму шальку, де знаходиться вантаж, означає відняти його масу від маси важків, що знаходяться на іншій шальці.

Для вибору системи важків можна скористатися наступними міркуваннями. Нехай за допомогою системи важків можна зважити будь-який вантаж від 1 до M. Оскільки важки можна класти на ту саму шальку, де лежить сам вантаж, масу наступного важка у системі можна взяти на одиницю більшу, ніж M+M (маса максимального вантажу для попередньої системи важків плюс маса всіх попередніх важків). Нехай M=1. Тоді наступний важок буде M+M+1=3. Отримаємо систему важків 1, 3, 9, 27, 81, ..., тобто степені трійки. Таким чином процес зважування аналогічний до запису числа у трійковій системі числення, але не звичайній (з цифрами 0, 1, 2), а в системі з цифрами -1, 0, 1 (-1 – важок лежить на тій шальці, де і вантаж, 0 – важок не використовується у зважуванні, 1 – важок на шальці,

де нема вантажу). Виникає питання, чи можна подати будь-яке натуральне число у такій специфічній трійковій системі числення. Відповідь позитивна, можна запропонувати наступний спосіб переходу від звичайної трійкової системи числення до специфічної для цієї задачі: якщо на будь-якому місці запису числа у звичайній трійковій системі числення є цифра 2, можна додати і відняти 1 і отримати одну одиницю вищого розряду і від'ємну одиницю попереднього розряду. Так, наприклад,

$12021_3=121(-1)1_3=2(-1)1(-1)1_3=1(-1)(-1)1(-1)1_3$.

На олімпіаді був запропонований більш складний варіант задачі у наступному вигляді.

Завдання:

Визначити мінімальний набір важків для зважування за допомогою терезів предметів масою в межах від 1 до 1000000000 г (маса – натуральне число). Дані про важки розмістити файлі „giriset.sol", що складається з n+1 рядка: перший рядок – кількість важків n, другий рядок – маса 1-го важка, 3-й рядок – маса 2-го важка, ... n+1 рядок – маса n-го важка.

Розробити програму "giri", за якою розподіляються знайдені важки між шальками терезів для зважування предмета вказаної маси. Вхідні дані – файл „giri.dat", що містить єдине ціле число – масу предмета. Вихідний файл „giri.sol" містить n рядків, в кожному з яких записано єдине число: -1, 0, 1 за правилом: якщо в і-му рядку знаходиться -1, це означає, що і-й важок знаходиться на лівій шальці, (там де і сам предмет, що зважують), якщо 1 – і-й важок знаходиться на правій шальці, 0 – важок не використовується у зважуванні.

Для написання програми "giri" можна скористатися алгоритмом, запропонованим в аналізі задачі, а можна запропонувати і інший алгоритм, що базується на аналізі наступного набору даних, в якому записано зважування перших шістнадцяти вантажів:

	27	**9**	**3**	**1**
1	0	0	0	1
2	0	0	1	-1
3	0	0	1	0
4	0	0	1	1
5	0	1	-1	-1
6	0	1	-1	0
7	0	1	-1	1
8	0	1	0	-1
9	0	1	0	0
10	0	1	0	1
11	0	1	1	-1
12	0	1	1	0
13	0	1	1	1
14	1	-1	-1	-1
15	1	-1	-1	0
16	1	-1	-1	1
...	...	...	...	...

З таблиці видно певну циклічність у використанні важків для зважування, а саме у стовпчику для і-го важка спочатку йде кількість нулів, що дорівнює сумі попередніх важків, а потім чергуються групи, що складаються з 1, -1 та 0, причому кількість цифр у групі дорівнює значенню цього важка. Наведемо опис цього алгоритму мовою Pascal:

```
var gr,res: array[1..25] of longint;
    g, Sum, V, V1, Vtmp, Block, Nom_in_Block: longint;
    n,i: integer;
    f0,f:text;
begin
  V:=1000000000;
  assign(f0,'giri.dat');
  reset(f0);
```

```
      readln(f0,v1);
      close(f0);
    {Формуємо систему важків у масиві gr, n – кількість важків у
системі}
      g:=1;
      n:=0;
      Sum:=0;
      while Sum<V do
      begin
        inc(n);
        Sum:=Sum+g;
        gr[n]:=g;
        g:=g*3;
      end;
    {проводимо зважування}
      Sum:=0;
      Vtmp:=V1;
      for i:=1 to n do
      begin
        Vtmp:=V1+Sum;
        Block:=Vtmp div gr[i];
        Nom_in_Block:=Block mod 3;
        if Nom_in_Block=2 then res[i]:=-1 else
          res[i]:=Nom_in_Block;
        Sum:=Sum+gr[i];
      end;
      assign(f,'giri.sol');
      rewrite(f);
      for i:=1 to n do
        writeln(f,res[i]);
      close(f);
    end.
```

Як бачимо, сам алгоритм досить простий, а основна проблема у розв’язуванні цієї задачі полягає якраз у побудові інформаційної моделі, студент повинен “побачити” зв’язок з трійковою системою числення.

Відповідно розв’язування таких задач дозволяє підвищувати рівень компетентностей не тільки стосовно основ алгоритмізації і програмування, а і стосовно систем числення.

Наведемо ще кілька олімпіадних задач з розв’язаннями, запропонованих на чернігівських обласних олімпіадах з інформатики для школярів.

Задача 4.35. "Дужки"

Математичний вираз з дужками зашифрували, замінивши в ньому всі підвирази між дужками літерою v. Пильний учень, спостерігаючи такі зашифровані вирази, помітив, що деякі пари дужок в них зайві, оскільки вони дублюють інші пари дужок. Так у виразі v((v))v одна пара дужок є зайвою, оскільки між відкриваючими дужками нема підвиразу і в той же час нема підвиразу і між відповідними їм закриваючими дужками. Потрібно допомогти учневі вилучити із зашифрованих виразів зайві дужки.

Вхідні дані. Дано вхідний файл QUOTES.DAT, в якому єдиний рядок довжиною від 1 до 255 символів – зашифрований вираз, в якому можуть бути зайві дужки.

Вихідні дані. Потрібно сформувати вихідний файл QUOTES.SOL, в якому єдиний рядок – зашифрований вираз без зайвих дужок. Наприклад:

QUOTES.DAT	**QUOTES.SOL**
v((v(((v)))v(v)v))	v(v(v)v(v)v)

Дана задача відноситься до задач на опрацювання рядків, тому може бути запропонована до розгляду при вивченні відповідної теми, до того ж вона є досить практикоорієнтованою, оскільки така проблема може виникнути при написанні, наприклад, програми-аналізатора математичних виразів. Ідея розв’язування може бути такою: знаходимо найлівішу праву дужку, а потім відповідну їй ліву. Всі пари дужок, що стоять безпосередньо за знайденою парою, є зайвими, і їх потрібно вилучити. Знайдену пару дужок замінюємо на квадратні дужки, щоб вони не

заважали у наступному аналізі виразу. Даний процес потрібно повторювати, поки можна знайти закриваючу круглу дужку. На основі сформульованої вище вербальної інформаційної моделі легко написати програмний код. Нижче наводиться цей код, описаний мовою Object Pascal.

```
program quotes;

{$APPTYPE CONSOLE}

uses
  SysUtils;
var fin,fout:textfile;
    s:string;
    l,r,i:integer;

begin
  assignfile(fin,'quotes.dat');
  assignfile(fout,'quotes.sol');
  reset(fin);
  rewrite(fout);
  readln(fin,s);
  repeat
    r:=pos(')',s);
    if r=0 then break;
    l:=r-1;
    while s[l]<>'(' do dec(l);
    while (r<length(s)) and (s[r+1]=')') and (s[l-1]='(') do
    begin
      delete(s,r+1,1);
      delete(s,l-1,1);
      l:=l-1;
      r:=r-1;
    end;
    s[l]:='[';
    s[r]:=']';
```

```
  until false;
  for i:=1 to length(s) do
  begin
    if s[i]='[' then s[i]:='(';
    if s[i]=']' then s[i]:=')';
  end;
  writeln(fout,s);
  closefile(fin);
  closefile(fout);
end.
```

Задача 4.36. “Гра”

Жителі країни Олімпія у вільний час люблять грати в наступну карткову гру: гравцеві роздають 6 карток із 36 карток (серед яких по 9 карток чотирьох кольорів, пронумерованих від 11 до 19), і об’являють, який колір є головним. Потім йому пропонують 4 картки. Гравець повинен повідомити, якою карткою із розданих йому шести карток він перекриває кожну запропоновану йому картку (картка з більшим номером перекриває картку з меншим номером, якщо вони одного кольору; картка головного кольору може перекривати будь-яку картку іншого кольору). Якщо хоча б одну картку перекрити не вдається, гравець повідомляє число 0. Кольори карток позначено числами від 1 до 4. Якщо є кілька варіантів перекривання запропонованих карток, потрібно повідомити будь-який один.

Вхідні дані. Перші шість рядків вхідного текстового файлу GRA.DAT містять дані про роздані гравцеві картки, тобто по 2 числа – величина номера картки і через пропуск її колір. Далі йде 1 рядок що містить одне число – головний колір. Потім йдуть 4 рядки, що містять дані про запропоновані гравцеві картки, тобто по 2 числа – номер картки і через пропуск її колір.

Вихідні дані. Якщо вдалося перекрити всі 4 картки, цей файл повинен містити 4 рядки, в кожному з яких 4 числа, розділених пропусками: номер

і колір запропонованої картки, номер і колір картки, якою перекрито запропоновану:

GRA.DAT	GRA.SOL
15 1	11 2 13 2
12 4	16 1 12 4
13 2	15 3 17 3
13 3	11 3 13 3
11 1	
17 3	
4	
11 2	
16 1	
15 3	
11 3	

Якщо хоча б одну з запропонованих карток перекрити не вдається, вихідний текстовий файл GRA.SOL повинен містити один рядок з числом 0:

GRA.DAT	GRA.SOL
15 1	0
15 2	
13 1	
13 2	
11 3	
12 3	
4	
14 1	
14 2	
14 3	
14 4	

Алгоритм розв’язування цієї задачі відноситься до так званих жадібних алгоритмів.

Жадібний алгоритм — простий і прямолінійний евристичний алгоритм, за яким приймається найкраще рішення, виходячи із наявних на поточному етапі даних, не турбуючись про можливі наслідки, сподіваючись нарешті отримати оптимальне рішення. Легкий в реалізації і, найчастіше, дуже ефективний за часом виконання [90].

В даній задачі “жадібність” полягає в тому, що кожну наступну картку потрібно перекривати карткою з найменшим номером серед тих, що залишилися у гравця і якою можна перекрити дану. Це дозволяє економити картки з більшими номерами для наступних перекривань. Для зручності пошуку найменшого номера картки, якою можна перекрити поточну, картки, роздані гравцеві, сортують за кольорами, а серед карток одного кольору за номерами карток.

Дану задачу доцільно запропонувати в курсі основ алгоритмізації при вивченні такої структури даних, як записи, оскільки доцільно змоделювати картку, як запис, що містить такі поля, як колір та номер на картці. Також у цей запис доцільно додати цілочисельне поле – індекс, яке знадобиться при сортуванні карток. Набір карток можна подати у вигляді масиву записів. Поле індекс заповнюємо за формулою *[колір]*20+[номер на картці]*. Відсортувавши масив карток за цим полем, отримаємо набір карток, відсортований за кольорами, а серед карток одного кольору за величинами номерів на картках. В запропонованій нижче програмі застосовано примітивний метод сортування за методом бульбашки, оскільки кількість карток досить мала. Задачу можна ускладнити, запропонувавши велику кількість кольорів і велику кількість карток кожного кольору. Тоді потрібно буде використати один із швидких методів сортування для отримання достатньої швидкодії програми.

Нижче наводиться код, для реалізації наведеної вище інформаційної моделі, описаний мовою Object Pascal.

```
program gra;

{$APPTYPE CONSOLE}
uses
  SysUtils;

type kartka=record
       nomer: Integer;
       colir: Integer;
       index: Integer;
     end;

var rozdano: array[1..6] of karta;
    hid,hidcopy,perekr: array[1..4] of kartka;
    fin,fout: text;
    i,j,k,n,gcolir,m: integer;
    tmp: kartka;
begin
  assign(fin,'gra.dat');
  reset(fin);
  assign(fout,'gra.sol');
  rewrite(fout);
  {зчитування карт, що роздано гравцеві}
  for i:=1 to 6 do
  begin
    readln(fin,k,m);
    rozdano[i].nomer:=k;
    rozdano[i].colir:=m;
    rozdano[i].index:=m*20+k;
  end;
{зчитування головного кольору}

  readln(fin,gcolir);
  {зчитування карток, розданих гравцеві}
  for i:=1 to 4 do
```

```
      begin
        readln(fin,k,m);
        hid[i].nomer:=k;
        hid[i].colir:=m;
      end;
      {створюємо копію карток, запропонованих гравцеві}
      hidcopy:=hid;
      {сортування}
      for i:=1 to 6 do
      for j:=1 to 5 do
        if rozdano[j].index> rozdano[j+1].index then
        begin
          tmp:= rozdano[j];
          rozdano[j]:= rozdano[j+1];
          rozdano[j+1]:=tmp;
        end;
      {перекриття карток}
      n:=0;
      for i:=1 to 4 do
      begin
      { шукаємо відповідний колір, якщо перекривання вдалось,
колір картки, що використана у перекриванні замінюємо на 0, щоб
виключити її із подальшого розгляду}
      for j:=1 to 6 do
      if (rozdano[j].colir=hid[i].colir) and (rozdano
[j].nomer>hid[i].nomer) then
        begin
          n:=n+1;
          perekr[n]:= rozdano[j];
          rozdano[j].colir:=0;
          hid[i].colir:=0;
          break;
        end;
      { якщо картка головного кольору не перекрита, то кінець}
      if hid[i].colir=gcolir then break;
```

```
      {якщо картка не головного кольору, але перекрити її карткою такого ж кольору не вдалось, шукаємо картку головного кольору з найменшим номером}
      if hid[i].colir<>0 then
        for j:=1 to 6 do
          if rozdano[j].colir=gcolir then
          begin
            n:=n+1;
            perekr[n]:= rozdano[j];
            rozdano[j].colir:=0;
            hid[i].colir:=0;
            break;
          end;
      end;
      {якщо перекрито менше 4-х карток то виводимо 0, інакше виводимо порядок перекривання карток}
      if n<4 then
        writeln(fout,0)
      else
        for i:=1 to 4 do
          writeln(fout,hidcopy[i].nomer,' ',hidcopy[i].colir,' ',perekr[i].nomer,' ',perekr[i].colir);
      close(fin);
      close(fout);
    end.
```

Оскільки студенти можуть запропонувати різні алгоритми для розв’язування цієї задачі, в тому числі і неправильні, але такі, за допомогою яких розв’язуються якісь частинні випадки цієї задачі, для автоматизації перевірки можна запропонувати наступну програму, за якою перевіряється правильність відповіді студента для конкретного тесту. Програма перевірки описана мовою Object Pascal.

```
    program testgra;

    {$APPTYPE CONSOLE}
```

```
uses
  SysUtils;

type kartka=record
       nomer: Integer;
       colir: Integer;
       index: Integer;
     end;

var rozdano: array[1..6] of kartka;
    hid,hidcopy: array[1..4] of kartka;
    f1,f2,f3: text;
    i,j,k,n,gcolir,m,k3,m3: integer;
    tmp,k1,k2: kartka;
    goodk1,goodk2: boolean;

begin
  assign(f1,'gra.dat');
  reset(f1);
  assign(f2,'gra.sol');
  reset(f2);
  assign(f3,'gra.txt');
  rewrite(f3);

  for i:=1 to 6 do
  begin
    readln(f1,k,m);
    rozdano[i].nomer:=k;
    rozdano[i].colir:=m;
    rozdano[i].index:=m*20+k;
  end;
  readln(f1,gcolir);
  for i:=1 to 4 do
  begin
    readln(f1,k,m);
```

```
  hid[i].nomer:=k;
  hid[i].colir:=m;
end;

hidcopy:=hid;
for i:=1 to 6 do
for j:=1 to 5 do
  if rozdano[j].index> rozdano[j+1].index then
  begin
    tmp:= rozdano[j];
    rozdano[j]:= rozdano[j+1];
    rozdano[j+1]:=tmp;
  end;

n:=0;
for i:=1 to 4 do
begin
readln(f2,k,m,k3,m3);
k1.nomer:=k;k1.colir:=m;k2.nomer:=k3;k2.colir:=m3;

goodk2:=false;
for j:=1 to 6 do
if (rozdano[j].colir=k2.colir) and (rozdano
    [j].nomer=k2.nomer) then
  begin
    goodk2:=true;
    ruka[j].colir:=0;
    break;
  end;

goodk1:=false;
for j:=1 to 4 do
if (hid[j].colir=k1.colir) and (hodj].nomer=k1.nomer) then
  begin
    goodk1:=true;
    hod[j].colir:=0;
```

```
          break;
        end;

      if goodk1 and goodk2 and (
((k1.colir=k2.colir) and (k1.nomer<k2.nomer)) or
                                  ((k1.colir<>gcolir) and
(k2.colir=gcolir))
                                )
                                   then

            n:=n+1;
      end;
      close(f1);
      close(f2);

      if n<4 then
        writeln(f3,'-----------')
      else
        writeln(f3,'++++++++++');
      close(f3);
    end.
```

При вивченні основ алгоритмізації у студентів потрібно сформувати компетентності роботи стосовно використання динамічних структур даних, таких як стек, черга, дерево. При вивченні стеку можна розглянути стекову модель обчислень, яка використовується, наприклад, для обчислення виразів, записаних зворотною польською нотацією.

Крім звичного подання математичних виразів, коли знаки операцій записують між операндами, а для зміни порядку виконання операцій використовують дужки, існує також бездужковий метод запису виразів, що називається зворотною польською нотацією. За цією нотацією спочатку записують операнди, а потім знаки операцій. Зворотна польська нотація була розроблена австралійським філософом і фахівцем в галузі теорії обчислювальних машин Чарльзом Хемблiном в середині 1950-х років на

основі польської нотації, яка була запропонована в 1920 році польським математиком Яном Лукасевичем.

Істотною перевагою зворотної польської нотації є простота та швидкість алгоритму обчислення виразу, записаного в такій нотації. Розглянемо цей алгоритм для випадку, коли у виразі зустрічаються тільки бінарні операції (такі, як додавання, віднімання, множення, ділення):

1. вираз аналізується справа наліво;
2. якщо у виразі зустрічається операнд, він заштовхується у стек;
3. якщо у виразі зустрічається знак операції, то ця операція виконується над операндами, що знаходяться у вершині стеку та вузлі стеку, наступному за вершиною, результат розміщується у вузлі, наступному за вершиною, а вершина стеку виштовхується.

Наведемо задачу, для розв'язування якої потрібно застосувати стекову модель обчислень.

Задача 4.37. "Вираз"

Для запису числових виразів жителі країни Олімпія використовують наступний формат:

Вираз складається з операндів та знаків операцій, розділених пропусками.

Операнди: додатні цілі числа.

Операції: додавання (позначається знаком +) та множення (позначається знаком *).

Вираз розглядається зліва направо. Якщо у виразі зустрічається знак операції, виконується відповідна операція над двома останніми операндами, результат операції замінює у виразі набір її операндів і знак операції, після чого обчислення продовжується за цим же правилом.

Результатом обчислення значення виразу є результат останньої операції.

Обчислимо, наприклад, вираз *3 4 2* * +

Перша операція у виразі – множення (*), вона буде виконана над останніми перед нею числами (*4* та *2*), отримаємо число *8*, що замінить групу *4 2 **, таким чином отримаємо вираз *3 8 +*.

У отриманому виразі перша операція, що зустрілась – додавання (+), вона буде виконана над останніми перед нею числами (*3* та *8*), отримаємо число *11*, що замінить групу *3 8 +*.

Оскільки операцій у виразі більше не залишилось, обчислення значення виразу завершилось. Результатом обчислення значення виразу є число *11*.

Обчислити значення виразу, якщо відомо, таке значення знаходиться у межах від 0 до 2000000000.

Вхідні дані. Вхідний текстовий файл EXP.DAT містить вираз, що складається з додатних цілих чисел та знаків операцій, розділених пропусками. Максимальна довжина виразу – 10000 символів.

Вихідні дані. Вихідний текстовий файл EXP.SOL містить єдине число – результат обчислення значення виразу, наприклад:

EXP.DAT	EXP.SOL
3 4 2 * +	11

EXP.DAT	EXP.SOL
58 2 + 10 + 2 *	140

Нижче наводиться код реалізації описаної вище інформаційної моделі, описаний мовою Object Pascal.

```
program exp;

{$APPTYPE CONSOLE}

uses
  SysUtils;
  type pNode=^Node;
       Node=record
```

```
        Data: Integer;
        Next: pNode;
      end;
  procedure Push(var Top: pNode; Adata: Integer);
  var t: pNode;
  begin
    New(t);
    t^.Data:=AData;
    t^.Next:=Top;
    Top:=t;
  end;
  procedure Pop(var Top: pNode);
  var t: pNode;
  begin
    if Top<>Nil then
    begin
      t:=Top^.Next;
      Dispose(Top);
      Top:=t;
    end;
  end;
  var fin,fout: TextFile;
      ch: Char;
      s: String;
      Top: pNode;
begin
  AssignFile(fin, 'exp.dat');
  Reset(fin);
  AssignFile(fout,'exp.sol');
  Rewrite(fout);
  Top:=Nil;
  while not Eof(fin) do
  begin
    Read(fin,Ch);
    if Ch in ['0'..'9','+','*'] then
    case Ch of
```

```
        '0'..'9':begin
                   s:='';
                   while Ch<>' ' do
                   begin
                     s:=s+Ch;
                     Read(fin,Ch);
                   end;
                   Push(Top,StrToInt(s));
                 end;
        '+': begin
               Top^.Next^.Data:=Top^.Next^.Data+Top^.Data;
               Pop(Top);
             end;
        '*': begin
               Top^.Next^.Data:=Top^.Next^.Data*Top^.Data;
               Pop(Top);
             end;
      end;
    end;
    Writeln(fout,Top^.Data);
    CloseFile(fin);
    Closefile(fout);
  end.
```

При вивченні основ алгоритмізації доцільно ознайомити студентів з можливими реалізаціями стохастичних моделей під загальною назвою “Методи Монте-Карло”. В цих методах використовуються стандартні випадкові числа (такі, що рівномірно розподілені на інтервалі (0, 1)) для моделювання траєкторій різноманітних випадкових процесів на комп’ютері з метою оцінювання певних характеристик цих процесів. Для отримання таких чисел використовують генератори випадкових чисел. Існують фізичні генератори випадкових чисел, що базуються на використанні спеціальних приладів, та генератори псевдовипадкових чисел (спеціальні комп’ютерні програми).

Фізичний генератор представляє собою прилад, що генерує або не генерує сигнал в даний момент часу. Багатократне звертання до такого приладу дозволяє отримати послідовність нулів та одиниць (наявність сигналу дає одиницю, а його відсутність – нуль), з яких формується дробова частина двійкового подання випадкового числа (ціла частина цього числа дорівнює нулю). Певна складність полягає в тому, що для застосування методу Монте-Карло потрібен генератор, за допомогою якого можна отримувати нулі і одиниці в дробовій частині двійкового подання випадкового числа з однаковою ймовірністю [29].

В якості такого генератора може бути, наприклад, рулетка, на якій фіксують тільки колір сектору, в якому зупинилась кулька. Випадання чорного кольору, можна пов'язати з генерацією одиниці, тоді випадання червоного кольору буде означати генерацію нуля. Мабуть можливість такого способу отримання випадкових чисел і зумовила назву методів Монте-Карло (в цьому ігровому центрі рулетки досить популярні). Зрозуміло, що такий генератор практично незастосовний, оскільки дуже повільний і відсутня автоматизація отримання випадкових чисел. У реальних задачах використовують, наприклад, квантові генератори випадкових чисел.

Проблему рівної ймовірності появи нулів і одиниць досить часто розв'язують наступним чином. Сигнал вимірюють двічі. Можливі такі ситуація: сигнал був обидва рази (СС), сигналу не було жодного разу (НН), перший раз сигнал був, а другий – ні (СН) та перший раз сигналу не було, а другий раз – був (НС). Якщо навіть ймовірність появи сигналу не дорівнює 1/2, все рівно ситуації НС і СН є рівноймовірними. Тоді можна фіксувати тільки ці дві ситуації (наприклад пов'язавши з НС одиницю, а з СН – нуль), а ситуації СС та НН ігнорувати [29].

Використання фізичних генераторів в розрахунках за методами Монте-Карло має певні недоліки:

- такі генератори – досить недешеві прилади, використання яких потребує спеціального інтерфейсу з комп'ютером;
- потрібна постійна перевірка послідовностей, що генеруються, оскільки будь-який пристрій може давати збої.

Тому у більшості розрахунків за методами Монте-Карло використовуються генератори псевдовипадкових чисел, як правило такі генератори описані мовами програмування у вигляді стандартних функцій (наприклад *Random*) і базуються на методі Лемера. Слід зауважити, що для цього методу існує проблема циклічності – через певну кількість звернень послідовність псевдовипадкових чисел починає повторюватись.

Розглянемо, як здійснюється чисельне інтегрування за методом Монте-Карло (див. [110]). Для знаходження інтеграла (площі під графіком функції) застосовують наступний алгоритм (Рис. 4.21):

1) обмежують функцію прямокутником, площу якого легко обчислити;
2) вибирають випадковим чином (навмання) в прямокутнику певну кількість точок (n), координати яких отримують за допомогою датчика (генератора) випадкових чисел;
3) визначають кількість точок (k), які потрапили під графік функції;
4) інтеграл визначають за формулою $I=S_{прмокут}*k/n$.

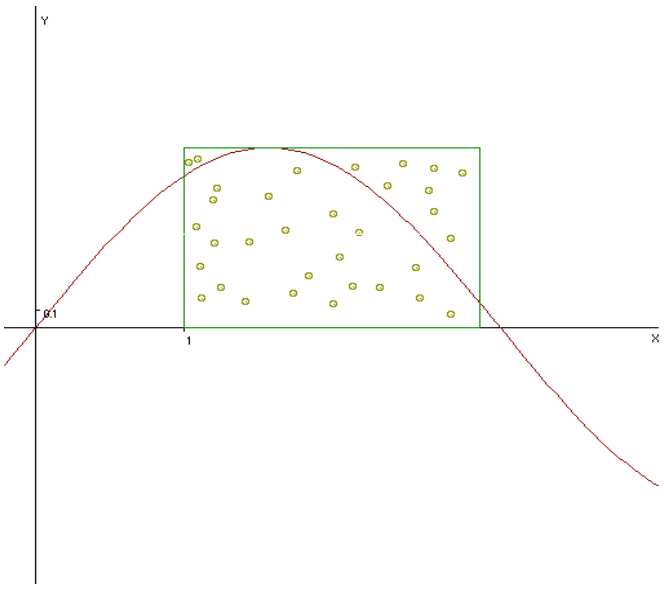

Рис. 4.21. Чисельне інтегрування за методом Монте-Карло.

Наведемо опис програми мовою Pascal, за якою реалізується наведений алгоритм. Відмітимо, що для генерації псевдовипадкових чисел у мові Pascal використовується функція *random*, за якою повертається дійсне псевдовипадкове число в діапазоні від 0 до 1. Для того, щоб таке число було в діапазоні від *c* до *d*, потрібно скористатися формулою *c+(d-c)*random*. Для визначеності знайдемо інтеграл від функції *y=sin(x)* на відрізку від 1 до 3.

```
    {через константи визначаються межі інтегрування та кількість
вибраних випадкових точок}
    const a=1;
    b=3;
    n=10000;
    {функція:}
    function f(x: Real): Real;
    begin
       f:=sin(x);
    end;
    begin
      {ініціалізуємо генератор псевдовипадкових чисел}
      randomize;
      {вибираємо n випадкових точок}
      for i:=1 to n do
      begin
          {обчислюємо x-координату випадкової точки}
          x:=a+b*random;
          {обчислюємо y-координату випадкової точки}
          y:=random;
         {якщо точка потрапила під графік, збільшуємо на 1
значення лічильника таких точок}
          if f(x)<=y then
             k:=k+1;
      end;
      {обчислюємо площу під графіком:}
      S:=(b-a)*2*k/n;
```

```
  writeln('S=',S:5:3);
end.
```

Цей алгоритм легко узагальнити на функцію кількох змінних. Якщо функція, що інтегрується, залежить від малої кількості аргументів, ефективність методу Монте-Карло нижча, ніж інших чисельних методів. Але в деяких випадках, коли функція задана неявно, а необхідно обчислити площу області, заданої складними нерівностями, стохастичний метод може виявитися більш ефективним. Також він стає більш переважним і у випадку, коли кількість аргументів функції велика, тому що обчислювальна складність методу Монте-Карло не дуже сильно залежить від кількості аргументів функції.

Інший підхід застосування методу Монте_Карло до обчислювання визначеного інтеграла наведено в [110, с.467-469]. Також в цьому посібнику розглянуто застосування методу Монте-Карло до розв'язування задачі Бюффона ([110, с.469-470]) та деяких інших цікавих задач.

4.3. Елементи змісту курсу "Математичне і комп'ютерне моделювання"

4.3.1. Інформаційне моделювання в педагогічних програмних засобах на прикладі методу найменших квадратів. Використання понять та методів інформатики дозволяє на якісно новому рівні вивчати не тільки власне інформатику, але і математику, розвиваючи у студентів компетентності щодо інформаційного моделювання при розв'язуванні математичних задач за допомогою спеціалізованих пакетів математичних програм. Оволодіння такими математичними пакетами у курсі інформатики дозволить студентам зосередити основну увагу якраз на етапі пошуку адекватної інформаційної моделі предметної галузі задачі, побудові на її основі математичної моделі та аналізу цієї моделі засобами математичної програми для встановлення числових характеристик об'єктів цієї моделі, взаємозв'язків між об'єктами, їх візуалізації без проведення

громіздких обчислень. Все це спонукує студентів до проведення досліджень, експериментів, що приводить до відповідних здогадок та відкриттів, привносить елементи творчості у їх навчальну діяльність.

Серед таких математичних програм слід відзначити програми Gran1, Gran2D та Gran3D, які досить широко використовуються у навчальному процесі школи та педагогічного університету в курсах математики і інформатики, і визнані як ефективні педагогічні інструменти. З ними працюють як педагоги-практики, так і педагоги-дослідники, існує велика кількість педагогічних досліджень і методичних матеріалів стосовно вивчення цих програм у курсі інформатики та застосування їх при вивченні математики та фізики у школі та педагогічному університеті ([4, 38, 43, 69, 75, 93, 95, 97, 98, 100, 101, 103, 104, 113, 114, 115, 126, 141, 184, 195, 196, 197, 243, 245, 303] та ін.).

Наведемо приклад того, як за допомогою програми Gran1 можна визначати і досліджувати математичні моделі результатів певних експериментів, застосовуючи метод найменших квадратів [59].

Важливою в експериментальних дослідженнях є проблема відшукання аналітичного виразу функції, значення якої якомога менше відхиляється від одержаних експериментально значень у заданих точках; тобто потрібно за заданим набором точок (x_i, y_i), $i = \overline{1..m}$ знайти функцію $f(x)$, значення $f(x_i)$ якої в точках x_i якомога менше відрізняється від заданого значення y_i [128, 155].

У багатьох випадках $f(x)$ шукають у вигляді полінома

$$f(x)=a_0+a_1x+a_2x^2+...+a_nx^n.$$

При $n \geq m$ (кількості експериментальних точок) можливо дібрати коефіцієнти так, щоб графік полінома проходив через всі експериментальні точки. Вирази для таких поліномів можна отримати, використовуючи інтерполяційні формули Лагранжа або Ньютона, причому степінь таких поліномів залежить від кількості експериментальних точок. Проте досить часто потрібно отримати поліном цілком певного степеня або аналітичну

функцію іншого виду. З іншого боку графік функції $y= f(x)$ не обов'язково повинен точно проходити через всі експериментальні точки, оскільки вони отримані, як правило, з певною похибкою.

Виникає задача: знайти функцію $y=f(x)$ певного наперед заданого виду і таку, графік якої проходить якомога ближче до експериментальних точок (x_i, y_i), $i = \overline{1..m}$. Метод найменших квадратів полягає у такому доборі аналітичного вигляду функції, щоб різниця квадратів ординат експериментальних точок та відповідних значень функції була мінімальною, тобто:

$$\sum_{i=1}^{m} (f(x_i) - y_i)^2 = S \to \min \qquad (1)$$

Як відомо, метод найменших квадратів розроблено К.Гаусом, ще коли йому було сімнадцять років.

Дані експерименту доцільно подати графічно, особливо в тих випадках, коли вид функціональної залежності наперед невідомий.

Виходячи з виду графіка або з деяких теоретичних міркувань, добирають емпіричну залежність, наприклад $y=a_1x+a_2$ або $y=a_1e^{a_2x}+a_3$, для якої і потрібно за методом найменших квадратів визначити значення коефіцієнтів a_1, a_2, a_3.

Нехай шукана функція залежить від n параметрів $a_1, a_2,..., a_n$:

$$y=f(x, a_1, a_2,..., a_n).$$

Тоді формула (1) набуде вигляду

$$\sum_{i=1}^{m} (f(x_i, a_1,..., a_n) - y_i)^2 = S \to \min \qquad (2)$$

Відшукання параметрів $a_1, a_2,...,a_n$ на основі методу найменших квадратів зводиться до розв'язування системи рівнянь, отриманих з (2) при дослідженні величини S на екстремум:

$$\frac{\partial S}{\partial a_1} = 0, \quad \frac{\partial S}{\partial a_2} = 0, \ ..., \ \frac{\partial S}{\partial a_n} = 0 .$$

Нехай потрібно отримати аналітичну функцію у лінійному вигляді, тобто $y=a_1+a_2x$, при цьому

$$S=\sum_{i=1}^{m}(a_1+a_2x_i-y_i)^2=(a_1+a_2x_1-y_1)^2+(a_1+a_2x_2-y_2)^2+...+$$

$$+(a_1+a_2x_m-y_m)^2.$$

Отримаємо систему рівнянь:

$$\frac{\partial S}{\partial a_1}=2(a_1+a_2x_1-y_1)+...+2(a_1+a_2x_m-y_m)=0,$$

$$\frac{\partial S}{\partial a_2}=2x_1(a_1+a_2x_1-y_1)+...+2x_m(a_1+a_2x_m-y_m)=0$$

Розділимо обидва рівняння на 2 і зведемо подібні доданки, отримаємо:

$$ma_1+(\sum_{i=1}^{m}x_i)a_2=\sum_{i=1}^{m}y_i \qquad (3)$$

$$(\sum_{i=1}^{m}x_i)a_1+(\sum_{i=1}^{m}x_i^2)a_2=\sum_{i=1}^{m}x_iy_i$$

Знайденими формулами можна скористатися при розв'язуванні конкретної задачі.

Задача 4.38. [155] В "Основах химии" Д.І.Менделєєва наводяться дані про розчинність азотнокислого натрію $NaNO_3$ в залежності від температури води. В 100 частинах води розчинюється наступна кількість умовних частин $NaNO_3$ при відповідних температурах:

Температура	0	40	100	150	210	290	360	510	680
Кількість умовних одиниць	66,7	71,0	76,3	80,6	85,7	92,9	99,4	113,6	125,1

Теоретичні міркування дають підстави для висновків, що кількісна сторона цього явища достатньо точно описується лінійною залежністю $y=a_1+a_2x$. Необхідно за методом найменших квадратів визначити коефіцієнти a_1 і a_2 даної лінійної функції. Для розв'язування скористаємося системою рівнянь (3), де:

$$\sum_{i=1}^{9} x_i = 234,\ \sum_{i=1}^{9} y_i = 811{,}3,\ \sum_{i=1}^{9} x_i^2 = 10144,\ \sum_{i=1}^{9} x_i y_i = 24695{,}3.$$

Розв’язавши систему, отримаємо $a_1=67{,}5$, $a_2=0{,}87$, тобто

$$y=67{,}5+0{,}87x$$

що і було доведено Мендєлєєвим у 1881 р.

Як видно з наведеної вище задачі, визначення за методом найменших квадратів параметрів (коефіцієнтів) навіть такої простої функції, як лінійна, вимагає значної кількості обчислень, тому при розв'язуванні таких задач доцільно застосовувати відповідні програмні засоби. Скористаємося програмою Granl. За допомогою цієї програми можна наближати (апроксимувати) за методом найменших квадратів таблично задані функції (таблиці експериментальних даних) поліномами до сьомого степеня включно.

Для побудови апроксимуючого поліному потрібно у вікні *"Список об'єктів"* встановити тип майбутнього об'єкта *"Таблична: Xi, Y(Xi)"* і скористатися командою *"Об'єкт/Створити..."*. На екрані з'явиться допоміжне вікно, в якому потрібно ввести таблицю значень аргументів та відповідних значень функції (таблицю експериментальних даних) та степінь полінома. На Рис.4.22 зображено заповнене даними розглянутої вище задачі допоміжне вікно.

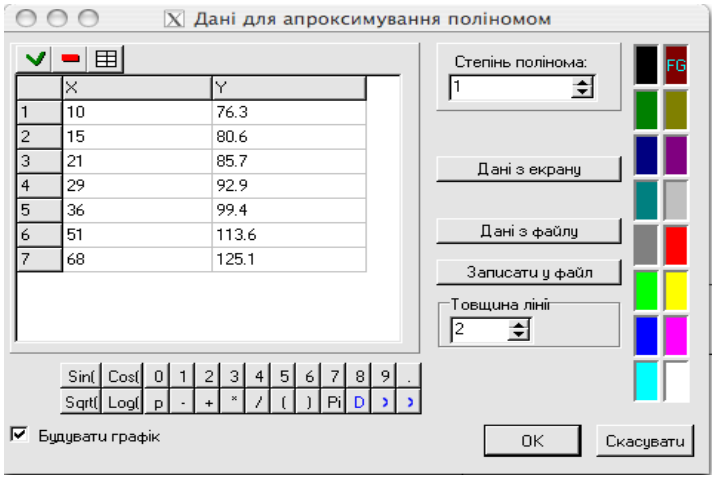

Рис. 4.22. Введення даних до Задачі 4.38.

Після “натиснення” кнопки *"OK"* визначається поліном, який записується до списку об'єктів (Рис. 3.23).

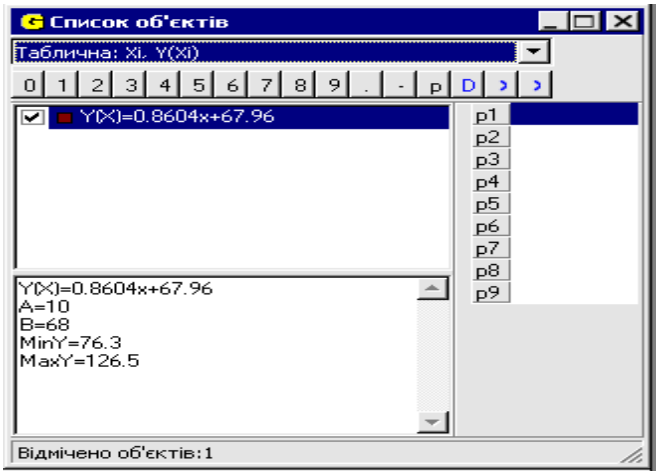

Рис. 4.23. Поліном до Задачі 4.38.

Якщо скористатися командою *"Графік/Побудувати"*, то отримаємо графік апроксимуючого полінома , а також зображення точок, вказаних у таблиці (Рис. 4.24):

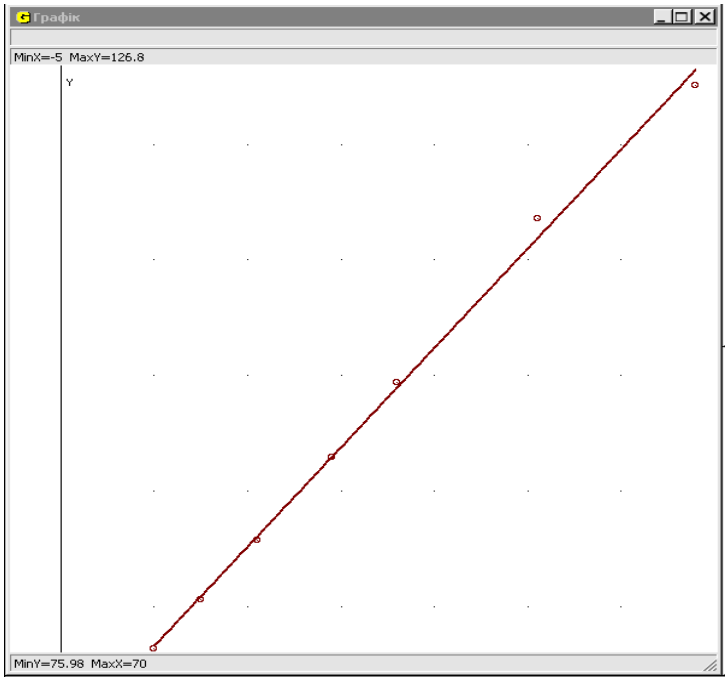

Рис.4.24. Графік апроксимуючого полінома до Задачі 4.38.

Крім введення даних з клавіатури, в програмі Granl передбачено і інші способи створення таблиці даних, а саме:

- завантажити з текстового файлу (числа в файлі повинні розділятися пропусками або починатися кожне з нового рядка), “натиснувши” кнопку *«Дані з файлу»*.

- за допомогою кнопки *«Дані з екрану»* перейти до вікна *«Графік»*, вказати точки, встановлюючи курсор мишки у необхідних місцях на екрані і щоразу натискуючи ліву клавішу мишки, і далі, “натиснути” кнопку *"OK"*.

Легкість визначення апроксимуючого поліному дозволяє студентам експериментувати, змінюючи його степінь та спостерігаючи, яким є його графік у кожному випадку.

Для підвищення творчої активності студентів доцільно запропонувати їм знайти альтернативний розв'язок класичної шкільної задачі про знаходження параболи, що проходить через три задані точки (x_1, y_1), (x_2, y_2), (x_3, y_3).

Традиційний спосіб розв'язування цієї задачі полягає в тому, що коефіцієнти параболи a, b, c можна визначити, розв'язавши систему рівнянь

$$(x_1)^2 a + x_1 b + c = y_1,$$

$$(x_2)^2 a + x_2 b + c = y_2,$$

$$(x_3)^2 a + x_3 b + c = y_3.$$

Альтернативний спосіб розв'язування полягає у застосуванні програми Granl для визначення поліному другого степеня за методом найменших квадратів.

Якщо для розв'язування деякої задачі виникає потреба апроксимувати не поліномом, а деякими іншими функціями, можна скористатися певними перетвореннями, як у задачі, наведеній нижче.

Задача 4.39. [344] Дано експериментальні дані:

x_i	0,1	0,2	0,3	0,4
y_i	2,5	5,1	9,8	20,0

Потрібно визначити вид відповідної функції, та її параметри.

Для визначення виду функції можна скористатися правилом, в якому говориться про те, що якщо значення аргументу утворюють точно або

майже точно арифметичну прогресію, а відповідні значення функції утворюють майже геометричну прогресію, то шукану залежність між аргументом і функцією потрібно шукати у вигляді

$y=a^{bx}+c$, $a>0$.

Зручно за основу брати число e.

Дані розглядуваної задачі задовольняють сформульоване вище правило, тому будемо шукати функцію у вигляді $y=e^{bx+c}$. Проте за програмою Granl відшукуються апроксимуючі функції тільки у вигляді поліномів, тому прологарифмуємо вираз шуканої функції і отримаємо $ln(y)=bx+c$. Введемо дані, записуючи $ln(y_i)$ замість y_i:

x_i	0,1	0,2	0,3	0,4
$ln(y_i)$	0,92	1,63	2,28	3,00

і встановимо степінь апроксимуючого полінома рівним *1*. Отримаємо $y=6,89x+0,235$. Таким чином, шуканою є функція $y=e^{6,89x+0.235}$.

Щоб впевнитися, що знайдена функція вдало апроксимує експериментальні дані, виконаємо наступні операції:

- вводимо таблицю експериментальних значень; оскільки апроксимуючий поліном зараз не береться до уваги, можна вказати його степінь *0*;
- вводимо функцію *y=exp(6.89*x+0.235)* на проміжку *[0; 0.4]*;
- будуємо графік цієї функції (Рис. 4.25).

Розв’язування таких задач дозволяє не тільки розвивати компетентності стосовно використання математичних програм, зокрема Gran1, для знаходження апроксимуючих функцій, але і у більш широкому сенсі познайомити студентів з інформаційними і відповідними їм математичними моделями з різних предметних галузей, де використовується апроксимація експериментально отриманих залежностей, формувати задатки вченого-дослідника, формувати міжпредметні зв’язки інформатики з математикою, фізикою, хімією,

біологією і іншими науками, де використовується опрацювання результатів експериментів.

Задача 4.40 [94, с. 554]. Метод найменших квадратів для побудови регресійної моделі.

Під регресією будемо розуміти статистичний зв'язок між випадковими величинами. Прості лінійні регресійні моделі встановлюють лінійну залежність між двома змінними: витратами на рекламу та обсягом продукції; витратами на відпустку та складом родини; витратами на споживання та валовим національним продуктом (ВНП); ціною і попитом тощо. При цьому одна із змінних вважається залежною (y) та розглядається як функція незалежної змінної (x – фактор).

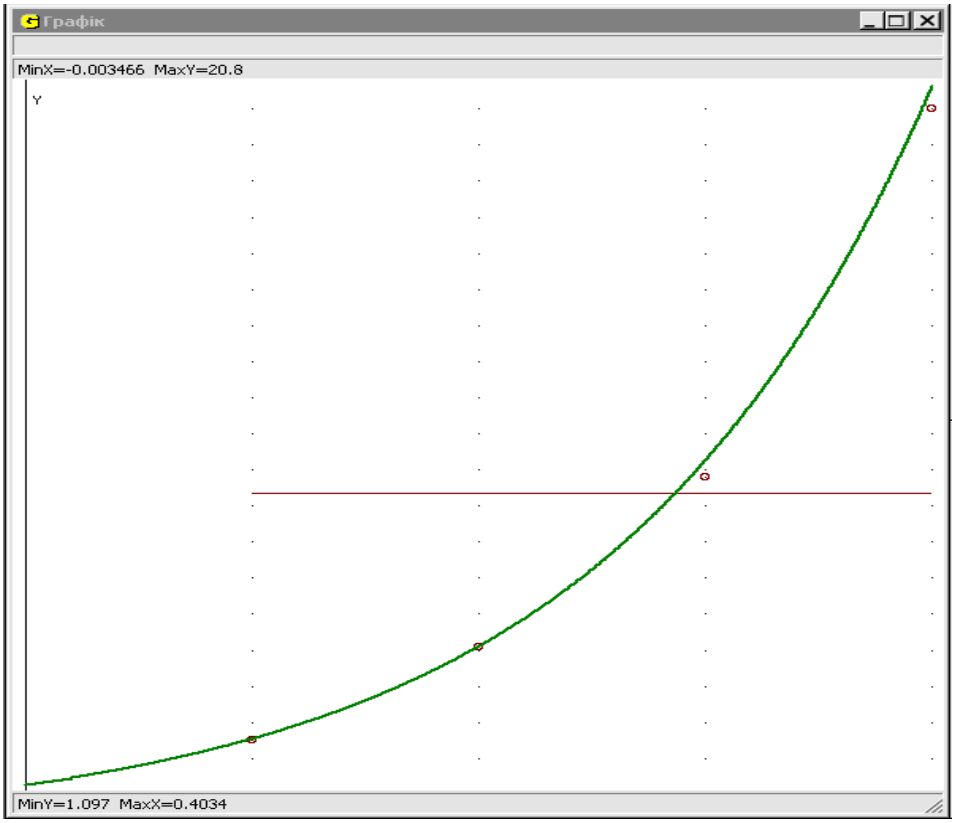

Рис. 4.25. Графік функції до Задачі 4.39.

У загальному вигляді лінійна регресійна модель записується так:

$$Y=aX+b$$

Нехай є набір спостережених значень змінної $y:(y_1, ..., y_n)$; набір спостережених значень змінної $x:\{x_1, ..., x_n\}$; a, b – невідомі коефіцієнти регресійної моделі.

Найчастіше коефіцієнти залежності $Y=aX+b$ добирають так, щоб сума квадратів відхилень $y_i-(ax_i+b)$ була найменшою, тобто щоб досягався

$$\min_{a,b}\sum_{i=1}^{n}(ax_i+b-y_i)^2 .$$

Прирівнюючи до нуля частинні похідні квадратичної функції $f(a, b)$ за змінними a та b, отримуємо:

$$\frac{\partial}{\partial a}\sum_{i=1}^{n}(ax_i+b-y_i)^2=2\sum_{i=1}^{n}(ax_i+b-y_i)x_i=0,$$

$$\frac{\partial}{\partial b}\sum_{i=1}^{n}(ax_i+b-y_i)^2=2\sum_{i=1}^{n}(ax_i+b-y_i)=0$$

або

$$a(\sum_{i=1}^{n}x_i^2)+b(\sum_{i=1}^{n}x_i)-\sum_{i=1}^{n}x_iy_i=0,$$

$$a(\sum_{i=1}^{n}x_i)+b\cdot n-\sum_{i=1}^{n}y_i=0.$$

Помноживши перше рівняння на n, друге на $\sum_{i=1}^{n}x_i$ і віднявши друге від першого, одержимо

$$a(n\sum_{i=1}^{n}x_i^2-(\sum x_i)^2)=n\sum_{i=1}^{n}x_iy_i-\sum_{i=1}^{n}x_i\sum_{i=1}^{n}y_i,$$

звідки

$$a=\frac{n\sum_{i=1}^{n}x_iy_i-\sum_{i=1}^{n}x_i\sum_{i=1}^{n}y_i}{n\sum_{i=1}^{n}x_i^2-(\sum x_i)^2}=\frac{\frac{1}{n}\sum_{i=1}^{n}x_iy_i-\frac{1}{n^2}\sum_{i=1}^{n}x_i\sum_{i=1}^{n}y_i}{\frac{1}{n}\sum_{i=1}^{n}x_i^2-(\frac{1}{n}\sum x_i)^2}=$$

$$=\frac{\overline{M}[XY]-\overline{M}[X]\overline{M}[Y]}{\overline{M}[X^2]-(\overline{M}[X])^2}=\frac{\overline{K}[X,Y]}{\overline{D}[X]}=\frac{\tilde{K}[X,Y]}{(\tilde{\sigma}[X])^2},$$

$$b=\frac{1}{n}\sum_{i=1}^{n}y_i-\frac{\overline{K}[X,Y]}{(\overline{\sigma}[X])^2}\frac{1}{n}\sum_{i=1}^{n}x_i=\overline{M}[Y]-\frac{\overline{K}[X,Y]}{(\overline{\sigma}[X])^2}\overline{M}[X].$$

Оскільки квадратична функція $\sum_{i=1}^{n}(ax_i+b-y_i)^2$ опукла донизу, то в знайденій точці *(a, b)* досягається її мінімум.

Аналогічно за методом найменших квадратів можна отримати рівняння прямої регресії X на Y.

Задача 4.41 [94]. Розглянемо конкретну економічну модель. Бюро економічного аналізу фірми *X* оцінює ефективність відділу маркетингу з продажу цукерок. Для такої оцінки вони мають досвід роботи у 5-ти географічних зонах з майже однаковими умовами (потенційні клієнти, ставлення до товарного знаку та ін.). У цих зонах фіксували x_i – обсяги продажу протягом деякого періоду (млн. коробок) і y_i – витрати на рекламу (млн. грн.) для просування товару на ринку:

i (№зони)	Xi(млн.кор)	Yi(млн.грн)
1	5	25
2	6	30
3	9	35
4	12	45
5	18	65

Можна припустити, що між двома змінними є лінійна залежність, тобто їх можна апроксимувати прямою лінією.

Необхідно побудувати лінійну модель $y=ax+b$, яка найкращим чином описує спостережені значення, тобто для таблично заданої функції

x_i	5	6	9	12	18
y_i	25	30	35	45	65

побудувати поліном 1-го степеня виду $y=P(x)=ax+b$, що найкраще наближає задану функцію в розумінні середнього квадратичного (за методом найменших квадратів).

Скориставшись програмою Gran1, в результаті одержимо, що рівняння шуканої прямої, яка описує лінійну залежність між продажем продукції і витратами на рекламу, має вигляд $y=3x+10$ (Рис. 4.26).

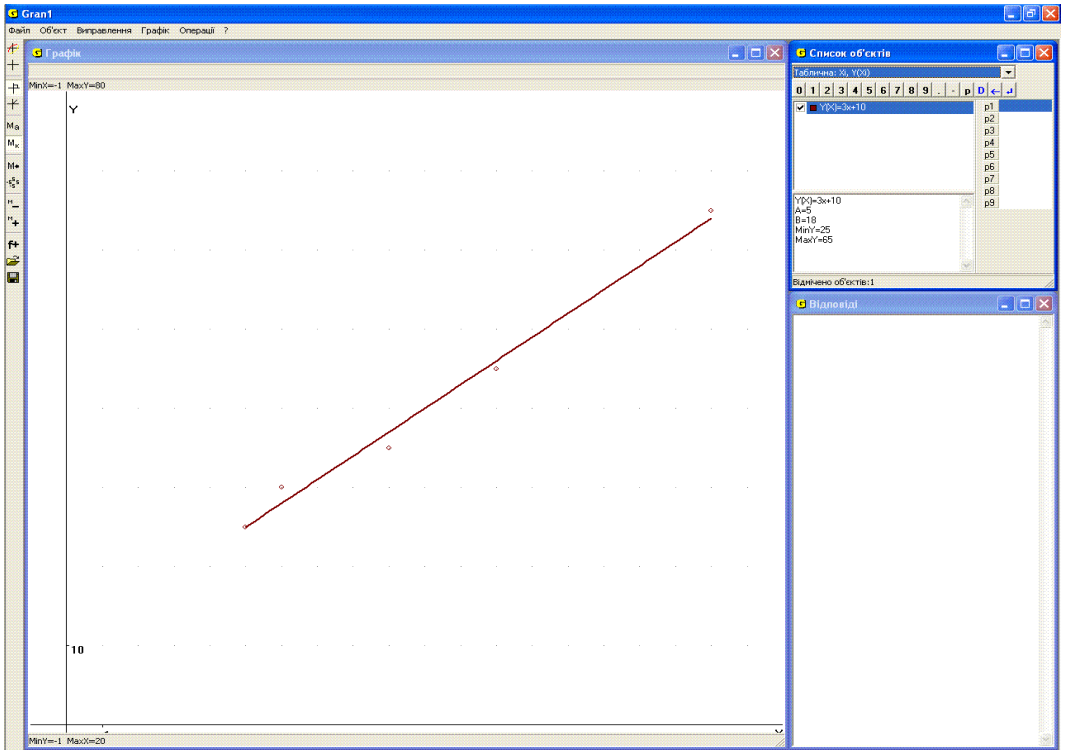

Рис. 4.26. Відповідь до Задачі 4.41.

4.3.2. Інформаційне моделювання та чисельні методи. Як було зазначено раніше, в результаті аналізу задачі будується її математична модель. На основі чисельних методів породжуються алгоритми, використання яких для аналізу відповідних математичних моделей дозволяє отримувати розв'язки задач, в основному наближені. Ці алгоритми реалізовано в багатьох математичних програмах, і студент, який оволодів такими програмами, наприклад Gran1, Maxima, Derive, Maple, Mathematica, MathCad, MathLab і ін., зможе за їх допомогою розв'язувати рівняння, системи рівнянь, обчислювати визначені інтеграли, будувати графіки функцій і т.д. Проте при цьому методи розв'язування задач можуть залишатися невідомими. А щоб оволодіти цими методами, студент повинен знати і вміти сам описувати ці алгоритми однією з мов програмування. Це може бути спеціалізована мова програмування,

вбудована в потужні математичні пакети, такі як Mathematica або Maple, а може бути універсальна мова програмування, наприклад Object Pascal або Free Pascal. Тому при вивченні курсу математичного та комп'ютерного моделювання певна кількість практичних завдань пов'язана з реалізацією основних алгоритмів чисельних методів за допомогою універсальної мови програмування з використанням математичних пакетів, як допоміжних засобів.

Наприклад одним із завдань може бути чисельне розв'язування рівнянь за методом дихотомії, методом хорд, методом дотичних, методом ітерацій тощо. Створена студентом програма повинна бути придатною для відшукання коренів різних рівнянь без модифікації коду програми. Тому за програмою повинні проводитися обчислення за введеним користувачем виразом.

Задача лексичного аналізу математичних виразів є досить складною для студентів, тому студентам пропонується використовувати в їхніх програмах модуль програми Gran1 під назвою *NGFunc*. В цьому модулі описано клас *TFunc*, призначений для зберігання даних про вираз функції, лексикографічного аналізу виразу, обчислення значень цього виразу. Даний клас містить конструктор *Create(SF: String)*. Він має єдиний параметр – вираз функції.

Наведемо приклад створення об'єкту типу TFunc, що міститиме вираз '*x*':

```
Var f: TFunc;
…
f:=TFunc.Create('x');
```

За функцією

```
function Eval(ArgX, ArgY: Extended): Extended;
```

повертається значення виразу для вказаних значень аргументів. Оскільки в програмі Gran1 вирази бувають як з однією, так і з двома змінними, то для

даної функції задаються значення двох аргументів. Якщо вираз містить тільки одну змінну, значення другого аргументу ігнорується. Якщо у процесі обчислення значення виразу виникла помилка (наприклад ділення на нуль, добування квадратного кореня із від'ємного числа тощо), за функцією *Eval()* повернеться значення константи *AllErr=1E+100* за допомогою опрацювання виключень. Це дозволяє аналізувати в подальшому такі помилки.

Процедура

```
procedure ChangeFunc(SF: String; var Chng: Boolean); virtual;
```

призначена для зміни виразу; в неї передається параметр *SF: String,* що містить новий вираз для об'єкту, а повертається значення параметра *var Chng: Boolean.* Якщо параметр *Chng* набув значення *True*, то зміна виразу відбулася, якщо ж цей параметр набув значення *False,* вираз не змінився (у випадку некоректності виразу *SF*).

Більш детально клас *TFunc* студенти вивчають у курсі дисципліни "Технології програмування та створення ППЗ".

Крім того студент повинен створити зручний графічний інтерфейс програми, в якому передбачалися б відповідні поля для введення виразу функції, для якої будуть шукатися нулі, виразів для кінців відрізка, на якому ці нулі шукатимуться, та точності обчислень, візуального об'єкта для виведення відповіді, кнопки, пов'язаної з процедурою пошуку кореня (коренів). Можна запропонувати студентам два варіанти завдання, виконання яких оцінюється різною кількістю балів. В простому варіанті передбачається написання програми, за якою відшукується корінь на такому відрізку, де він єдиний. В більш складному варіанті передбачається введення такого відрізка, де коренів може бути кілька. Математичні програми (Gran1, Maxima і ін.) використовуються студентом для відокремлення коренів та для перевірки правильності обчислень (потрібно результати, отримані за студентською програмою, порівняти з

результатами, отриманими за математичними програмами Gran1, Maxima чи іншими).

Аналогічний підхід повинен використовуватися при розв'язуванні завдань на дослідження алгоритмів чисельного інтегрування.

Розв'язування таких задач дозволить розвивати у студентів компетентності стосовно основ алгоритмізації та об'єктно-орієнтованого програмування, побудови математичних моделей та програм зі зручним інтерфейсом користувача для аналізу цих моделей; з'ясовувати особливості і відмінності різних математичних моделей, відповідних тим чи іншим чисельним методам, аналізувати і використовувати міжпредметні зв'язки математики та інформатики.

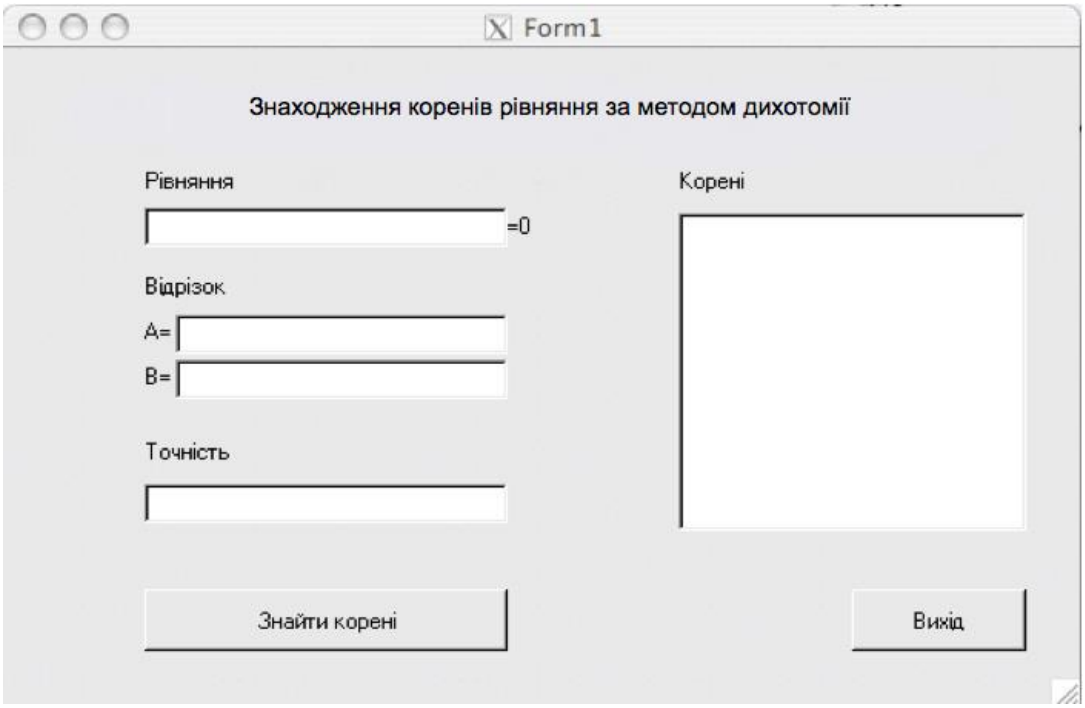

Рис. 4.27. Вид вікна програми для використання методу дихотомії.

Наведемо приклади завдань для написання програм для аналізу математичних моделей за деякими чисельними методами.

1. Розробити програму для відшукання коренів рівняння *f(x)=0* за методом:

а) дихотомії (ділення відрізка пополам);

б) методом ітерацій;

в) методом хорд;

г) методом дотичних;

д) комбінованим методом.

В програмі повинен бути зручний графічний інтерфейс користувача. У вікні програми необхідно передбачити поля для введення виразу функції, введення виразів кінців відрізка, введення точності обчислень та, якщо це вимагається за методом, додаткові поля (всі поля типу *TEdit*). Знайдені корені потрібно вивести у поле типу *TMemo*. Передбачити кнопки “Знайти корені” та “Вихід”. Вид вікна програми для використання методу дихотомії показано на Рис. 4.27.

Для роботи з функціональними виразами потрібно під’єднати до програми модуль *TFunc*.

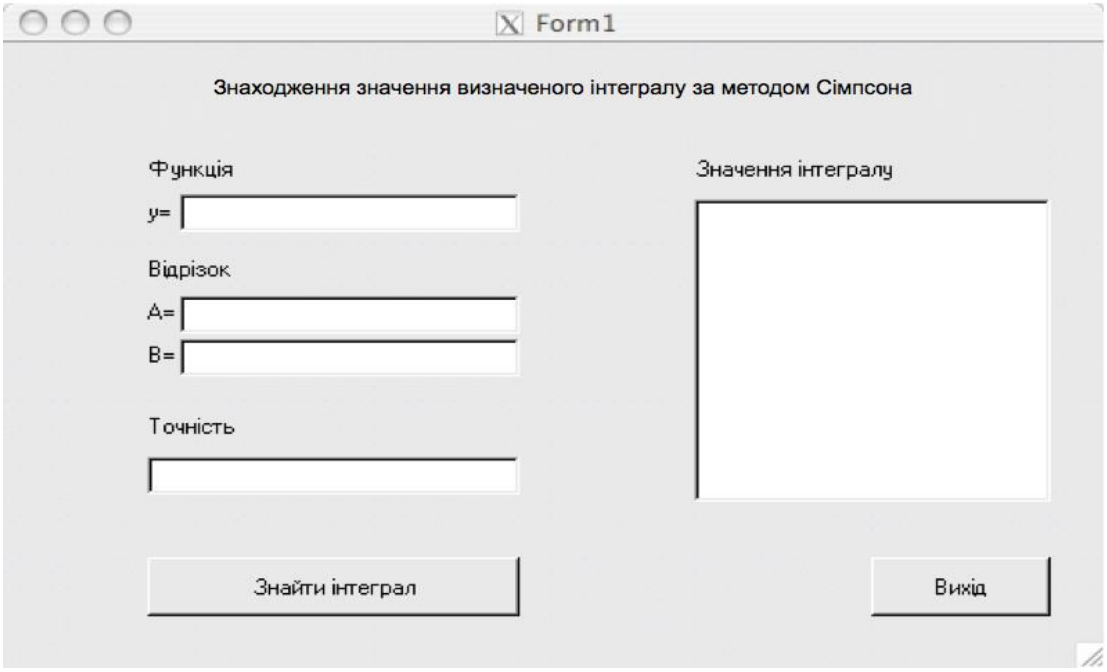

Рис. 4.28. Вид вікна програми для використання методу Сімпсона.

2. Розробити програму для обчислення значення визначеного інтеграла від функції *y=f(x)* на відрізку *[a, b]* за методом:

а) лівих прямокутників;

б) правих прямокутників;

в) трапецій;

г) Сімпсона (парабол);

д) Монте-Карло.

Програма повинна бути оснащена зручним графічним інтерфейсом користувача. У вікні програми передбачити поля для введення виразу функції, введення виразів кінців відрізка, введення точності обчислень (всі поля типу *TEdit*). Значення інтеграла потрібно вивести у поле типу *TMemo*. Передбачити кнопки “Знайти інтеграл” та “Вихід”. Вікно програми повинно мати вигляд, показаний на Рис. 4.28.

Для роботи з виразами функцій потрібно під’єднати до програми модуль *TFunc*.

4.3.3. Інформаційне моделювання в біології. Моделі популяції. Питання вивчення моделей популяції в біології з допомогою використання табличного процесора розглянуто в посібнику І.О. Теплицького [274]. Нижче запропоновано підхід до розгляду цих питань за допомогою програми Gran1.

Популяція – сукупність організмів, що займають обмежений ареал, мають спільне походження за фенотипом та географічно ізольовані від інших популяцій даного виду. Важливим питанням вивчення популяції є дослідження збільшення чи зменшення її чисельності – співвідношення народжуваності і смертності та динаміки цього збільшення чи зменшення чисельності.

Існує досить багато математичних моделей, що використовуються для опису динаміки росту популяції для різних природних умов та інших факторів.

Лінійні моделі

Спочатку розглянемо дискретні лінійні моделі. Дані в цих моделях подано у вигляді таблиці, перший рядок якої – моменти часу, а другий – чисельність популяції у відповідний момент часу:

t_i	t_1	t_2	...	t_k
N_i	N_1	N_2	...	N_k

Будемо вважати, що кількість осіб, що народилися і померли, пропорційна до об’єму популяції, тобто

$N_t=N_{t-1}+aN_{t-1}-bN_{t-1}$

де a – коефіцієнт народжуваності, b – коефіцієнт смертності, $a \in [0, 1]$, $b \in [0, 1]$,

або

$N_t=mN_{t-1}$, $m=1+a-b$,

або

$N_t=m^tN_0$

Таким чином отримується геометрична прогресія. При $m<1$ прогресія буде спадною, тобто чисельність популяції буде зменшуватися, при $m=1$ чисельність популяції буде сталою, а при $m>1$ чисельність популяції буде зростати.

Продемонструємо цю залежність за допомогою програми Gran1. Для цього в одній системі координат побудуємо графіки функцій *y=p1^x*p2*, *y=1^x*p2*, *y=p3^x*p2*. В першій і третій функціях значення *m* задане як параметр. Змінюючи значення цих параметрів, можна спостерігати за зростанням та спаданням чисельності популяції. Як параметр задано і початкову чисельність популяції, у всіх трьох функціях він однаковий. Оскільки досліджуються дискретні величини, то у властивостях функцій потрібно вказати, що графік будується точками (у прикладі таких точок 15) (Рис. 4.29).

Значення параметра $m=1$ називають критичним, або біфуркаційним.

При переході параметра через це значення характер динаміки якісно змінюється: від зростання чисельності популяції до спадання або навпаки.

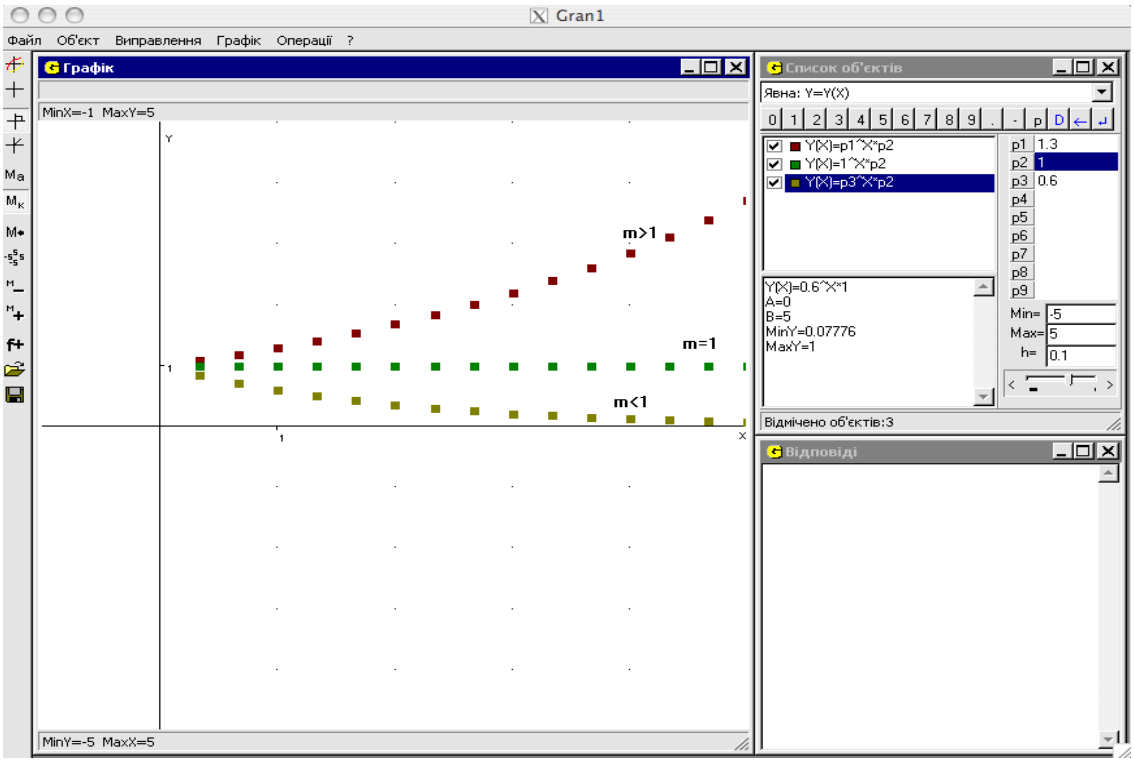

Рис.4.29. Види графіків для лінійної моделі.

Якщо певна популяція тварин вимирає *(m<1)*, то її поповнюють, ввозячи тварин, наприклад, з інших територій. В цьому випадку математична модель виглядатиме так:

$N_t=mN_{t-1}+g,$

де g – кількість тварин що додаються до популяції в кожний момент часу.

Тоді

$N_t=m^tN_0+g(1-m^t)/(1-m)$

Якщо визначити границю цього виразу при $t\to0$, і врахувавши, що $m<1$, отримаємо:

$N=g/(1-m)$

N – стаціонарне значення, до якого прямує чисельність популяції. Легко помітити, що це значення не залежить від початкової чисельності популяції N_0. Таку модель називають моделлю з притоком.

Якщо $m>1$, то спостерігатиметься збільшення чисельності популяції, що називають демографічним вибухом. Для регулювання чисельності популяцій тварин з демографічним вибухом застосовують відлов тварин. Математична модель в цій ситуації виглядатиме так:

$$N_t=mN_{t-1}-g,$$

де g – кількість тварин що виловлюються. Тоді

$$N_t=m^tN_0-g(m-1)/(m-1)$$

Ця модель називається моделлю з відтоком.

Скористаємося програмою Gran1 і введемо вираз

*p1^x*p2−p4*(p1^x−1)/(p1−1),* де $g=p4$.

Змінюючи параметр *p4*, помічаємо, що існує певне значення цього параметра, при якому популяція стабільна (Рис. 4.30), при значеннях параметра, більших за це значення, відбувається демографічний вибух, а при значеннях, менших від нього, популяція гине.

Тому потрібен більш тонкий механізм управління чисельністю популяції. Ним може бути механізм із зворотнім зв'язком, коли g є функцією від N_t:

$$g(N_t)=g_0+k(N_t-N)$$

Тут N – бажана чисельність популяції, g_0 – постійна складова відлову, $k(N_t-N)$ – коефіцієнт, що дає можливість змінювати рівень відлову в залежності від відхилення реальної чисельності популяції від бажаного рівня. Отримаємо наступну математичну модель чисельності популяції:

$$N_t=mN_{t-1}-g_0-k(N_{t-1}-N)$$

або

$$N_t=(m-k)N_{t-1}+kN-g_0$$

Щоб бажана чисельність популяції стала положенням рівноваги, потрібно покласти

$$g_0=(m-1)N$$

тоді

$$N_t=(m-k)N_{t-1}+(1-m+k)N$$

або

$$N_t=(m-k)^tN_0+(1-m+k)N(1-(m-k)^t)/(1-m+k)$$

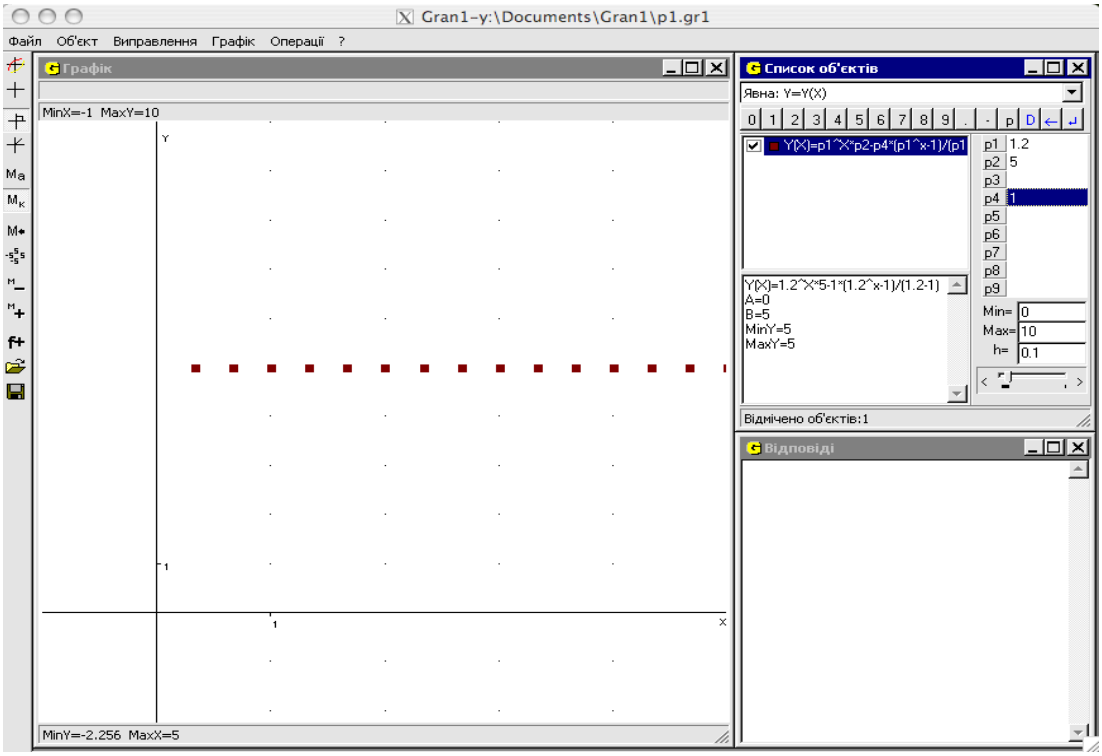

Рис. 4.30. Графік стабільної чисельної популяції.

Позначимо в програмі Gran1 $m=p1$, $N_0=p2$, $k=p3$, $N=p4$ і введемо функцію $(p1-p3)^x*p2+(1-p1+p3)*p4*(1-(p1-p3)^x)/(1-p1+p3)$ на відрізку від 0 до 20, причому виберемо пункт *"Графік точками"* і встановимо кількість точок побудови 20. Встановимо $m=p1=0.3$ (популяція вимирає), $N_0=p2=50$ (початкова чисельність популяції), $N=p4=20$ (бажана чисельність популяції). Будемо досліджувати, як змінюється чисельність популяції при зміні $k=p3$.

Якщо $0<m-k<1$, то чисельність популяції монотонно наближається до значення рівноваги N (Рис. 4.31):

Якщо $-1<m-k<0$, то чисельність популяції теж наближається до значення рівноваги N, але не монотонно, а так, як на Рис. 4.32:

Якщо $m-k=-1$, то чисельність популяції набуває по черзі значень N_0 та $2N-N_0$ (Рис. 4.33). Відмітимо, що в порівняні з попередніми експериментами

змінено значення $N=p4$, його значення покладено рівним 30, щоб популяція не вимерла.

При $m-k<-1$ спостерігається наступна картина (Рис. 4.34).

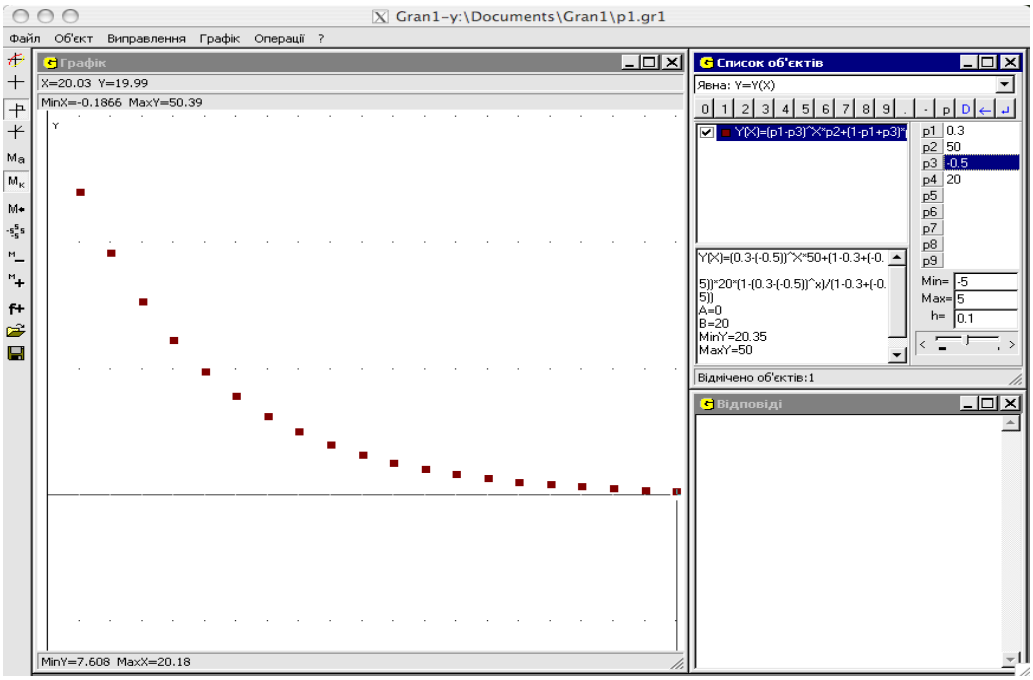

Рис. 4.31. Монотонне наближення чисельної популяції до значення рівноваги.

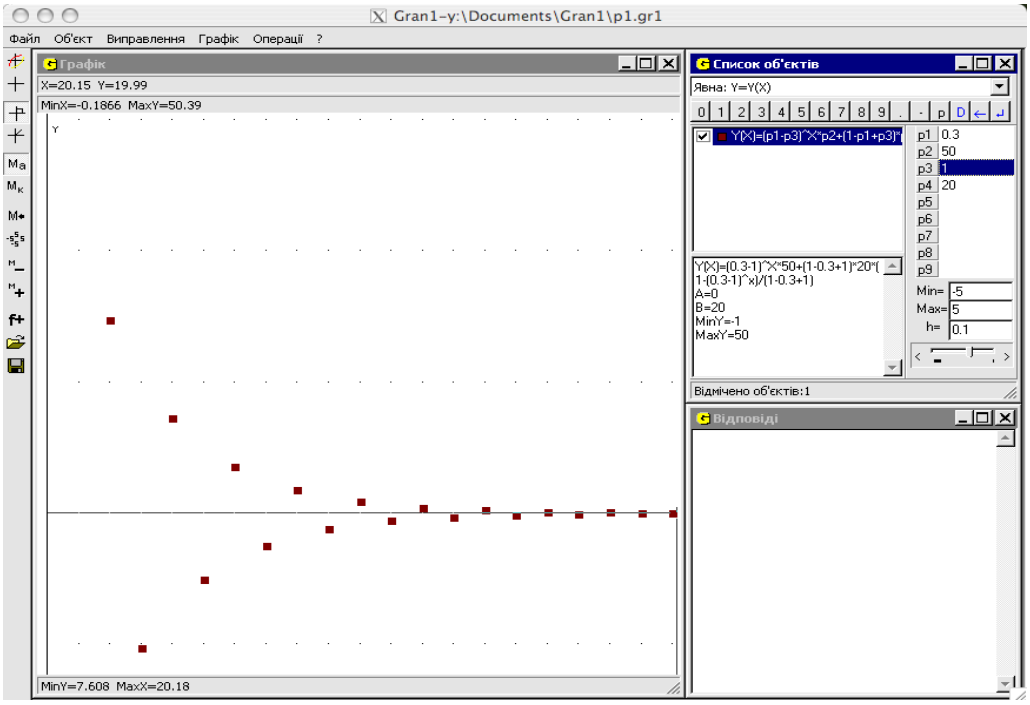

Рис. 4.32. Немонотонне наближення чисельної популяції до значення рівноваги.

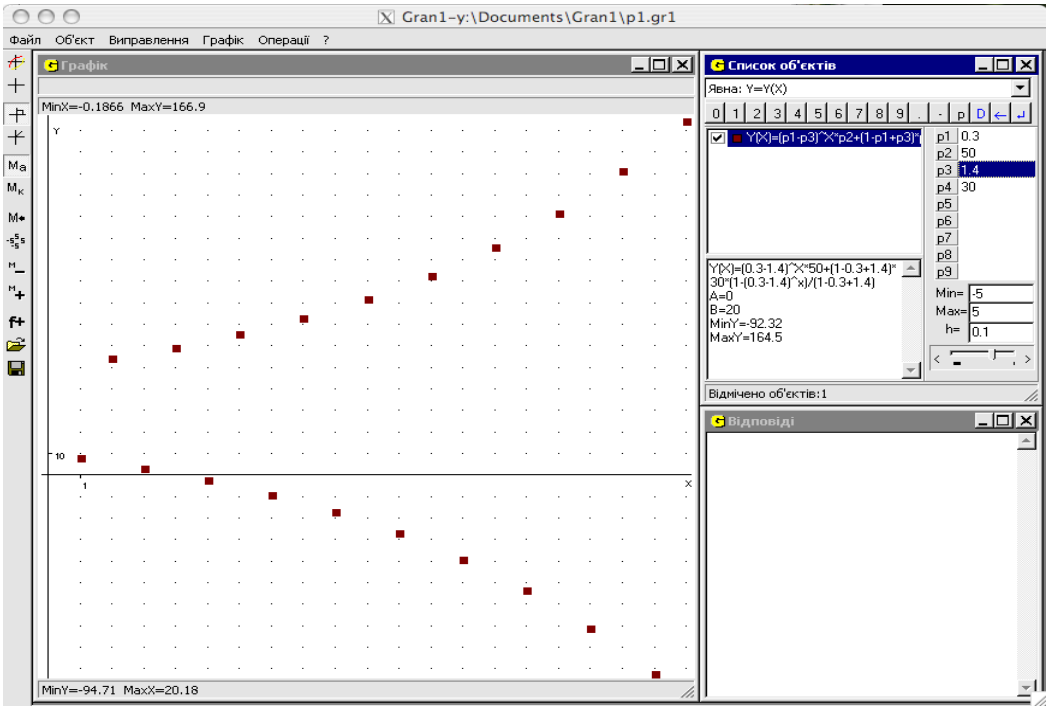

Рис. 4.33. Чисельність популяції по черзі набуває двох значень.

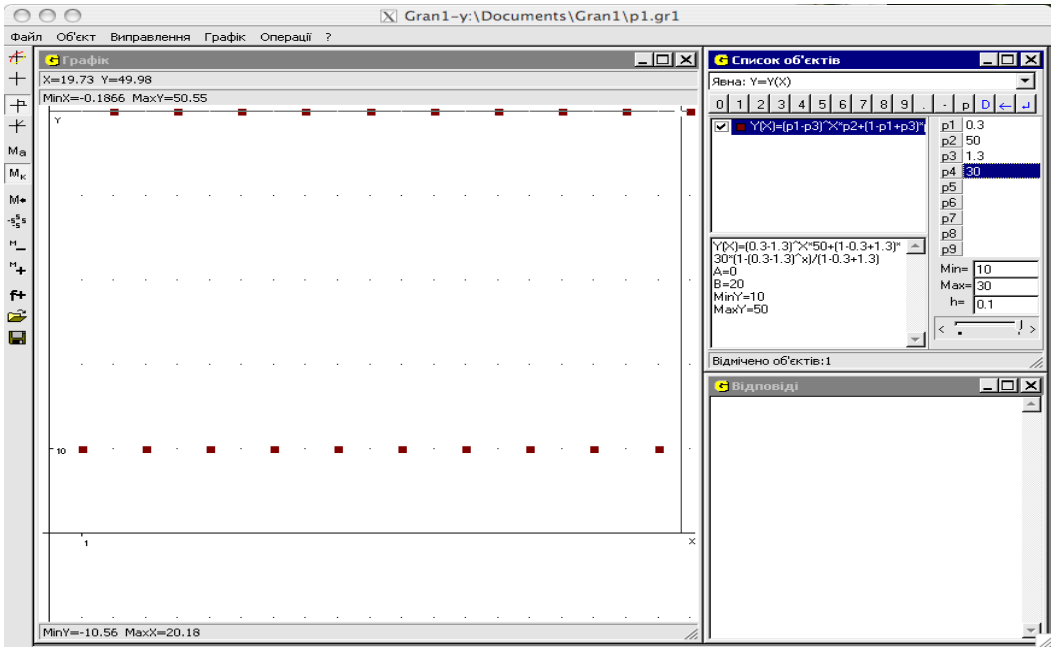

Рис. 4.34. Чисельність популяції по черзі то збільшується, то зменшується.

Одновимірна модель динаміки чисельності популяції

Розглянуті вище моделі були дискретними. Розглянемо тепер неперервні моделі. Спочатку розглянемо ситуацію, коли ресурси для

популяції необмежені. Тоді, якщо позначити через y чисельність популяції в момент t, через a – коефіцієнт народжуваності, через b – коефіцієнт смертності, отримаємо

$$y'=ay-by.$$

Якщо позначити $m=a-b$ – коефіцієнт природного приросту, отримаємо

$$y'=my.$$

Розв’язком цього рівняння буде

$$y(t)=e^{mt}y_0.$$

Скористаємося програмою Gran1 для дослідження цієї функції, позначивши m через $p1$, y_0 через $p2$.

Змінюючи параметр $m=p1$, можна помітити, що:

1. При $m<0$ чисельність популяції прямує до нуля, тобто при перевищенні смертності над народжуваністю популяція вимирає.
2. При $m=0$ чисельність популяції залишається сталою завдяки балансу між народжуваністю і смертністю.
3. При $m>0$ чисельність популяції експоненційно зростає – спостерігається демографічний вибух (Рис. 4.35).

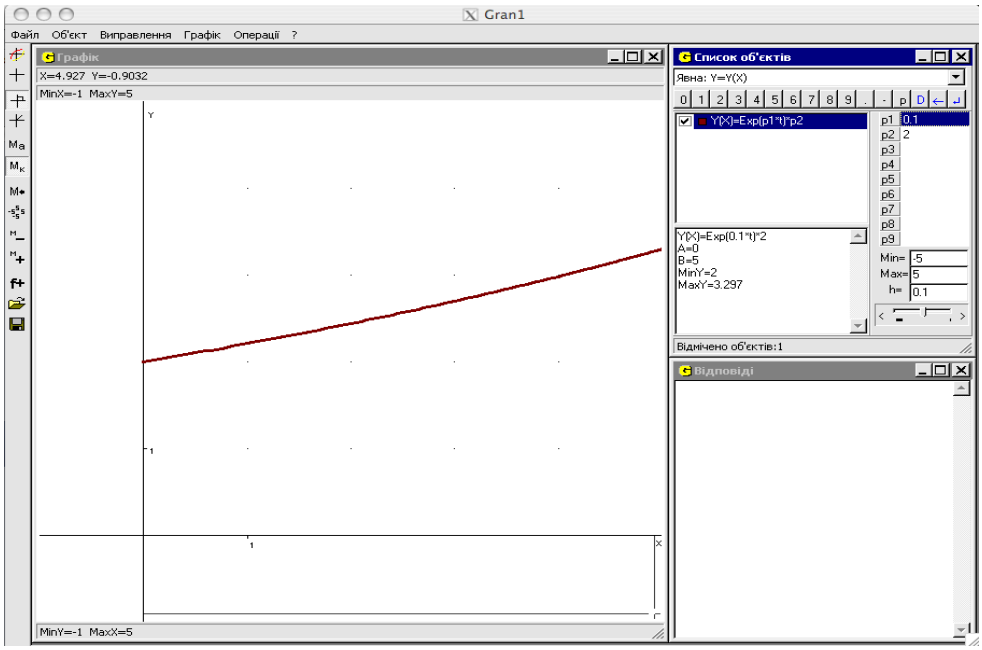

Рис. 4.35. Демографічний вибух.

Ця модель була розглянута Мальтусом в 1798 р. Він помітив що зростання чисельності популяції (експоненційне) перевищує зростання кількості продуктів харчування (яке тоді змінювалося лінійно), з чого зробив висновок, що в результаті цього через певний час наступить голод, а тому запропонував обмежити народжуваність.

Проте існують фактори, які не дозволяють популяції зростати нескінченно. Серед таких факторів може бути недостатність якого-небудь ресурсу (ресурсів), наприклад харчів. що призводить до конкуренції за цей ресурс, і в результаті популяція виходить на певний стаціонарний рівень. Для опису такої ситуації використовують модель Ферхюльста.

Модель Ферхюльста

Практично завжди у природі ресурси обмежені (як харчові, так і територіальні). Обмеженість ресурсів призводить до внутрівидової конкуренції. Поки чисельність популяції мала, то конкуренція не впливає на швидкість зростання популяції *m*. Коли ж чисельнісь зростає і наближається до певного граничного значення *K*, швидкість зростання популяції падає до нуля. Значення *K* називають ємністю екологічної ніші популяції. Ця величина відповідає такій чисельності популяції, при якій швидкість розмноження в результаті конкуренції так знижується, що популяція в кожному новому поколінні може тільки відновлювати свою чисельність.

Якщо припустити, що залежність швидкості росту популяції від її чисельності лінійна, отримаємо таке рівняння:

$y'=y(m-my/K)$.

Це рівняння отримало назву “рівняння логічного росту” або “рівняння Ферхюльста”. В цьому рівнянні швидкість народжуваності (*m*) не залежить від часу і чисельності популяції, а смертність пропорційна чисельності популяції. Збільшення смертності з ростом чисельності популяції може відбуватися завдяки скупченості особин популяції та зростаючій конкуренції між ними за харчові ресурси. Якщо розкрити дужки:

$y'=my-my^2/K$

то перший доданок буде відповідати необмеженому росту чисельності популяції, а другий – впливу міжвидової конкуренції (негативному впливу взаємодії двох особин одного виду) на ріст чисельності популяції.

Розв’яжемо це диференціальне рівняння і отримаємо:

$y(t)=Ky_0e^{mt}/(K-y_0+y_0e^{mt})$

Скористаємося програмою Gran1 для побудови графіка отриманої функції, позначивши m через $p1$, y_0 через $p2$, K через $p3$:

При $t\to+\infty$ чисельність популяції прямує до значення K, тобто до величини екологічної ніші, причому, якщо $y0>K$, тобто чисельність популяції в початковий момент часу більша за величину екологічної ніші, то чисельність популяції буде зменшуватися до K (Рис. 4.36).

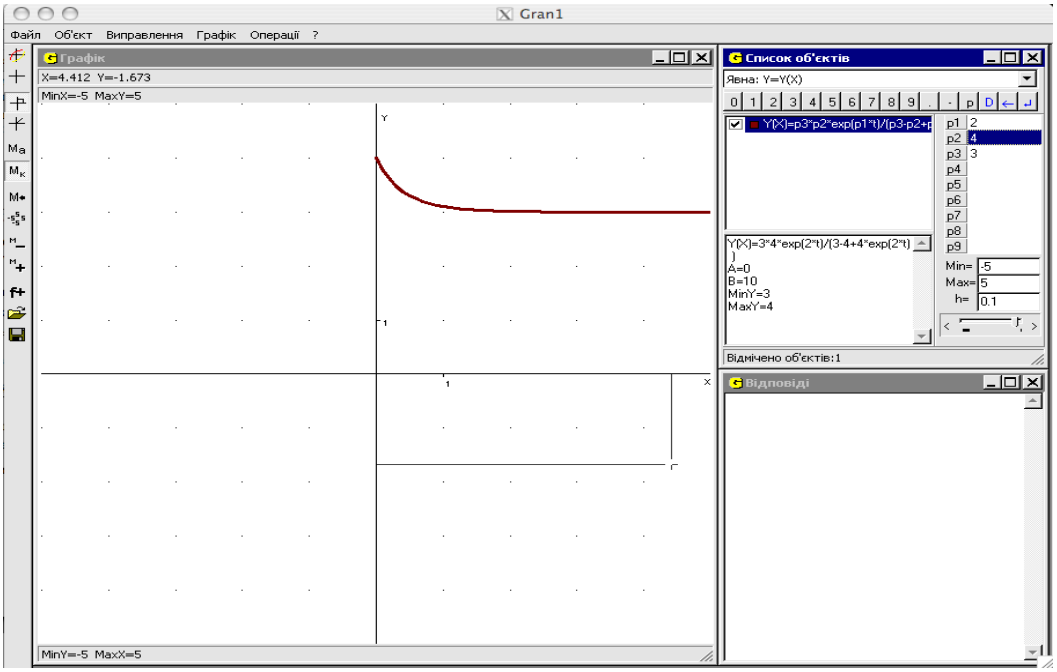

Рис. 4.36. Чисельність популяції зменшується до демографічної ніші.

Якщо початкова чисельність популяції менша величини екологічної ніші, тобто $y_0<K$, то з часом її чисельність буде рости, наближаючись до свого граничного значення K (Рис. 4.37), якщо при цьому початкова чисельність популяції складає менше половини екологічної ніші, тобто $y_0<K/2$, то на початковому етапі швидкість росту чисельності популяції буде

збільшуватися, поки чисельність популяції не досягне *K/2*, а потім почне знижуватись, прямуючи до нуля (Рис. 4.38).

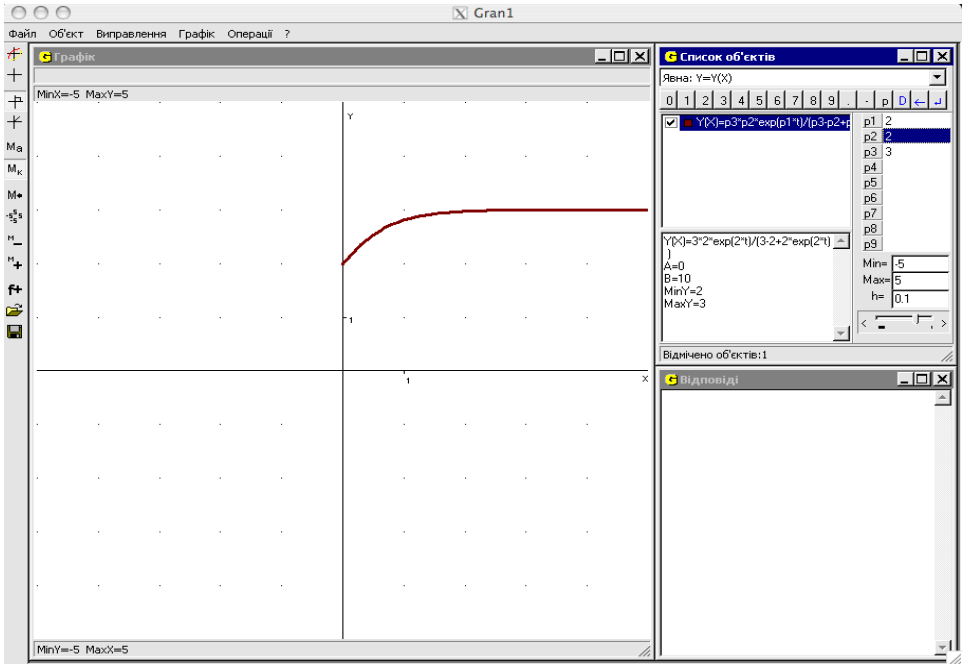

Рис. 4.37. Чисельність популяції зростає до демографічної ніші.

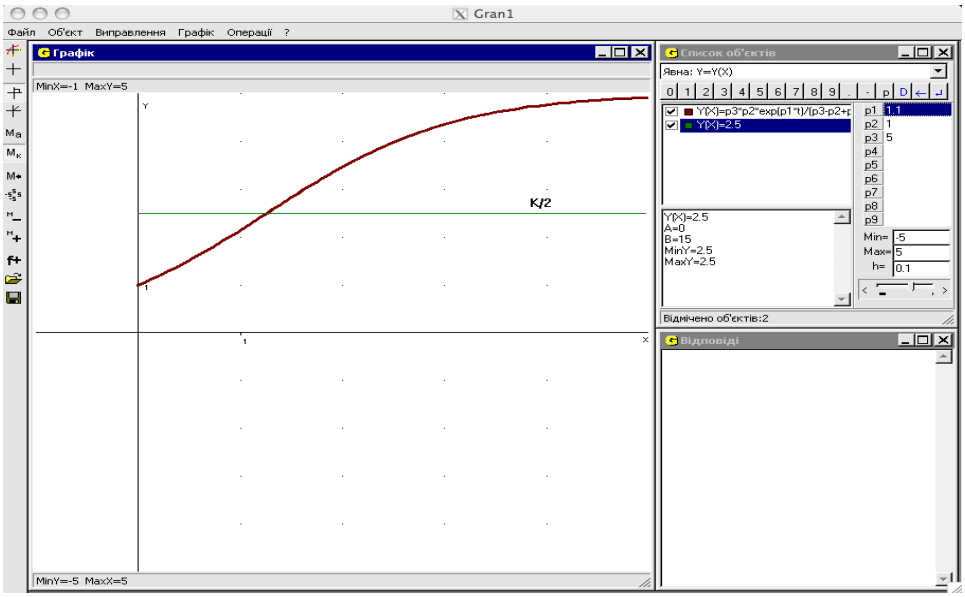

Рис. 4.38. Після інтенсивного зростання швидкість зростання чисельності популяції прямує до нуля.

Існують і більш складні моделі популяції, наприклад модель популяції з найменшою критичною чисельністю, але розглядати їх потрібно на спеціалізованих факультетах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. А. с. № 7937. Програмний комплекс GRAN : комп’ютерна програма / М. І. Жалдак, О. В. Вітюк, Ю. В. Горошко. – 11.07.2003.
2. Амамия М. Архитектура ЭВМ и искусственный интеллект / М. Амамия, Ю. Танка. – М. : Мир, 1993. – 400 с.
3. Андриевский Б. Р. Элементы математического моделирования в программных средах MATLAB 5 и Scilab / Б. Р. Андриевский, А. Л. Фрадков. – СПб. : Наука, 2001. – 286 с.
4. Архіпова Т. Л. Активізація навчально-пізнавальної діяльності учнів 7 -9 класів у процесі вивчення геометрії з використанням комп'ютера : дис. ... канд. педаг. наук : 13.00.02 / Т. Л. Архіпова ; НПУ ім. М. П. Драгоманова. – Київ, 2002. – 236 с.
5. Бабанский Ю. К. Методы обучения в современной общеобразовательной школе / Ю. К. Бабанский. – М. : Просвещение, 1985. – 208 с.
6. Бауэр Ф. Л. Информатика. Ч.1 / Ф. Л. Бауэр, Г. Гооз ; перевод с нем. М. К. Валиева и др., под ред. А. П. Ершова. – 2-е изд., перераб. и расширенное. – М. : Мир, 1990. – 324 с. : ил.
7. Бауэр Ф. Л. Інформатика. Ч. 2. / Ф. Л. Бауэр, Г. Гооз ; Перевод с нем. М. К. Валиева и др.; Под ред. А. П. Ершова. – 2-е изд., перераб. и расширенное. – М. : Мир, 1990. – 742 с.
8. Белошапка В. К. О языках, моделях и информатике / В. К. Белошапка // Информатика и образование. – 1987. – № 6. – С. 12–16.
9. Бешенков С. А. Гуманитарная информатика: от технологий и моделей к информационним принципам / С. А. Бешенков, Е. А. Ракитина, М. И. Шутикова. – М. : Лаборат. базовых знаний, 2004. – 236 с.
10. Бешенков С. А. Информатика и информационные технологии / С. А. Бешенков, А. Г. Гейн, С. Г. Григорьев. – Екатеринбург: Уральский рабочий, 1995. – 134 с.
11. Бешенков С. А. Информатика. Систематический курс. /

С. А. Бешенков, Е. А. Ракитина – М. : Бином, 2006. – 432 с.

12. Бешенков С. А. Моделирование и формализация. / С. А. Бешенков, Е. А. Ракитина. – М. : Лаборат. базовых знаний, 2002. – 336 с.
13. Бешенков С. А. Несколько замечаний о содержании школьного курса информатики / С. А. Бешенков, И. И. Трубина, В. В. Мозолин // Вестник Моск. гос. пед. ун-та. Серия «Информатика и информатизация образования». – М., 2004. – № 1(2).
14. Бешенков С. А. Развитие содержания обучения информатике в школе на основе понятий и методов формализации : дис. ... доктора пед. наук : 13.00.02 «Теория и методика обучения информатике» / С. А. Бешенков. – М., 1994. – 250 с.
15. Биков В. Ю. Моделі організаційних систем відкритої освіти: Монографія / В. Ю. Биков. – К.: Атіка, 2008. – 684 с: іл.
16. Болонський процес і кредитно-модульна система організації навчального процесу : (метод. рек. для викладачів та студ.) / [В. І. Євдокімов, О. М. Микитюк, Л. П. Харченко, В. В. Луценко]. – Харків : ХНУРЕ, 2004. – 40 с.
17. Болонський процес у фактах і документах. Сорбонна – Болонья – Саламанка – Прага – Берлін / [упоряд. : М. Ф. Степко, Я. Я. Болюбаш, В. Д. Шинкарук, В. В. Грубіянко, І. І. Бабин]. – Тернопіль : Вид-во ТДПУ ім. В. Гнатюка, 2003. – 52 с.
18. Бороненко Т. А. Теоретическая модель системы методической подготовки учителя информатики : дис. ... доктора пед. наук : 13.00.02 / Т. А. Бороненко. – СПб., 1997. – 335 с.
19. Бочкин А. И. Методика преподавания информатики / А. И. Бочкин. – Минск : Вышэйшая шк., 1998. – 431 с.
20. Братко И. Программирование на языке Пролог для искусственного интеллекта: пер. с англ. – М.: Мир, 1990. – 560 с., ил.
21. Бумагин А. Знание, сила и Интернет / А.Бумагин. // Comuterra.ru. – 2009. – № 7. – Режим доступу :

http://www.computerra.ru/interactive/404840/

22. Буч Г. Объектно-ориентированное проектирование с примерами применения: Пер. с англ. / Г. Буч. – К. : Диалектика; М. : АО «И.В.К», 1992. – 230 с.
23. Великий тлумачний словник сучасної української мови / Голов. ред. В.Т. Бусел, редактори-лексикографи: В.Т. Бусел, М.Д. Василега-Дерибас, О.В. Дмитрієв, Г. В. Латник, Г. В. Степенко. – К.: Ірпінь: ВТФ «Перун», 2005. – 1728 с.
24. Вентцель Е. С. Теория вероятностей / Е. С. Вентцель. – М, 1969. – 576 с.:ил.
25. Вербицкий А. А. Активное обучение в высшей школе : контекстный подход / А. А. Вербицкий. – М. : Высш. шк., 1991. – 204 с.
26. Винер Н. Кибернетика, или управление и связь в животном и машине / Н. Винер. – М.: Сов. радио. – 1958. – 344 с.
27. Вища освіта України і Болонський процес : навч. посіб. / М. Ф. Степко, Я. Я. Болюбаш, В. Д. Шинкарук та ін. ; за ред. В. Г. Кременя. – Тернопіль : Навч. кн. : Богдан, 2004. – 384 с.
28. Відкриття геометрії через комп’ютерні експерименти в пакеті DG : Посіб. для вчителів математики / С. А. Раков, В. П. Горох, К. О. Осенков, та ін. – Харків : Вікторія, 2002. – 136 с.
29. Войтишек А. О случайных и псевдослучайных числах. [Электронный ресурс]. – Режим доступу : http://www.computerra.ru/interactive/574366/
30. Волинський В. П. Методичні рекомендації до використання педагогічних програмних засобів у навчальному процесі / В. П. Волинський, Г. О. Козлакова. – К.: НПУ ім. М. П. Драгоманова, 2007. – 59 с.
31. Востроknутов И. Е. Оценка визуальных сред на экране монитора или почему болят глаза при работе на компьютере / И. Е. Востроknутов // Информатика и образование. – 2002. – № 1. – С. 64-67.
32. Габай Т. В. Учебная деятельность и ее средства / Т. В. Габай. – М.:

МГУ, 1988. – 256 с.

33. Габрусєв В. Ю. Зміст і методика вивчення шкільного курсу інформатики на основі вільно поширюваної операційної системи LINUX : дис. ... канд. педаг. наук : 13.00.02 «Теорія та методика навчання інформатики» / В. Ю. Габрусєв; Національний педагогічний ун-т ім. М.П. Драгоманова. – К., 2003. – 221 с.
34. Гайна Г. А. Основи проектування баз даних : навч. посіб. / Г. А. Гайна. – К.: Кондор, 2008. – 200 с.
35. Галузеві стандарти вищої освіти. Напрям підготовки 0101 Педагогічна освіта. Спеціальність 6.010100 Педагогіка і методика середньої освіти. Математика. – К. : Вид-во НПУ імені М. П. Драгоманова, 2003. – 148 с.
36. Галыгина И. В. Обучение информационному моделированию на базе модельного и объектного подходов. – Режим доступу: http://www.ict.edu.ru/vconf/index.php?a=vconf&c=getForm&r=thesisDesc&d=light&id_sec=112&id_thesis=3700
37. Гальперин П. Я. Развитие исследований по формированию умственных действий / П. Я. Гальперин // Психологическая наука в СССР. – М. : Изд-во АПН РСФСР, 1959. – Т. 1. – 599 с.
38. Ганжела С. І. Формування пізнавальної самостійності учнів основної школи в навчанні геометрії з використанням інформаційних технологій: дис. ... канд. педаг. наук : 13.00.02 / С. І. Ганжела ; НПУ ім. М.П. Драгоманова. – К., 2010. – 255 с.
39. Глушков В. М. Кибернетика. Вопросы теории и практики / В. М. Глушков. – М. : Наука, 1986. – 488 с.
40. Глушков В. М. Основы безбумажной информатики / В. М. Глушков. – 2-е изд., испр. – М. : Наука, 1987. – 552 с.
41. Гмурман В. Е. Введение в теорию вероятностей и математическую статистику / В. Е. Гмурман. – М.: Высшая школа, – 1966. – 378 с.
42. Гнєденко Б. В. Курс теорії ймовірностей / Б. В. Гнєденко. – К. : Рад.

шк., 1950. – 360 с.

43. Головань М. С. Розвиток пізнавальної активності учнів в процесі навчання алгебри і початків аналізу на основі НІТ : дис. ... канд. педаг. наук : 13.00.02 / М. С. Головань ; УДПУ ім. М. П. Драгоманова. – Київ, 1997. – 177 с.
44. Гольдин А. Образование 2.0: взгляд педагога. – / А. Гольдин // Comuterra.ru. – 2009. – № 1. – Режим доступу : http://www.computerra.ru/readitorial/393364/
45. Гончаренко С. У. Фундаменталізація професійної освіти як дидактичний принцип / С. У. Гончаренко // Теорія і практика управління соціальними системами : філософія, психологія, педагогіка, соціологія. – 2008. – № 2. – С. 87–91.
46. Гороховцева Л. А. Формирование умений информационного моделирования в процессе решения учебных задач : дис. ... канд. пед. наук : 13.00.01 / Л. А. Гороховцева. – Оренбург, 2004. – 373 с.
47. Горошко Ю. В. Автоматизація документообігу кафедри / Ю.В. Горошко, Г. Ю. Цибко, О. В. Корнієць // Вісник Чернігівського державного педагогічного університету імені Т.Г. Шевченка. Серія: Педагогічні науки. – 2006. – Вип. 42. – С. 126-132.
48. Горошко Ю. В. Активізація пізнавальної діяльності учнів на уроках математики з використанням НІТ. Проблеми інформатизації освіти / Ю. В. Горошко, А. В. Пеньков. – К. : УДПУ, 1994.- С. 47-54.
49. Горошко Ю. В. Вивчення пакету MS Office у курсі інформатики для студентів фізико-математичного факультету / Ю. В. Горошко // Комп’ютерно-орієнтовані системи навчання: зб. наук. праць. – К. : НПУ імені М. П. Драгоманова, 2002. – Вип. 5. – С. 62-68.
50. Горошко Ю. В. Використання комп'ютерних програм для створення динамічних моделей при вивченні математики / Ю. В. Горошко, Є. Ф. Вінниченко // Науковий часопис НПУ імені М. П. Драгоманова. Серія № 2 . Комп’ютерно-орієнтовані системи навчання: зб. наук.

праць. – К., 2006. – № 4(11). – С. 56-62.

51. Горошко Ю. В. Використання ППЗ „NUMET” при вивченні елементів чисельних методів / Ю. В. Горошко, А. О. Костюченко // Вісник Чернігівського державного педагогічного університету імені Т.Г. Шевченка. Серія: Педагогічні науки. – 2003. – Вип. 19. – С. 26-30.
52. Горошко Ю. В. Изучение информационного моделирования на физико-математическом факультете педагогического вуза / Ю. В. Горошко // Информатизация образования. 2008: материалы междунар. науч. конф., (г. Минск, 22-25 октября 2008 г.) / редкол.: И. А. Новик (отв. ред.) [и др.]. – Минск, 2008. – С. 126-129.
53. Горошко Ю. В. Использование GRAN1 на Linux при помощи утилиты Wine / Ю. В. Горошко, Д. А. Покрышень // Информатизация образования - 2008: материалы междунар. науч. конф., (г. Минск, 22-25 октября 2008 г.) / редкол.: И. А. Новик (отв. ред.) [и др.]. – Минск, 2008. – С. 130-134.
54. Горошко Ю. В. Использование пакета "GRAN" при изучении математики в школе / Ю. В. Горошко, Е. Ф. Винниченко // Методология и технологии образования в XXI веке: математика, информатика, физика: материалы Междунар. науч. конф., (г.Минск, 17-18 ноября 2005 г.) / Белорус. гос. пед. ун-т им.М.Танка. – Минск, 2006. – 352 с.
55. Горошко Ю. В. Інформаційне моделювання у вивченні інформатики і математики / Ю. В. Горошко // Вісник Чернігівського державного педагогічного університету імені Т.Г.Шевченка. Серія: Педагогічні науки. – 2008. – Вип. 58. – С. 23-26.
56. Горошко Ю. В. Інформаційні технології і елементи стохастики в школі / Ю. В. Горошко, М. І. Жалдак // Сучасні інформаційні технології в навчальному процесі : зб. наук. праць. – К., 1997. – С. 13-32.
57. Горошко Ю. В. Концептуальні питання створення інформаційної

мережі вищого навчального закладу / Ю. В. Горошко, А. В. Пеньков // Комп’ютерно-орієнтовані системи навчання: зб. наук. праць. – К.: НПУ імені М. П. Драгоманова, 2000. – Вип. 2. – С. 73-76.

58. Горошко Ю. В. Курс "Комп'ютерні інформаційні технології" для магістрів з математики та фізики педагогічного університету / Ю. В. Горошко // Вісник Чернігівського державного педагогічного університету імені Т.Г.Шевченка. Серія: Педагогічні науки. – 2006. – Вип. 42. – С. 132-134.
59. Горошко Ю. В. Метод найменших квадратів та його реалізація засобами НІТ / Ю. В. Горошко // Комп’ютерно-орієнтовані системи навчання: зб. наук. праць. – К.: НПУ імені М. П. Драгоманова, – 2003. – Вип. 6. – С. 106-112.
60. Горошко Ю. В. Методика вивчення ППЗ GRAN-2D на уроках інформатики та його застосування в планіметрії / Ю. В. Горошко, Л. Грамбовська // Комп'ютер в сім'ї та школі. – 2008. – № 3. – С. 14-22.
61. Горошко Ю. В. Міжпредметні зв’язки інформатики з математикою та фізикою у навчанні майбутнього інженера / Ю. В. Горошко, Д. А. Покришень // Інформаційні технології і засоби навчання: електронне наук. фахове видання [Електронний ресурс] – 2009. – № 1(9). – Режим доступу : http://www.nbuv.gov.ua/e-journals/ITZN/em9/emg.html
62. Горошко Ю. В. НІТН математики і активізація навчально-пізнавальної діяльності учнів / Ю. В. Горошко, А. В. Пеньков, М. Я. Ігнатенко // Комп’ютерно-орієнтовані системи навчання: зб. наук. праць. – К., – 2002. –Вип. 5. — С. 11-20.
63. Горошко Ю. В. Обласні олімпіади з інформатики на Чернігівщині / Ю. В. Горошко, А. О. Костюченко // Комп'ютер в сім'ї та школі. – 2007. – № 7(63). – С. 47-50.
64. Горошко Ю. В. Про альтернативний підхід до вивчення непроцедурних мов програмування / Ю. В. Горошко, А. В. Пеньков //

Сучасні інформаційні технології в навчальному процесі : зб. наук. праць. – К., 1997. – С. 101-113.

65. Горошко Ю. В. Про вивчення програмних засобів підготовки текстів та документів / Ю. В. Горошко, А. В. Пеньков // Комп'ютерно-орієнтовані системи навчання : зб. наук. праць. – К.: НПУ імені М. П. Драгоманова, 1998. – С. 90-95.
66. Горошко Ю. В. Про посилення практичної спрямованості курсу інформатики / Ю. В. Горошко, А. В. Пеньков, Н. П. Сергієнко // Наукові записки Ніжинського державного педагогічного університету. Серія : Фізико-математичні науки. – Ніжин, 1995. – Т.XV. – С. 43-46.
67. Горошко Ю. В. Програма Gran1 для Windows / Ю. В. Горошко // Комп'ютерно-орієнтовані системи навчання: зб. наук. праць. – К.: НПУ імені М. П. Драгоманова, 2001. – Вип. 3. — С. 83-89.
68. Горошко Ю. В. Програма GRAN1 для вивчення математики в школі і вузі : методичні рекомендації / Ю. В. Горошко, М. І. Жалдак. – К. : вид-во КДПІ, 1992. – 48 с.
69. Горошко Ю. В. Програма GRAN1 для супроводу вивчення математики в школі і в ВУЗі : метод. рек. / Ю. В. Горошко // Нові інформаційні технології навчання : міжнар. наук.-пед. електрон. журн. – К., 1992. – Т.1, вип. 1-2.
70. Горошко Ю. В. Розв'язування задач з математичної статистики з використанням програми Gran1 / Ю. В. Горошко // Комп'ютерно-орієнтовані системи навчання: зб. наук. праць. – К., 2003. – Вип. 7. – С. 105-114.
71. Горошко Ю. В. Розв'язування задач із параметрами за допомогою програми GRAN1 / Ю. В. Горошко, Є. Ф. Вінниченко // Математика в шк. – 2006. – № 4. – С. 25-28.
72. Горошко Ю. В. Розв'язування математичних задач практичного змісту за допомогою комп'ютера. / Ю. В. Горошко, А. В. Пеньков // Сучасна інформаційна технологія в навчальному процесі : зб. наук. праць. – К.,

1991. – С. 41-51.

73. Горошко Ю. В. Теорія і методика розробки педагогічних програмних засобів / Ю. В. Горошко, А. О. Костюченко. – Чернігів: Виготовлення Єрмоленко О.М., 2011. – 144 с.
74. Грабарь М. И. Применение математической статистики в педагогических исследованиях. Непараметрические методы / М. И. Грабарь, К. А. Краснянская. – М.: Педагогика, 1977. – 136 с.
75. Грамбовська Л. В. Особистісно орієнтоване навчання геометрії в основній школі: дис. ... канд. педаг. наук : 13.00.02 / Л. В. Грамбовська; НПУ ім. М.П. Драгоманова. – Київ, 2009. – 313 с.
76. Гришко Л. В. Концептуальні підходи до навчання основ програмування у вищій школі / Л. В. Гришко // Науковий часопис Національного педагогічного університету імені М. П. Драгоманова. Серія 2 : Комп'ютерно-орієнтовані системи навчання. – К.: НПУ імені М. П. Драгоманова, 2004. – № 1(8). – С. 134–147.
77. Губарь Ю. В. Введение в математическое моделирование / Ю. В. Губарь. – Режим доступу : http://www.intuit.ru/department/calculate/intromathmodel/1/2.html
78. Даль В. И. Толковый словарь живого великорусского языка / В. И. Даль. – М.: Цитадель, 1998.
79. Дейт К. Введение в системы баз данных / К. Дейт. – 8-е изд. – М.; СПб.: Вильямс, 2005. – 440 с.
80. Депман И. Я. История арифметики : пособие для учителей / И. Я. Депман. – М., 1965. – 415 с.
81. Державна національна програма «Освіта. Україна XXI століття». – К. : Райдуга, 1994. – 61 с.
82. Державний стандарт освітньої галузі "Технології" : (проект) для загальноосвіт. серед. шк. / В. Ю. Биков, М. І. Жалдак, Н. В. Морзе та ін. // Освіта України. – 2003. – № 3-4. – 10 с.
83. Дидактика средней школы : некоторые проблемы современной

дидактики / под ред. М. А. Данилова и М. Н. Скаткина. – М. : Просвещение, 1975. – 304 с.

84. Дичківська І. М. Інноваційні педагогічні технології : навч. посіб. / І. М. Дичківська. – К. : Академвидав, 2004. – 352 с.
85. Дринфельд Г.И. Интерполирование и способ наименьших квадратов / Г. И. Дринфельд. – К. : Вищ. шк., 1984. – 103 с.
86. Експертні системи. – Режим доступу : http://uk.wikipedia.org/wiki/Експертні_системи
87. Ершов А. П. Избранные труды / А. П. Ершов. – Новосибирск : Наука, 1994. – 416 с.
88. Ершов А. П. Компьютеризация школы и математическое образование / А. П. Ершов // Математика в шк. – 1989. – № 1. – С. 14-31.
89. Ершов А. П. О предмете информатики / А. П. Ершов // Вестн. АН СССР. – 1984. – № 2. – С. 113.
90. Жадібний_алгоритм. – Режим доступу : http://uk.wikipedia.org/wiki/Жадібний_алгоритм
91. Жалдак М. И. Система подготовки учителя к использованию информационной технологии в учебном процессе : дис. ... в форме науч. докл. доктора педаг. наук : 13.00.02 / М. И. Жалдак ; АПН СССР; НИИ содержания и методов обучения. – М., 1989. – 48 с.
92. Жалдак М. І. Елементи стохастики : посіб. для вчителів / М. І. Жалдак, Г. О. Михалін. – 3-тє вид. – К.: Шк. світ, 2004. – 120 с.
93. Жалдак М. І. Елементи стохастики з комп'ютерною підтримкою : посіб. для вчителів / М. І. Жалдак, Г. О. Михалін // Інформатика. – 2006.– № 29-30 (серпень). – 119 с.
94. Жалдак М. І. Збірник задач і вправ з теорії ймовірностей і математичної статистики: для студентів фіз.-мат. спеціальностей пед. ун-тів / М. І. Жалдак, Н. М. Кузьміна, Г. О. Михалін. – Полтава: Довкілля-К, 2010. – 717 с.
95. Жалдак М. І. Інформатика – 7 : експеримент. навч. посіб. для учнів 7

кл. загальноосвіт. шк. / М. І. Жалдак, Н. В. Морзе. – К. : ДіаСофт, 2000. – 207 с.

96. Жалдак М. І. Інформатика : навч. посіб. / М. І. Жалдак, Ю. С. Рамський ; за ред. М. І. Шкіля. – К. : Вища шк., 1991. – 319 с.
97. Жалдак М. І. Комп'ютер і елементи стохастики у шкільному курсі математики / М. І. Жалдак, Ю. В. Горошко // Комп'ютер у школі та сім'ї. – 1998. – №№ 3. – С. 16-20.
98. Жалдак М. І. Комп'ютер і елементи стохастики у шкільному курсі математики / М. І. Жалдак, Ю. В. Горошко // Комп'ютер у школі та сім'ї. – 1998. – №№ 4. – С. 22-27.
99. Жалдак М. І. Комп'ютер на уроках геометрії : посіб. для вчителів / М. І. Жалдак, О. В. Вітюк. – К. : ДІНІТ, 2003. – 168 с.
100. Жалдак М. І. Комп'ютер на уроках математики : посіб. для вчителів / М. І. Жалдак. – К. : Техніка, 1997. – 303 с.: іл.
101. Жалдак М. І. Комп'ютер на уроках фізики: Посібник для вчителів / М. І. Жалдак, Ю. К. Набочук, І. Л. Семещук – Рівне: «ТЕТІС», 2005. – 228 с.
102. Жалдак М. І. Математика (тригонометрія, геометрія, елементи стохастики) з комп'ютерною підтримкою : навч. посіб. / М. І. Жалдак, А. В. Грохольська, О. Б. Жильцов. – К. : Міжрегіон. акад. управління персоналом, 2004. – 456 с.
103. Жалдак М. І. Математика з комп'ютером : посіб. для вчителів / М. І. Жалдак, Ю. В. Горошко, Є. Ф. Вінниченко. – К.: РУНЦ „ДИНИТ", 2004. – 251 с.
104. Жалдак М. І. Математика з комп'ютером : посіб. для вчителів. – 2-ге вид. / М. І. Жалдак, Ю. В. Горошко, Є. Ф. Вінниченко. – К.: вид-во НПУ імені М. П. Драгоманова, 2009. – 274 с.
105. Жалдак М. І. Методика вивчення основ інформатики та обчислювальної техніки в педагогічному вузі : навч. посіб. / М. І. Жалдак. – К., 1986. – 74 с.

106. Жалдак М. І. Основи теорії і методів оптимізації : навч. посіб. для студ. мат. спец. вищ. навч. закл. / М. І. Жалдак, Ю. В. Триус – Черкаси : Брама-Україна, 2005. – 607 с.
107. Жалдак М. І. Педагогічний потенціал комп'ютерно-орієнтованих систем навчання математики / М. І. Жалдак // Комп'ютерно-орієнтовані системи навчання : зб. наук. праць. – К.: НПУ імені М. П. Драгоманова, 2003. – Вип. 7. – С. 3–16.
108. Жалдак М. І. Програма шкільного курсу «Інформатика» для базової школи (7-9 кл.) / М. І. Жалдак, Н. В. Морзе, Г. Г. Науменко // Інформатика. – 2003. – № 3. – С. 3-26.
109. Жалдак М. І. Теорія ймовірностей і математична статистика з елементами інформаційної технології: навч. посіб. / М. І. Жалдак, Н. М. Кузьміна, С. Ю. Трохимчук. – К.: Вища шк. – 1995. – 351 с.:іл..
110. Жалдак М. І. Теорія ймовірностей і математична статистика: підручник для студ. фіз.-мат. фак-тів пед. ун-тів. – Вид. 2-ге, перероб. і доп./ М. І. Жалдак, Н. М. Кузьміна, Г. О. Михалін. – Полтава: Довкілля-К, 2009. – 500 с.
111. Жалдак М. І. Формування інформаційної культури вчителя [Електронний ресурс] / М. І. Жалдак, О. А. Хомік. – Режим доступу : http://www.icfcst.kiev.ua/SYMPOSIUM/Proceedings/Galdak.doc
112. Жалдак М. І. Чисельні методи математики : посіб. для самоосвіти вчителів / М. І. Жалдак, Ю. С. Рамський. – К. : Рад. шк., 1984. – 206 с.
113. Жильцов О. Б. Розв'язування деяких задач математичного програмування з використанням комп'ютера / О. Б. Жильцов // Комп'ютерно-орієнтовані системи навчання: зб. наук. праць. – К., 2000. – Вип. 2. — 326 с.
114. Жук Ю. А. Решение исследовательских задач по физике с использованием новых информационных технологий: дис. ... канд. педаг. наук : 13.00.02 / Ю. А. Жук ; УГПУ им. М.П. Драгоманова. – Киев, 1995. – 217с.

115. Зайцева Т. В. Комп’ютерні технології на уроках алгебри та початків аналізу / Т. В. Зайцева // Комп’ютер у школі та сім’ї. – 1999. – № 4. – С.34-37.
116. Закон України «Про вищу освіту» / Верховна Рада України, Ін-т законодавства. – К., 2002. – 96 с.
117. Иванова Н. В. Использование принципа аналогии при обучении программированию / Н. В. Иванова, Лю Минь // Информатизация образования. 2008. Интеграция информационных и педагогических технологий: материалы междунар. науч. конф., (г. Минск, 22-25 октября 2008 г.) / редкол.: И. А. Новик (отв. ред.) [и др.]. – Минск, 2008. – С. 223-226.
118. Информатика и информационные технологии. Тема 9. Информационное моделирование. Разработка Института дистантного образования Российского университета дружбы народов, 2006. Режим доступу http://www.ido.rudn.ru/nfpk/inf/inf9.html.
119. Информатика. 7-9 кл. : базовый курс : практикум-задачник по моделированию / под ред. Н. В. Макаровой. – СПб. : Питер, 2007. – 176 с.: ил.
120. Информатика. 7-9 кл. : базовый курс : теория / под ред. Н. В. Макаровой. – СПб. : Питер, 2007. – 366 с.
121. История и перспективы школьной информатики в России / А. А. Кузнецов // Вестник Московского городского педагогического университета. Серия : Информатика и информатизация образования. – М., 2004. – № 1. – С. 1-7.
122. Ігнатенко М. Я. Активізація навчально-пізнавальної діяльності учнів старших класів при вивченні математики / М. Я. Ігнатенко. – К. : Тираж, 1997. – 300 с.
123. Ігнатенко М. Я. Активізація навчально-пізнавальної діяльності учнів старших класів при вивченні математики : дис. ... доктора пед. наук: 13.00.02 / М. Я. Ігнатенко – К., 1997. – 355 с.

124. Інформатика. 10-11 кл. : програми для загальноосвіт. навч. закл. / М. І. Жалдак, Н. В. Морзе, О. І. Мостіпан, Г. Г. Науменко. – Кам'янець-Подільський : Абетка–НОВА, 2002. – 80 с.
125. Інформатика: 10 кл. : дворівн. навч. посіб. для загальноосвіт. навч. закл./ [В. А. Ребрина, Й. Я. Ривкінд, Л. А. Чернікова, В. В. Шакотько] ; за заг. ред. М.З. Згуровського. – К. : Генеза, 2008. – 344 с.: іл.
126. Інформатика: підруч. для 11 кл. загальноосвіт. навч. закл.: академ. рівень, профіл. рівень. / [Й.Я. Ривкінд, Т.І. Лисенко, Л.А. Чернікова, В.В. Шакотько]; за заг. ред. М.З. Згуровького. – К.: Генеза, 2011. – 304 с.: іл.
127. Інформаційні технології в аналітичній геометрії / С. А. Раков, В. П. Горох, Т. О. Олійник та ін. – Харків : РЦНИТ, 2000. – 192 с.
128. Калоша В. К. Математическая обработка результатов эксперимента / В. К. Калоша, С. И. Лобко, Т. С. Чикова. – Минск: Вышэйша шк. – 1982. – 103 с.
129. Клочко В. І. Використання технології «клієнт-сервер» для побудови навчальних систем / В. І. Клочко, І. В. Жовтяк // Вісник Вінницького політехнічного інституту. – 2001. – № 3. – С. 117-122.
130. Клочко В. І. Застосування новітніх інформаційних технологій при вивченні вищої математики у технічному вузі : навч.-метод. посіб. / В. І. Клочко. – Вінниця : ВДТУ, 1997. – 300 с.
131. Клочко В. І. Нові інформаційні технології навчання математики в технічній вищій школі : дис. ... доктора пед. наук : 13.00.02 / В. І. Клочко; Вінницький держ. технічний ун-т. – Вінниця, 1998. – 396 с.
132. Кобильник Т. П. Компетентнісний підхід при вивченні «математичної інформатики» у педагогічному університеті [Електронний ресурс] / Т. П. Кобильник // Інформаційні технології і засоби навчання. – 2007. – Вип. 2. – Режим доступу : http://www.nbuv.gov.ua/e-journals/ITZN/em2/emg.html

133. Кобильник Т. П. Системи комп’ютерної математики: Maple, Mathematica, Maxima / Т. П. Кобильник. – Дрогобич : Ред.-вид. відділ ДДПУ імені Івана Франка, 2008. – 316 с.
134. Кобильник Т. П. Фундаментальність інформатичної освіти / Т. П. Кобильник // Науковий часопис Національного педагогічного університету імені М. П. Драгоманова. Серія № 2 : Комп’ютерно-орієнтовані системи навчання : зб. наук. праць. – К., 2007 – № 5 (12). – С. 78–81.
135. Колде Я. К. Практикум по теории вероятностей и математической статистике : учеб. пособие для техникумов / Я. К. Колде. – М.: Высш. шк. – 1991. – 157 с. : ил.
136. Компьютер обретает разум : пер с англ./ ред. и предисл. В.Л. Стефанюка. – М.: Мир. – 1990. – 240 с.: ил.
137. Компьютерные модели популяционной динамики. – Режим доступу : elar.usu.ru/bitstream/1234.56789/1353/4/1324624_lectures.ppt
138. Кондратенко С. В. Maxima/MathML – новый интерфейс к системе компьютерной алгебры Maxima / С. В. Кондратенко, Н. В. Моисеенко, С. А. Семериков, И. А. Теплицкий // Проблеми підготовки та перепідготовки фахівців у сфері інформаційних технологій : матеріали IV Міжнар. наук.-техн. конф. «Комп’ютерні технології в будівництві» (г. Київ–Севастополь, 18-21 вересня 2006 р.) – Кривий Ріг, 2006. – С. 33–34.
139. Концепція Державної програми розвитку освіти на 2006-2010 роки // Збірник нормативно-правових документів з питань вищої освіти. – К., 2007. – 87 с.
140. Котов В. М. О стандартах среднего образования по информатике / В. М. Котов, О. И. Мельников // Информатизация образования. 2008. Интеграция информационных и педагогических технологий: материалы междунар. науч. конф. (г. Минск, 22-25 октября 2008 г.) / редкол.: И.А.Новик (отв. ред.) [и др.]. – Минск, 2008. – С. 291-294.

141. Крамаренко Т. Г. Формування особистісних якостей школяра у процесі комп'ютерно-орієнтованого навчання математики: дис. ... канд. педаг. наук : 13.00.02 / Т. Г. Крамаренко ; НПУ ім. М.П. Драгоманова. – Київ, 2008. – 270 с.
142. Кремень В. Г. Вища освіта і наука – пріоритетні сфери розвитку суспільства у XXI столітті / В. Г. Кремень // Вища шк. – 2002. – № 4–5. – С. 3–33.
143. Криволуцкая Н. П. Дистанционный курс “Теоретические основы компьютерного моделирования. – Режим доступу: http://schools.keldysh.ru/courses/distant-5/k-2.doc
144. Кузнецов А. А. Современный курс информатики: от элементов к системе / А. А. Кузнецов, С. А. Бешенков, Е. А. Ракитина // Информатика и образование. – 2004. – № 1. – С. 1-7.
145. Кузнецов С. Д. Базы данных : вводный курс. – Режим доступа : http://www.citforum.ru/database/advanced_intro/7.shtml
146. Кузьменко М. В. Развитие межпредметных связей курса математики в средних профессиональных учебных заведениях (для специальностей группы “Информатика и вычислительная техника”) : автореф. дис. на соискание уч. степени канд. пед. наук / М. В. Кузьменко. – М., 2004. – 20 с.
147. Кузьміна Н. М. Компетентнісний підхід до навчання інформаційних систем і технологій майбутніх вчителів економіки [Електронний ресурс] / Н. М. Кузьміна, О. В. Струтинська // Інформаційні технології в освіті: Збірник наукових праць. – Херсон: Видавництво ХДУ, 2011. – Вип. 9. – С. 57-63. – Режим доступу: http://ite.ksu.ks.ua/webfm_send/204
148. Кук Д. Компьютерная математика / Д. Кук, Г. Бейз – М. : Наука, 1990. – 384 с.
149. Кухтенко А. И. Кибернетика и фундаментальные науки / А. И. Кухтенко. – К. : Наук. думка, 1987. – 140 с.

150. Кыверялг А. А. Методы исследования в профессиональной педагогике. / А. А. Кыверялг. – Таллин: Валгус, 1980. – 334 с.
151. Лапчик М. П. Методика преподавания информатики : учеб. пособие для студ. пед. вузов / М. П. Лапчик, И. Г. Семакин, Е. К. Хеннер; под общей ред. М. П. Лапчика. – М. : Академия, 2001. – 624 с.
152. Лапчик М. П. Структура и методическая система подготовки кадров информатизации школы в педагогических вузах : дис. … доктора пед. наук в форме научн. докл. : 13.00.02 / М.П. Лапчик. – М., 1999. – 82 с.
153. Леонова А. А. Историческая реконструкция как моделирование исторического процесса / А. А. Леонова. – СПб. : ЛГПИ, 1990. – 27 с.
154. Леонова Н. А. До питання розробки та впровадження системи символьної математики Maxima у ВНЗ України / Н. А. Леонова, І. О. Теплицький, С. О. Семеріков // Информационные технологии в учебном процессе : сб. тр. IV науч.-метод. семинара. – Одесса, 2003. – С. 183–185.
155. Линник Ю. В. Метод наименьших квадратов и основы математико-статистической обработки наблюдений / Ю. В. Линник. – Л. : Физматгиз. – 1962. – 352 с.
156. Линькова В. П. Развитие методической системы обучения информатике на основе информационного и информационно-логического моделирования : автореф. дис. на соискание ученой степени доктора пед. наук : спец. 13.00.02 «Теория и методика обучения информатике» / В. П. Линькова. – М., 2000. – 37 с.
157. Липский И. А. Общие теоретические основы современных дидактических технологий : материалы межвузовской науч.-практ. конф., 18 июня 2003 г. – М., 2004. – С. 107-119. – Режим доступа : http://hotel-olimp.ru/page/Statii/Technol-2.html
158. Литвинова С. Г. Інформаційно-комунікаційні компетентності вчителів загальноосвітніх навчальних закладів / С. Г. Литвинова // Комп’ютер у школі та сім’ї. – 2011. – № 4, 5.

159. Ліннік О. П. Об’єктно-орієнтоване моделювання у підготовці майбутніх учителів фізики / О. П. Ліннік, Н. В. Моісеєнко, В. М. Євтєєв та ін. // Збірник наукових праць Кам’янець-Подільського державного університету. Серія педагогічна: Проблеми дидактики фізики та шкільного підручника фізики в світлі сучасної освітньої парадигми. – Кам’янець-Подільський, 2006. – Вип. 12.– С. 127-130.
160. Любченко К. М. Елементи математичної логіки з комп’ютерною підтримкою : посіб. для вчителів / К. М. Любченко, Ю. В. Триус. – Черкаси : Черкас. нац. ун-т ім. Б. Хмельницького, 2004. – 87 с.
161. Машбиц Е. И. Психологические основы управления педагогической деятельностью: Метод. пособие / Е. И. Машбиц. – К.: Вища шк., 1987. – 223 с.
162. Машбиц Е. И. Психолого-педагогические проблемы компьютеризации обучения / Е. И. Машбиц. – М. : Педагогика, 1988. – 192 с.
163. Машбиць Ю.І. Основи нових інформаційних технологій навчання: Посібник для вчителів / Ю. І. Машбиць, О. О. Гокунь, М. І. Жалдак, О. Ю. Комісаров, Н. В. Морзе. – К., 1997. – 260 с.
164. Методологические проблемы развития педагогической науки. – М. : Педагогика, 1985. – 240 с.
165. Минькович Т. В. Обучение компьютерным технологиям с позиции решения сквозных проблем информатики / Т. В. Минькович // Материалы XI Международной конференции-выставки «Информационные технологии в образовании» («ИТО-2001»). – М., 2001.
166. Михалін Г. О. Професійна підготовка вчителя математики у процесі навчання математичного аналізу / Г. О. Михалін. – К. : ДІНІТ, 2003. – 320 с.
167. Михалін Г. О. Формування основ професійної культури вчителя математики у процесі навчання математичного аналізу : дис. ... доктора педаг. наук : 13.00.04 / Г. О. Михалін; Національний

педагогічний ун-т ім. М. П. Драгоманова. – К., 2004. – 413 с.

168. Модель данных. – Режим доступу : http://ru.wikipedia.org/wiki/Модель_данных
169. Мозолин В. В. Информационная подготовка в профессиональном вузе / В. В. Мозолин. – М. : Образование и Информатика, 2005. – 128 с.
170. Монахов В. М. Технологические основы проектирования и конструирования учебного процесса / В. М. Монахов. – Волгоград : Перемена, 1995. – 152 с.
171. Морзе Н. В. Лабораторний практикум з методики навчання інформатики / Н. В. Морзе, Т. В. Дубова. – К. : Курс, 2003. – 242 с.
172. Морзе Н. В. Методика навчання інформатики. Ч. 1. Загальна методика навчання інформатики / Н. В. Морзе. – К. : Навч. кн., 2003. – 254 с.
173. Морзе Н. В. Методика навчання інформатики. Ч. 2. Методика навчання інформаційних технологій / Н. В. Морзе. – К. : Навч. кн., 2003. – 288 с.
174. Морзе Н. В. Методика навчання інформатики. Ч. 3. Методика навчання основним послугам глобальної мережі Інтернет / Н. В. Морзе. – К. : Навч. кн., 2003. – 196 с.
175. Морзе Н. В. Методика навчання інформатики. Ч. 4. Методика навчання основам алгоритмізації і програмування / Н. В. Морзе. – К. : Навч. кн., 2003. – 250 с.
176. Морзе Н. В. Основи методичної підготовки вчителя інформатики : монографія / Н. В. Морзе. – К. : Курс, 2003. – 372 с.
177. Морзе Н. В. Система методичної підготовки майбутніх вчителів інформатики в педагогічних університетах : дис. ... доктора педаг. наук : 13.00.02 / Н. В. Морзе ; Національний педагогічний ун-т ім. М. П. Драгоманова. – К., 2003. – 605 с.
178. Навчальна програма з математики для загальноосвітніх навчальних закладів, 10-11 класи (академічний рівень) // Математика в школі. – 2011. – №6. – С. 11-20.

179. Навчальна програма з математики для загальноосвітніх навчальних закладів. 10-11 класи (профільний рівень) // Математика в школі. – 2011. – №№7-8. – С. 3-16.
180. Наумов А. А. Образование 2.0 стучится в дверь... откроем? / А. А. Наумов // Компьютерра, – 2008. – № 44 (25 ноября). – Режим доступу : http://offline.computerra.ru/2008/760/388331/
181. Національна доктрина розвитку освіти України у XXI столітті // Освіта України. – 2002. – № 6. – С. 3–9.
182. Нейронные сети. – Режим доступу: www.inf-man.ru/download/sysan/lecture02sa.doc
183. Неуймин Я. Г. Модели в науке и технике: история, теория, практика / Я. Г. Неуймин. – Л. : Наука, 1984. – 188 с.
184. Новик И. А. Методы решения стандартных и нестандартных задач, содержащих знак модуля (с использованием программного обеспечения) / И. А. Новик, Н. В. Бровка, О. В. Хайновская. – Мн.: Ольден, 2006. – 106 с.
185. Новик И. Б. О моделировании сложных систем / И. Б. Новик. – М. : Мысль, 1965. – 118 с.
186. Новые педагогические и информационные технологии в системе образования / Е. С. Полат, М. Ю. Бухаркина, М. В. Моисеева, А. Е. Петров. – М. : Академия, 2005. – 272 с.
187. Об утверждении перечня приоритетных направлений фундаментальных и прикладных научных исследований Республики Беларусь на 2006-2010 годы : постановление Совета Министров Республики Беларусь, 17 мая 2005 г. № 512.
188. Окулов С. М. Когнитивная информатика : монография / С. М. Окулов. – Киров : Изд-во ВятГУ, 2003. – 224 с.
189. Онлайн Энциклопедия Кругосвет. Семантика. – Режим доступу : http://www.krugosvet.ru/enc/gumanitarnye_nauki/lingvistika/SEMANTIKA.html
190. Операционная система BeOS (БиОС). Особенности версии 5.0. –

Режим доступу : http://avs-info.ru/os/os_beos/beos.htmlhttp://avs-info.ru/os/os_beos/beos.html

191. Основи нових інформаційних технологій навчання : посіб. для вчителів / О. О. Гокунь, М. І. Жалдак, Ю. І. Машбиць. [та ін.] – К. : Віпол, 1997. – 262 с.
192. Осуга С. Обработка знаний: пер. с япон. / С. Осуга. – М. : Мир, – 1989. – 293 с., ил.
193. Открытые системы : концепция и реальность // Открытые системы. – 1993. – № 4. – С. 53-58.
194. Панфилова А. П. Игровое моделирование в деятельности педагога / А. П. Панфилова; под общ. ред. В. А. Сластенина, И. А. Колесниковой. – М. : ACADEMIA, 2006. – 363 с.
195. Параскевич С. П. Астроїда – це цікаво! : навч.-метод. посібник для самоосвіти / С. П. Параскевич. – К.: Вид-во НПУ імені М.П. Драгоманова, 2011. – 87 с.: іл.
196. Параскевич С. П. Дивосплетіння ліній: Посібник для вчителів математики та інформатики / За редакцією академіка НАПН України Жалдака М.І. / С. П. Параскевич. – К.: НПУ імені М.П.Драгоманова, 2011. – 272 с.
197. Параскевич С. П. Сім уроків астроарту з комп'ютерною підтримкою / С. П. Параскевич. – К.: Вид-во НПУ імені М.П. Драгоманова, 2011. – 120 с.: іл.
198. Пеньков А. В. Использование новой информационной технологии при преподавании математики в старших классах средней школы : дис … канд. пед. наук : 13.00.02 / А. В. Пеньков. – К., 1992. – 171 с.
199. Пермякова О. С. Застосування нейронних мереж у задачах прогнозування / О. С. Пермякова, С. О. Семеріков // Молодий науковець XXI століття : матеріали Міжнар. наук.-практ. конф. (м. Кривий Ріг, 17–18 листопада 2008 р.). – Кривий Ріг, 2008. – С. 237–239.

200. Полищук А. П. Методы вычислений в классах языка С++: учеб. пособие / А. П. Полищук, С. А. Семериков. – Кривой Рог : Издательский отдел КГПИ, 1999. – 350 с.
201. Поліщук О. П. Методичні та організаційні проблеми навчання комп'ютерного програмування у вищих навчальних закладах / О. П. Поліщук, С. О. Семеріков // Теорія та методика навчання математики, фізики, інформатики : зб. наук. праць : в 3-х томах. – Кривий Ріг : Видавничий відділ НМетАУ, 2006. – Т.3, вип. VI : Теорія та методика навчання інформатики. – С. 8–11.
202. Поліщук О. П. Систематичне навчання моделювання в підготовці майбутнього вчителя / О. П. Поліщук, І. О. Теплицький, С. О. Семеріков // Комп'ютерне моделювання в освіті : матеріали Всеукраїнського наук.-метод. семінару (м. Кривий Ріг, 26 квітня 2006 р.). – Кривий Ріг, 2006. – С. 48-49.
203. Похлебаев С. М. Методологические и содержательные основы преемственности физики, химии, биологии при формировании фундаментальных естественно-научных понятий : автореф. дис. на соискание ученой степени доктора пед. наук : спец. 13.00.02 / С. М. Похлебаев ; Челяб. гос. пед. ун-т. – Челябинск, 2007. – 48 с.
204. Про затвердження Державної програми "Інформаційні та комунікаційні технології в освіті і науці" на 2006-2010 роки : постанова Кабінету Міністрів України // Офіційний вісник України. – 2005. – № 49. – С. 40.
205. Про Національну доктрину розвитку освіти : Указ Президента України // Законодавчі акти України з питань освіти. – К.: Парламент. вид-во, 2004.
206. Проблемы компьютерного обучения : [сб. ст.] / под общ. ред. В. Г. Разумовского. – М. : Знание, 1986. – 61 с. – Серия 2.
207. Програма для освітніх навчальних закладів "Основи інформатики та обчислювальної техніки" / М. І. Жалдак, Н. В. Морзе, Г. Г. Науменко,

О. І. Мостіпан. – К. : Шкільний світ, 2001. – 63 с.

208. Пролог (мова програмування). – Режим доступу : http://uk.wikipedia.org/wiki/Пролог_(мова_програмування)
209. Ракитина Е. А. Теоретические основы построения концепции непрерывного курса информатики / Е. А. Ракитина. – М. : Информатика и образование, 2002. – 88 с.
210. Раков С. А. Дослідницький підхід у математичній освіті, пакети динамічної геометрії та динамічні опорні конспекти / С. А. Раков // Комп'ютер у школі і сім'ї. – 2005. – № 5. – С. 17–21.
211. Раков С. А. Использование пакета Derive в курсе математики : учеб. пособие / С. А. Раков, Т. А. Олейник, Е. В. Скляр. – Харьков : РЦНИТ, 1996. – 160 с.
212. Раков С. А. Компьютерные эксперименты в геометрии / С. А. Раков, В.П. Горох. – Х. : РЦНИТ. – 1996. – 176 с.
213. Раков С. А. Математична освіта : компетентнісний підхід з використанням ІКТ : монографія / С. А. Раков. – Х. : Факт, 2005. – 360 с.
214. Раков С. А. Формування математичних компетентностей учителя математики на основі дослідницького підходу у навчанні з використанням інформаційних технологій : дис. ... доктора педаг. наук : 13.00.02 / С. А. Раков ; Харківський національний педагогічний ун-т ім. Г.С. Сковороди. – Х., 2005. – 516 с.
215. Рамський Ю. С. Логічні основи інформатики : навч. посіб. для студ. фіз.-мат. спец. вищих пед. навч. закл. / Ю. С. Рамський. – К. : вид-во НПУ ім. М.П. Драгоманова, 2003. – 286 с.
216. Рамський Ю. С. Методичні основи вивчення експертних систем у школі / Ю. С. Рамський, Н. Р. Балик. – К. : Логос, 1997. – 114 с.
217. Рамський Ю. С. Основи програмування (мовою Паскаль) : короткий курс лекцій : лаборатор. практикум : [навч. посіб. для студ.] / Ю. С .Рамський, Г. Ю. Цибко. – К. : НПУ ім. М. П.Драгоманова, 2004.

– 141 с.

218. Рамський Ю. С. Проектування і опрацювання баз даних : посіб. для вчителів / Ю. С. Рамський, Г. Ю. Цибко. – К., 2003. – 136 с.
219. Рамський Ю. С. Формування інформаційної культури вчителя математики при вивченні методів обчислень у педагогічному вузі / Ю. С. Рамський // Комп'ютерно-орієнтовані системи навчання : збірник наукових праць. – К., 2000. – Вип. 2. – С. 25-47.
220. Роберт И. В. Современные информационные технологии в образовании: дидактические проблемы; перспективы использования / И. В. Роберт. – М. : Школа-Пресс, 1994. – 204 с.
221. Самарский А. А. Математическое моделирование / А. А. Самарский, А. П. Михайлов. – М.: Наука, 1997. – 316 с.
222. Сейдаметова З. С. Методическая система уровневой подготовки будущих инженеров-программистов по специальности «Информатика» : дис. ... доктора пед. наук : 13.00.02 / З. С. Сейдаметова ; НПУ им. М.П. Драгоманова. – К., 2007. – 559 с.
223. Семакин И. Г. Научно-методические основы построения базового курса информатики : автореф. дис. на соискание ученой степени доктора пед. наук : спец. 13.00.02 «Теория и методика обучения информатике» / И. Г. Семакин. – Омск, 2002. – 42 с.
224. Семакин И. Г. Преподавание базового курса информатики в средней школе : метод. пособие / И. Г. Семакин, Т. Ю. Шеина. – М. : Лаборатория базовых знаний, 2000. – 496 с.
225. Семантична мережа. – Режим доступу : http://uk.wikipedia.org/wiki/Семантична_мережа
226. Семантична павутина. – Режим доступу : http://uk.wikipedia.org/wiki/Семантична_павутина
227. Семёнов А. Л. Математическая информатика в школе / А. Л. Семёнов // Информатика и образование. – 1995. – № 5. – С. 54–58.
228. Семенюк Э. П. Общенаучные категории и подходы к познанию /

Э. П. Семенюк. – Львов : Вища шк., 1978. – 295 с.

229. Семеріков С. О. CLIPS : локалізована оболонка експертної системи для вітчизняної системи освіти / С. О. Семеріков, І. О. Теплицький. – Кривий Ріг, 2006. – 34 с.
230. Семеріков С. О. Maxima – система комп'ютерної математики для вітчизняної системи освіти / С. О. Семеріков, І. О. Теплицький, С. В. Шокалюк // Науковий часопис Національного педагогічного університету імені М. П. Драгоманова. Серія № 2 : Комп'ютерно-орієнтовані системи навчання : зб. наукових праць. – К.: НПУ ім. Драгоманова, 2008. – № 6 (13). – С. 32–39.
231. Семеріков С. О. Maxima 5.13 : довідник користувача / С. О. Семеріков; за ред. академіка АПН України М.І. Жалдака. – К. : Національний педагогічний ун-т ім. М.П. Драгоманова, 2007. – 48 с.
232. Семеріков С. О. Вільне програмне забезпечення як фактор стабілізації вузівських курсів інформатики / С. О. Семеріков, І. О. Теплицький // Інформаційні технології в освіті : матеріали Всеукраїнської наук.-практ. конф., 24–26 травня 2006 р. – Мелітополь, 2006. – С. 55–56.
233. Семеріков С. О. Еволюція та сучасний стан курсу чисельних методів у вищій школі / С. О. Семеріков // Збірник наукових праць Кам'янець-Подільського державного педагогічного університету : Серія педагогічна : Дидактики дисциплін фізико-математичної та технологічної освітніх галузей. – Кам'янець-Подільський, 2002. – Вип. 8. – С. 189-193.
234. Семеріков С. О. Застосування системи комп'ютерної алгебри Maxima для генерування математичних текстів в системі дистанційного навчання / С. О. Семеріков, І. О. Теплицький // Актуальні проблеми психології : Психологічна теорія і технологія навчання. – К., 2007. – Т. 8, вип. 3. – С. 85-95.
235. Семеріков С. О. Модель Гумбольдта як першоджерело Болонського процесу / С. О. Семеріков // Проектування освітніх середовищ як

методична проблема : матеріали Всеукраїнської наук.-практ. конф. – Херсон, 2008. – С. 70–72.

236. Семеріков С. О. Огляд інтерфейсів системи комп’ютерної математики Maxima / С. О. Семеріков, І. О. Теплицький // Модернізація освіти : пошуки, проблеми, перспективи : матеріали міжнар. наук.-практ. конф. (м. Київ; Переяслав-Хмельницький, 22–25 травня 2006 року). – К.; Переяслав-Хмельницький, 2006. – С. 178–181.
237. Семеріков С. О. Основи комп’ютерного моделювання у школі та педагогічному ВНЗ / С. О. Семеріков, І. О. Теплицький // Информационные технологии и информационная безопасность в науке, технике и образовании «Инфотех–2004» : материалы междунар. науч.-практ. конф., 20–25 сентября 2004 г. – К.; Севастополь, 2004. – С. 197-207.
238. Семеріков С. О. Розробка системи символьної математики для системи вищої освіти України / С. О. Семеріков // Формування духовної культури особистості в процесі навчання математики в школі та вищому навчальному закладі : матеріали всеукраїнської наук.-практ. конф., 22-24 травня 2003 р. – Луцьк, 2003. – С. 46-47.
239. Семеріков С. О. Теоретико-методичні основи фундаменталізації навчання інформатичних дисциплін у вищих навчальних закладах: дис. ... доктора педаг. наук: 13.00.02 / С. О. Семеріков ; – К.: НПУ ім. Драгоманова, 2009. – 522 с.
240. Семеріков С. О. Штучний інтелект в курсі інформатики педагогічного ВНЗ / С. О. Семеріков, І. О. Теплицький // Інформаційні технології в освіті, науці і техніці : матеріали IV Всеукраїнської конф. молодих наук. ІТОНТ–2004 (м. Черкаси, 28–30 квітня 2004 р.) – Черкаси, 2004. – Ч. 2. – С. 180-183.
241. Середа С. В. Разработка программного обеспечения для моделирования бот-сетей / С. В. Середа, С. А. Семериков // Молодий науковець XXI століття : матеріали Міжнародної наук.-практ. конф.

(м. Кривий Ріг, 17–18 листопада 2008 р.). – Кривий Ріг, 2008. – С. 247–249.

242. Сидоренко Е. В. Методы математической обработки в психологии / Е. В. Сидоренко. – СПб. : Речь, 2003. – 350 с.
243. Скафа Е. И. Эвристическое обучение математике: теория, методика, технология / Е. И. Скафа. – Донецк: Изд-во ДонНУ, 2004. – 439 с.
244. Скоробогатова Н. В. Наглядное моделирование профессионально-ориентированных задач в обучении математике студентов инженерных направлений технических вузов : автореф. дис. на соискание ученой степени канд. пед. наук : спец. 13.00.02 «Теория и методика обучения математике» / Н. В. Скоробогатова ; Ярославский гос. пед. ун-т им. К. Д. Ушинского – Ярославль, 2006. – 25 с.
245. Смалько О. А. Розвиток творчого мислення старшокласників на уроках математики з використанням інформаційних технологій навчання : дис. ... канд. педаг. наук : 13.00.02 / О. А. Смалько ; НПУ ім. М. П. Драгоманова. – Київ, 2003. – 252 с.
246. Смирнова-Трибульска Є. М. Дистанційне навчання з використанням системи MOODLE : навч.-метод. посіб. для студ. вищих пед. навч. закл. / Є. М. Смирнова-Трибульска ; наук. ред. д-р. пед. наук, академік АПН України, проф. М. І. Жалдак. – Херсон : Айлант, 2007. – 492 с.
247. Смирнова-Трибульска Є. М. Інформаційно-комунікаційні технології в професійній діяльності вчителя : посібник для вчителів / Є. М. Смирнова-Трибульска ; науковий редактор д.пед.н., академік АПН України, проф. М. І. Жалдак. – Херсон : Айлант, 2007. – 560 с.
248. Смирнова-Трибульская Е. Н. Основы формирования информатических компетентностей учителей в области дистанционного обучения : монография / Е. Н. Смирнова-Трибульская; научный редактор академик АПН Украины, д-р. пед. наук, проф. М. И. Жалдак. – Херсон : Айлант, 2007. – 704 с.
249. Смирнова-Трибульська Є. М. Теоретико-методичні основи

формування інформатичних компетентностей вчителів природничих дисциплін у галузі дистанційного навчання : дис . ... докт. педаг. наук : 13.00.02 / Є. М. Смирнова-Трибульська ; НПУ ім. М. П. Драгоманова. – Київ, 2008. – 676 с.

250. Соловйов В. М. Методи математичного моделювання : лабораторний практикум з курсу / В. М. Соловйов, І. О. Теплицький, С. О. Семеріков. – Кривий Ріг; Черкаси, 2003. – 104 с.
251. Соловьев В. Н. Особенности компьютерного моделирования в социально-гуманитарных науках / В. Н. Соловьев, С. А. Семериков, И. А. Теплицкий // Комп'ютерне моделювання та інформаційні технології в науці, економіці та освіті : зб. наук. праць : в 2-х т. – Кривий Ріг, 2001. – Т. 1. – С. 230–236.
252. Соловьев В. Н. Синергетический подход к компьютерному моделированию социально-экономических процессов / В. Н. Соловьев, С. А. Семериков, И. А. Теплицкий // Информационные технологии и информационная безопасность в науке, технике и образовании «Инфотех–2002» : материалы международной науч.-практ. конф., 30 сентября – 5 октября 2002 г. – К.; Севастополь, 2002. – С. 61–62.
253. Спірін О. М. Система інформаційно-технологічних компетентностей учителя інформатики / О. М. Спірін // Інформаційно-комунікаційні технології навчання: матеріали міжнар. наук.-практ. конференції. – Умань: ПП Жовтий, 2008. – С. 160-162.
254. Спірін О. М. Теоретичні та методичні основи кредитно-модульної системи навчання майбутніх учителів інформатики: Дис... д-ра педаг. наук: 13.00.04 / О.М. Спірін. – К., 2009. – 495 с.
255. Степанов А. Г. Объектно-ориентированная модель информатики как предмета обучения [Электронный ресурс] / А. Г. Степанов // Материалы конференции ИТО-2006. – М., 2006. – Режим доступу : http://ito.edu.ru/2006/Moscow/I/1/I-1-6306.html

256. Степанов А. Г. Объектно-ориентированный подход к отбору содержания обучения информатике / А. Г. Степанов. – СПб. : Политехника, 2005. – 229 с.
257. Стефанова Н. Л. Теоретические основы развития системы методической подготовки учителя математики в педагогическом вузе : автореф. дис. на соискание ученой степени доктора пед. наук : спец. 13.00.02 «Теория и методика обучения математике» / Н. Л. Стефанова. – СПб., 1996. – 26 с.
258. Структура ИКТ-компетентности учителей. Рекомендации ЮНЕСКО. – Редакция 2.0. [Електронний ресурс] – Режим доступу: http://iite.unesco.org/pics/publications/ru/files/3214694.pdf
259. Суворова Т. Н. Совершенствование методики изучения информационных технологий в школьном курсе информатики : автореф. дис. на соискание ученой степени канд. пед. наук : спец. 13.00.02 «Теория и методика обучения информатике» / Т. Н. Суворова; Вятский гос. гуманит. ун-т – М., 2007. – 22 с.
260. Суханов А. П. Информация и прогресс / А. П. Суханов; отв. ред. А. Л. Симанов. – Новосибирск : Наука : Сиб. отд-ние, 1988. – 190 с. – (Сер. «Наука и техн. прогресс»).
261. Сычкова Н. В. Формирование у будущих учителей умений исследовательской деятельности в условиях классического университета: дис. … доктора педаг. наук: 13.00.08 / Н. В. Сычкова. – М., 2003. – 442 с.
262. Талызина Н. Ф. Деятельностный подход к построению модели специалиста / Н. Ф. Талызина // Вестник высшей школы. – 1986. – № 3. – С. 10–14.
263. Талызина Н. Ф. Педагогическая психология : учеб. для студ. сред. пед. учеб. заведений / Н. Ф. Талызина. – 3-е изд., стереотип. – М. : Академия, 2001. – 288 с.
264. Талызина Н. Ф. Теоретические основы разработки модели

специалиста / Н. Ф. Талызина. – М. : Знание, 1986. – 108 с.

265. Талызина Н. Ф. Управление процессом усвоения знаний (психологические основы) / Н. Ф. Талызина. – 2-е изд., доп. и испр. – М.: изд-во МГУ, 1984. – 343 с.
266. Татур Ю. Г. Высшее образование: методология и опыт проектирования: учеб.-методич. пособие / Ю. Г. Татур. – М.: Логос: Универ. кн., 2006. – 252 с.
267. Тенденции в реформировании высшего образования, развитии стандартизации и образовательных стандартов высшей школы в странах СНГ. – М. : Исследовательский центр проблем качества подготовки специалистов, 2007. – 232 с.
268. Теорія ймовірностей і статистичні методи обробки результатів спостережень : навч. посіб. для студ. вищ. фармац. та мед. закл. III-IV рівнів акредитації/ Б. Ф. Горбуненко, Ф. Г. Дягілева, Г. В. Жиронкіна та ін. – Х.: Золоті сторінки, 2003. – 188 с.
269. Теплицкий И. А. Введение в программирование систем искусственного интеллекта на языке Лисп : лабораторный практикум / И. А. Теплицкий, С. А. Семериков. – Кривой Рог : КГПУ, 2004. – 88 с.
270. Теплицкий И. А. Создание 3D-моделей физических процессов в среде Python / И. А. Теплицкий, С. А. Семериков // Дні науки : зб. тез доп. : в 3 т. / ред. кол. В. М. Огаренко та ін. – Запоріжжя, 2005. – Т. 2. – С. 157-159.
271. Теплицький І. О. «Віртуальний фізичний лабораторний практикум» як актуальна проблема сучасної дидактики / І. О. Теплицький, С. О. Семеріков // Теорія та методика навчання математики, фізики, інформатики : зб. наук. праць : в 3-х т. – Кривий Ріг , 2004. – Т. 2, вип. 4 : Теорія та методика навчання фізики. – С. 414–421.
272. Теплицький І. О. Дослідження дидактичних можливостей мови Ліспу як засобу побудови інтелектуальних систем у шкільному курсі інформатики / І. О. Теплицький, С. О. Семеріков // Проблеми

сучасного підручника : зб. наук. праць. – К., 2004. – Вип. 5., ч. ІІ. – С. 183–191.

273. Теплицький І. О. Дослідження дидактичних можливостей мови Лісп як засобу побудови інтелектуальних систем у шкільному курсі інформатики / І. О. Теплицький, С. О. Семеріков // Інформатика та комп'ютерна підтримка навчальних дисциплін у середній і вищій школі : матеріали Всеукраїнської наук.-практ. конф. (м. Бердянськ, 23-26 червня 2004 року). – Бердянськ, 2004. – С. 112–115.
274. Теплицький І. О. Елементи комп'ютерного моделювання : навч. посіб. / І. О. Теплицький. – Кривий Ріг : КДПУ, 2010. – 264 с.: іл.
275. Теплицький І. О. З досвіду використання вільного програмного забезпечення у підготовці майбутнього вчителя / І. О. Теплицький, С. О. Семеріков // Рід. шк. – 2003. – № 5. – С. 40–41.
276. Теплицький І. О. Інформаційна безпека як нова складова інформаційної культури / І. О. Теплицький, С. О. Семеріков // Рід. шк. – 2006. – № 2. – С. 63–64.
277. Теплицький І. О. Комп'ютерне моделювання абсолютних та відносних рухів планет Сонячної системи / І. О. Теплицький, С. О. Семеріков // Збірник наукових праць Кам'янець-Подільського державного університету : Серія педагогічна : Дидактика фізики і підручники фізики (астрономії) в умовах формування європейського простору вищої освіти. – Кам'янець-Подільський, 2007. – Вип. 13. – С. 211-214.
278. Теплицький І. О. Комп'ютерне моделювання визначальних фізичних експериментів / І. О. Теплицький, С. О. Семеріков // Комп'ютерне моделювання та інформаційні технології в науці, економіці та освіті : зб. наук. праць / відповід. ред. проф. В. М. Соловйов. – Кривий Ріг, 2007. – С. 167-170.
279. Теплицький І. О. Комп'ютерне моделювання механічних рухів у середовищі електронних таблиць. Ч. 1. Механічні коливання / І. О. Теплицький, С. О. Семеріков // Фізика та астрономія в школі. –

2002. – № 5. – С. 40-46.

280. Теплицький І. О. Комп’ютерне моделювання рухів тіл в центральному полі зі змінним потенціалом / І. О. Теплицький, С. О. Семеріков // Збірник наукових праць Кам’янець-Подільського державного університету : Серія педагогічна : Проблеми дидактики фізики та шкільного підручника фізики в світлі сучасної освітньої парадигми. – Кам’янець-Подільський, 2006. – Вип. 12. – С. 313–316.
281. Теплицький І. О. Комп’ютерне моделювання рухів тіл в центральному полі зі змінним потенціалом / І. О. Теплицький, С. О. Семеріков // Теорія та методика навчання математики, фізики, інформатики : Зб. наук. праць : в 3-х т. – Кривий Ріг, 2006. – Т. 2, вип. VI : Теорія та методика навчання інформатики. – С. 69–78.
282. Теплицький І. О. Комп’ютерне моделювання рухів тіл під дією сили всесвітнього тяжіння / І. О. Теплицький, С. О. Семеріков // Інформатика та інформаційні технології в навчальних закладах. – 2008. – № 1. – С. 85–95.
283. Теплицький І. О. Комп’ютерне моделювання рухів тіл під дією сили всесвітнього тяжіння / І. О. Теплицький, С. О. Семеріков // Науковий часопис Національного педагогічного університету імені М. П. Драгоманова. – Серія № 2 : Педагогічні науки : реалії та перспективи / за ред. П. В. Дмитренка, В. Д. Сиротюка. – К, 2008. – Вип. 12. – С. 319–328.
284. Теплицький І. О. Комп’ютерне моделювання руху тіл під дією сили всесвітнього тяжіння / І. О. Теплицький, С. О. Семеріков // Збірник наукових праць Кам’янець-Подільського державного університету : Серія педагогічна : Дидактики дисциплін фізико-математичної та технологічної освітніх галузей. – Кам’янець-Подільський, 2004. – Вип. 10. – С. 166–172.
285. Теплицький І. О. Психологічні умови ефективності творчої діяльності з комп’ютерного моделювання / І. О. Теплицький, С. О. Семеріков //

Проблеми підготовки та перепідготовки фахівців у сфері інформаційних технологій : матеріали V Міжнар. наук.-техн. конф. «Комп'ютерні технології в будівництві» (м. Київ; Севастополь, 18-21 вересня 2007 р.) – Кривий Ріг, 2008. – С. 85–86.

286. Теплицький І. О. Системи керування базами даних : в 3-х ч. : навч. посіб. / І. О. Теплицький. – Кривий Ріг : КДПУ, 2001.
287. Триус Ю. В. Віртуальне середовище для дистанційного навчання в Internet / Ю. В. Триус, А. П. Мещеряков, Н. О. Коваль // Комп'ютерне моделювання та інформаційні технології в науці, економіці та освіті : зб. наук. праць. – Черкаси : Брама ІСУЕП, 2003. – С. 161–165.
288. Триус Ю. В. Інформаційні технології в математичних дослідженнях / Ю. В. Триус // Матеріали тринадцятої наукової сесії Наукового Товариства ім. Шевченка у Черкасах. – Черкаси, 2002. – С. 50–54.
289. Триус Ю. В. Комп'ютерно-орієнтовані методичні системи навчання математичних дисциплін у вищих навчальних закладах : дис. ... доктора пед. наук : 13.00.02 / Ю. В. Триус ; Черкаський нац. ун-т ім. Б. Хмельницького. – Черкаси, 2005. – 649 с.
290. Триус Ю. В. Комп'ютерно-орієнтовані методичні системи навчання математичних дисциплін у вищих навчальних закладах : автореф. дис. на здобуття наук. ступеня доктора педаг. наук : спец. 13.00.02 «Теорія та методика навчання інформатики» / Ю. В. Триус ; Національний педагогічний ун-т ім. М. П. Драгоманова. – К., 2005. – 48 с.
291. Триус Ю. В. Особливості створення методичної системи навчання основ програмування для підготовки майбутніх інженерів-програмістів / Ю. В. Триус, О. О. Богатирьов, Л. В. Гришко // Вісник Черкаського університету. Серія: Педагогічні науки. – Черкаси, 2002. – Вип. 35. – С. 133–141.
292. Указ Президента України «Про невідкладні заходи щодо забезпечення функціонування та розвитку освіти в Україні» від 04.06.05р. № 1013/2005 // Збірник нормативно-правових документів з вищої

освіти. – К., 2007. – 87 с.

293. Українська радянська енциклопедія : у 12 т. – 2-ге вид. – К., 1982. – Том 7. – 526 с.
294. Уэзерелл Ч. Этюды для программистов / Ч. Уэзерелл. – М. : Мир, 1982. – 288 с.: ил.
295. Фокин Р. Р. Метамодель обучения информатике в высшей школе : автореф. дис. на соискание ученой степени доктора пед. наук : 13.00.02 «Теория и методика обучения информатике» / Р. Р. Фокин. – СПб., 2000. – 32 с.
296. Функціональне програмування. – Режим доступу : uk.wikipedia.org/wiki/Функціональне_програмування
297. Хон Р. Л. Педагогическая психология. Принципы обучения / Р. Л. Хон. – М. : Деловая книга, 2002. – 736 с.– (Gaudeamus).
298. Хуторской А. В. Современная дидактика / А. В. Хуторской. – СПб. : Питер, 2001. – 544 с.
299. Хуторською А. В. Ключові освітні компетентності. Відкритий урок. Професійний журнал для вчителів / А. В. Хуторський. – Режим доступу http://osvita.ua/school/theory/2340
300. Цибко Г.Ю. Підвищення рівня теоретичної підготовки з інформатики на фізико-математичних факультетах педагогічних вузів : дис. ... канд. пед. наук : 13.00.02 / Г. Ю. Цибко ; НПУ ім. М.П. Драгоманова. – Київ, 1998. – 200 с.
301. Частиков А. П. Разработка экспертных систем. Среда CLIPS / А. П. Частиков, Т. А. Гаврилова, Д. Л. Белов. – СПб. : БХВ-Петербург, 2003. – 608 с. : ил.
302. Челак Е. Н. Развивающаяся информатика : метод. пособие / Е. Н. Челак, Н.К. Конопатова. – М. : Лаборатория Базовых Знаний, 2001. – 208 с.
303. Чепрасова Т. І. Підвищення практичної значущості результатів навчання інформатики в старших класах середньої школи в умовах

НІТН: дис. ... канд. педаг. наук : 13.00.02 / Т. І. Чепрасова; НПУ ім. М.П. Драгоманова. – Київ, 1998. – 217 с.

304. Шадриков В. Д. Деятельность и способности / В. Д. Шадриков. – М. : Логос, 1994. – 320 с.
305. Швець В. О. Оновлення методичної системи навчання математики / В. О. Швець // Проблеми навчання математики в університеті й школі : тези доп. наук.-метод. конф. математичного ф-ту. – Донецьк, 1994. – С. 3–6.
306. Шень А. Х. Программирование : теоремы и задачи / А. Х. Шень. – М. : МЦНМО, 1995. – 264 с.
307. Шестаков А. П. Профильное обучение информатике в старших классах средней школы на примере курса "Компьютерное математическое моделирование" : автореф. дис. на здобуття наук. ступеня канд. пед. наук : спец. 13.00.02 / А. П. Шестаков. – Омск, 1999. – 20 с.
308. Шефтель З. Г. Теорія ймовірностей : підручник / З. Г. Шефтель. – К.: Вища шк., 1994. – 192 с.:іл.
309. Широких А. А. Методическая система подготовки учителя информатики по основам искусственного интеллекта : автореф. дис. на соискание ученой степени канд. пед. наук : спец. 13.00.02 «Теория и методика обучения информатике» / А. А. Широких ; Пермский гос. пед. ун-т – Омск, 2007. – 23 с.
310. Шлеер С. Объектно-ориентированный анализ: моделирование мира в состояниях / С. Шлеер, С. Меллор. – К. : Диалектика, 1993. – 240 с.
311. Шокалюк С. В. Інформаційні технології математичного призначення у навчальних та наукових дослідженнях / С. В. Шокалюк // Наукові записки Тернопільського національного педагогічного університету імені Володимира Гнатюка. Серія : Педагогіка. – Тернопіль, 2008. – № 7. – С. 37–42.
312. Штепа Ю. П. Роль обучения решению задач по информационному

моделированию для развития ИКТ-компетентности старшеклассников / Ю. П. Штепа // Вестник Дальневосточной государственной социально-гуманитарной академии. – Серия 1 : Гуманитарные науки. – Биробиджан, 2009. – № 1(2).

313. Штофф В. А. Моделирование и философия / В. А. Штоф. – М. : Наука, 1966. – 301 с.
314. Штучна нейронна мережа : матеріал з Вікіпедії – вільної енциклопедії. [Електронний ресурс] – Режим доступу: http://uk.wikipedia.org/wiki/Штучна_нейронна_мережа
315. Якунин В. А. Педагогическая психология : учеб. пособие / В. А. Якунин. – СПб. : Полиус, 1998. – 639 с.
316. Яшанов С. М. Теоретико-методичні засади системи інформатичної підготовки майбутніх учителів трудового навчання : дис. ... докт. педаг. наук : 13.00.02 / С. М. Яшанов ; НПУ ім. М.П. Драгоманова. – К., 2010. – 529 с.
317. Alexander, P. A. College instruction and concomitant changes in students' knowledge, interest, and strategy use : A study of domain learning / Alexander, P. A., Murphy, P. K., Woods, B. S., Duhon, K. E. & Parker, D. // Contemporary Educational Psychology. – 1997. – Vol. 22. – P. 125-146.
318. Bates, A. W. Educational Multimedia in a Networked Society / Bates, A. W. // Educational Multimedia and Hypermedia. Proceedings of ED-MEDIA World Conference on Educational Multimedia and Hypermedia, 1994.
319. Be File System : [материал из Википедии – свободной энциклопедии]. – Режим доступа : http://ru.wikipedia.org/wiki/BeFS
320. Bobrow D.G. If Prolog is the answer, what is the question / D.G. Bobrow // Fifth Generation of Computer Systems (Tokyo, Japan, November 1984.) / Institute for New Generation Computer Technology (ICOT). – North-Holland, 1984. – P. 138–145.
321. Budd T.A. Multy-Paradigm Programming in LEDA / T.A. Budd. –

Addison-Wesley; Reading; Massachusets, 1995. – 320 p.

322. Campbell, S. Modeling and Simulation in Scilab/Scicos / Campbell, S., Chancelier, J.-Ph., Nikoukhah, R. – New York : Springer Science, 2006. – XI, 313 p.
323. Computing curricula 1991 // Communications of the ACM. – 1991. – Vol. 34. – №6. – P. 69-84.
324. Dottori Dino. Applied mathematics for today: senior si metric edition / Dino Dottori, George Knill, John Seumour. – McGraw; Hill Ryerson Limited, 1977. – 486 p.
325. Druin, A. KidPad : A Design Collaboration Between Children, Technologists, and Educators / Druin, A., Stewart, J. [et al.] // Proceedings of CHI'97. – Atlanta, Georgia : ACM/Addison-Wesley, 1997. – P. 463-470.
326. Engeström, Y. Learning by expanding : An activity-theoretical approach to developmental research / Engeström, Y. – Helsinki : Orienta-Konsultit, 1987.
327. Friedman L.W. Comparative programming languages: generalizing the programming function / L.W. Friedman. – Prentice Hall, 1991. – 188 p.
328. Fryer Kennth D. Holt mathetatics 5 / Fryer Kennth D., Dunkley Ronald G., Elliott H.A., Hill Norman J., MacKay R.Jock. – Holt, Rinechart and Winston of Canada, Limited, 1980. – 373 p.
329. Hanwell Alfred P. Holt mathetatics 4 / Alfred P. Hanwell, Marshall P. Bye, Thomas J. Grifiths. – Holt, Rinechart and Winston of Canada, Limited, 1980. – 470 p.
330. Horoshko Y. Geometria w szkole z programem GRAN-2D (część II) / Y. Horoshko, E.Smyrnowa-Trybulska, E.Vinnichenko, A.Vituk, M.Żaldak // Matematyka i Komputery : czasopismo Grupy Roboczej SNM „Matematyka i Komputery" № 19, 2004, Wydawnictwo PWN.
331. Horoshko Y. GRAN – komputerowe wspomaganie nauczania matematyki : XX ogólnopolska Konferencja Naukowa „Informatyka w szkole"

(Wrocław, 6-9 września, 2004) / Y. Horoshko, E.Smyrnowa-Trybulska, E.Vinnichenko, A.Vituk, M.Żaldak ; pod red. prof. Macieja M. Sysło. – Wrocław, 2004.

332. Horoshko Y. Komputerowe wspomoganie nauczania matematyki pakietem programów GRAN (część I) / Y. Horoshko, E.Smyrnowa-Trybulska, E.Vinnichenko, A.Vituk, M.Żaldak // Matematyka i Komputery : czasopismo Grupy Roboczej SNM „Matematyka i Komputery”. – 2004. – № 18.
333. Horoshko Y. Matematyka z GRAN-1W : poradnik metodyczny dla nauczycieli / Y.Horoshko, M.Zhaldak, E.Vinnychenko, E.Smyrnova-Trybulska. – Sosnowiec : Wyższa Szkoła Zarządzania i Marketingu, 2005. – 287 s.: il.
334. Horoshko Y. Stereometria z programem GRAN-3D (część III) / Y. Horoshko, E.Smyrnowa-Trybulska, E.Vinnichenko, A.Vituk, M.Żaldak //Matematyka i Komputery : czasopismo Grupy Roboczej SNM „Matematyka i Komputery” № 20, 2004, Wydawnictwo PWN.
335. Invitation to the Social Media Classroom and Collaboratory. – Режим доступу : http://socialmediaclassroom.com/
336. Mayer R.E. When is an illustration worth ten thousand words? / R.E. Mayer, J.K. Gallini // Journal of Educational Psychology. – 1990. – № 82(4). – P.715-726.
337. Perlin, K. Pad : An Alternative Approach to the Computer Interface / Perlin, K., Fox, D. // Proceedings of SIGGRAPH '93. – New York, 1993. – P. 57-64.
338. Rickel, J. Intelligent tutoring in virtual reality : A preliminary report / Rickel, J., Johnson, L. // Artificial Intelligence in Education / Ed. B. du Boulay and R. Mizoguchi. – Amsterdam : IOS Press, 1997. – P. 294-301.
339. Sharples, M. Learning as conversation : Transforming education in the mobile age / Sharples, M. // Proceedings of Conference on Seeing, Understanding, Learning in the Mobile Age, Budapest, Hungary. –

Budapest, 2005. – P. 147–152.

340. Sharples, M. Socio-Cognitive Engineering : A Methodology for the Design of Human-Centred Technology / Sharples, M., Jeffery, N. [et al.] // European Journal of Operational Research. – 2002. – Vol. 132(2). – P. 310-323.
341. Shaw, M. We Can Teach Software Better / Mary Shaw // Computing Research News. – September 1992. – 4(4) – P. 2–4, 12.
342. Shriver B.D. Software paradigms. IEEE Software, 3(1):2, January 1986.
343. Skliarenko, E. Transitional economies : trends in new higher educational systems in Easter Europe – the example of Ukraine [Electronic resource] / Skliarenko, E. – Proceedings of ICED2004. – 2004, June 21–23. – Mode of access: http://www.uottawa.ca/services/tlss/iced2004/pages/doc/ski.doc
344. Sokolnikoff I.S. Mathematics of physics and modern engineering / I.S. Sokolnikoff, R.M. Redheffer. – McGraw-Hill, Inc. – 1966.– 752 c.
345. Spinellis D.D. Programming paradigms as object classes: a structuring mechanism for multiparadigm programming. PhD thesis, University of London. – London SW7 2BZ ; United Kingdom, February 1994.
346. Spotlight: быстрый поиск чего угодно и где угодно. – Режим доступу : http://iland.com.ua/articles/spotlight-osxtiger-review/
347. Web 2.0 : [википедия]. – Режим доступа : http://ru.wikipedia.org/wiki/Веб_2.0
348. Wegner P. Concepts and paradigms of object-oriented programming. // ACM SIGPLAN OOPS Messenger. – 1990. – V.1, № 1.
349. Whitsed, N. Learning and Teaching / Whitsed, N. // Health Information & Libraries Journal. –2004. – Dec., Vol. 21. – P. 273–275.
350. WinFS : [материал из Википедии – свободной энциклопедии]. – Режим доступа : http://ru.wikipedia.org/wiki/WinFS
351. Zhaldak M. I. Mathematics with a computer: The Teachers guide. / M. I. Zhaldak, Y. V. Horoshko, E. F. Vinnichenko, G. Y. Tsybko. – K.: National Dragomanov Pedagogical University, 2012. – 250 p.