КРИВОРІЗЬКИЙ ДЕРЖАВНИЙ ПЕДАГОГІЧНИЙ УНІВЕРСИТЕТ

На правах рукопису

СЕМЕРІКОВ Сергій Олексійович

УДК 378.147+518.5

АКТИВІЗАЦІЯ ПІЗНАВАЛЬНОЇ ДІЯЛЬНОСТІ СТУДЕНТІВ ПРИ ВИВЧЕННІ ЧИСЕЛЬНИХ МЕТОДІВ У ОБ'ЄКТНО-ОРІЄНТОВАНІЙ ТЕХНОЛОГІЇ ПРОГРАМУВАННЯ

13.00.02 – теорія та методика навчання інформатики

Дисертація на здобуття наукового ступеня
кандидата педагогічних наук

Науковий керівник
СОЛОВЙОВ Володимир Миколайович,
доктор фізико-математичних наук, доцент

КРИВИЙ РІГ – 2000

ЗМІСТ

ВСТУП

В умовах розбудови національної системи вищої педагогічної освіти особливо актуальною стає професійна спрямованість навчання у педвузі та посилення зв'язку змісту навчання з повсякденним життям, що вимагає збільшення у навчальних планах долі курсів інтегративного характеру, які сприятимуть формуванню міжпредметних та міжгалузевих зв'язків, вихованню висококваліфікованих педагогів, цілеспрямована активна діяльність яких у значній мірі визначатиме майбутнє нашого суспільства. Одним з таких курсів є курс чисельних методів, що традиційно посідає центральне місце як у навчальних планах (III–IV курси), так і за своїм значенням, втілюючи у собі практичну спрямованість курсів математики та інформатики.

Тенденція до збільшення частки самостійної навчальної та науково-педагогічної діяльності студентів із одночасним зменшенням аудиторного навантаження вимагає таких форм навчальної роботи, що активізують пізнавальну діяльність студентів. Активізація пізнавальної діяльності студентів є одним з пріоритетних напрямків досліджень педагогіки вищої школи, оскільки в ній містяться джерела багатьох проблем формування особистості майбутнього вчителя: розвиток пізнавальних інтересів, самостійності, ініціативності, цілеспрямованості, відповідальності, вольових якостей тощо.

Проблема активізації пізнавальної діяльності студентів вимагає пошуку нових підходів до подальшого удосконалення форм, методів та засобів навчання. Різні аспекти цієї проблеми розглядали А.М. Алексюк, Л.П. Арістова, С.І. Архангельський, Ю.К. Бабанський, Д.Б. Богоявленська, М.Я. Ігнатенко, О.Н. Кабанова-Меллер, Р.А. Нізамов, О.М. Матюшкін, Н.А. Менчинська, С.Л. Рубінштейн, А.В. Петровський, Н.Ф. Тализіна, Т.І. Шамова, Г.І. Щукіна та інші.

Як показали дослідження, що в останні роки інтенсивно ведуться в Україні (М.І. Жалдак, Ю.С. Рамський, С.А. Раков, Ю.В. Триус, А.В. Пеньков, Ю.В. Горошко, Г.Ю. Цибко, Н.В. Морзе, І.М. Забара, О.Б. Жильцов, М.С. Головань, Т.І. Чепрасова та ін.), впровадження засобів НІТ у навчальний процес відіграє позитивну роль в активізації вивчення як окремих навчальних предметів, так і навчального процесу в цілому [42, 70, 73, 75, 217]. Разом з тим варто зазначити, що ряд аспектів цієї проблеми потребує подальшого дослідження. Поза увагою дослідників залишається проблема активізації пізнавальної діяльності студентів в процесі навчання чисельних методів. Відсутня методика цілеспрямованого розвитку пізнавальної активності в процесі навчання чисельних методів на основі об'єктного підходу, який має для цього значні дидактичні можливості.

Одне з основних методичних утруднень, що виникає при викладанні курсу чисельних методів, полягає у необхідності одночасного засвоєння обчислювальних алгоритмів та особливостей їх програмної реалізації в умовах дефіциту навчального часу. Існуючі курси чисельних методів передбачають, як правило, незалежне вивчення математичних основ деякого методу та його реалізації засобами процедурного програмування.

Традиційно, при вивченні обчислювального алгоритму оперують з абстракціями високого рівня – поліномами, матрицями, векторами тощо. Та при програмній реалізації у процедурній методології навіть прості операції з такими об'єктами породжують громіздкі конструкції з безліччю вкладених циклів, що віддаляє програму від алгоритму, а реалізацію – від метода, і це є першим аспектом проблеми. Другий її аспект – це повторюваність одних й тих самих обчислювальних процедур у різних розділах чисельних методів. Обидва аспекти в поєднанні із слабко вираженою практичною спрямованістю традиційних курсів чисельних методів призводять до

протиріччя між потенціалом цього предмету у формуванні особистості майбутнього фахівця та реальною педагогічною практикою.

Досить поширеним підходом при викладанні чисельних методів є використання спеціалізованих математичних бібліотек, проте існуючі зараз універсальні математичні бібліотеки є в основному результатом тривалої еволюції процедурного програмування і мають обмежені можливості для подальшого розвитку. Об'єктно-орієнтований підхід (ООП), який зарекомендував себе ефективною технологією в системному і прикладному програмуванні, забезпечує більш радикальні засоби для бажаної інтеграції і модифікації програмного забезпечення. В зв'язку з цим наукового і практичного значення набуває дослідження можливостей застосування ООП до програмування задач обчислювального характеру, а також створення єдиного об'єктно-орієнтованого математичного середовища для підтримки курсу чисельних методів.

Природна спільність принципів ООП і методологічних основ обчислювальної математики була досліджена у роботах С.В. Морозова [123], О.П. Поліщука [66], В.А. Семенова [169], О.А. Тарлапана [194], Є.Ю. Ширяєвої [173], Й. Арндта [234], К.Г. Баджа [238], Т. Вельдхайзена [263], Р. Позо та К. Ремінгтон [254], А.Д. Робінсона [256], М. Томміла [262], Б. Хайбле [241], К.С. Хорстманна [244] та інших, проте застосування ООП до вивчення курсу чисельних методів досі є недослідженою проблемою. Оскільки ООП забезпечує достатньо потужну і гнучку технологію розвитку існуючого програмного забезпечення, а класи математичної бібліотеки реалізують найзагальніші проблемно-інваріантні поняття, бібліотека класів може розглядатися в якості базового інструментального середовища для розробки курсу чисельних методів. При цьому інтерфейс бібліотечних класів утворює своєрідну – максимально наближену до природної математичної – мову, використання якої зближує етапи алгоритмічного

і програмного проектування і істотно полегшує програмування обчислювальних задач.

ОБ'ЄКТОМ ДОСЛІДЖЕННЯ є активізація пізнавальної діяльності студентів педвузів в процесі навчання чисельних методів засобами об'єктно-орієнтованого програмування.

ПРЕДМЕТОМ ДОСЛІДЖЕННЯ є методична система навчання чисельних методів у об'єктно-орієнтованій технології програмування.

ГІПОТЕЗА ДОСЛІДЖЕННЯ – систематичне і цілеспрямоване використання в процесі навчання чисельних методів об'єктного підходу та спеціалізованих об'єктно-орієнтованих математичних бібліотек дозволить суттєво активізувати пізнавальну діяльність студентів, привести підготовку з цієї дисципліни у відповідність до сучасних вимог впевненого володіння студентами НІТ (зокрема, новими технологіями програмування), підвищити практичну значущість вивчення чисельних методів у педвузі та природним способом інтегрувати етапи вивчення чисельних методів та їх програмування.

ОСНОВНА МЕТА ДОСЛІДЖЕННЯ – дослідити можливості об'єктного підходу щодо активізації пізнавальної діяльності студентів в процесі навчання чисельних методів та розробити методику викладання чисельних методів у об'єктно-орієнтованій технології програмування.

Відповідно до мети, було необхідно розв'язати наступні **ЗАВДАННЯ**:

1. Здійснити психолого-педагогічний аналіз сучасного стану досліджень з проблеми пізнавальної активності з метою встановлення факторів, що сприяють її розвитку, та з'ясування шляхів її формування при вивченні чисельних методів.

2. Проаналізувати еволюцію та сучасний стан вітчизняних та зарубіжних курсів обчислювальної математики і розробити зміст курсу чисельних методів для педвузів.
3. Розробити психолого-педагогічні основи активізації пізнавальної діяльності студентів при вивченні чисельних методів в об'єктній методології.
4. Дослідити можливості застосування об'єктно-орієнтованого програмування до розв'язання задач обчислювальної математики.
5. Провести об'єктно-орієнтований аналіз і проектування об'єктної бібліотеки математичних об'єктів, обчислювальних алгоритмів та чисельних проблем.
6. Створити програмно-методичне забезпечення курсу чисельних методів у об'єктно-орієнтованій технології програмування в педвузі.
7. Експериментально перевірити ефективність розробленої методики.

Для розв'язання поставлених завдань застосовувались такі **МЕТОДИ ДОСЛІДЖЕНЬ**: теоретичний аналіз наукової, психолого-педагогічної та методичної літератури з проблеми дослідження, педагогічне спостереження, бесіди, анкетування, тестування, аналіз досвіду роботи викладачів, об'єктно-орієнтований аналіз та проектування, педагогічний експеримент (констатуючий, пошуковий та формуючий) із статистичним аналізом його даних. Вибір методів дослідження визначався особливостями розв'язуваних нами завдань.

МЕТОДОЛОГІЧНОЮ ОСНОВОЮ дослідження є теорія пізнання, системно-структурний підхід до аналізу навчальної діяльності, положення психології та педагогіки про активність особистості у процесі пізнання, основні положення теорії діяльності, загально-дидактичні положення. В ході дослідження враховувались також основні положення концепції об'єктних наукових обчислень.

НАУКОВА НОВИЗНА ДОСЛІДЖЕННЯ полягає в розробці методики викладання чисельних методів у об'єктно-орієнтованій технології програмування, спрямованої на активізацію пізнавальної діяльності студентів.

ТЕОРЕТИЧНЕ ЗНАЧЕННЯ ДОСЛІДЖЕННЯ полягає в такому:

1. Запропоновано загальні підходи до моделювання об'єктів чисельного аналізу як комп'ютерних інтерпретацій відповідних алгебраїчних структур.
2. Досліджено дидактичні можливості використання об'єктного підходу до викладання курсу чисельних методів.

ПРАКТИЧНЕ ЗНАЧЕННЯ ДОСЛІДЖЕННЯ визначається тим, що:

1) розроблено методику активізації пізнавальної діяльності студентів в процесі навчання чисельних методів у об'єктно-орієнтованій технології програмування;
2) розроблено ефективні педагогічні програмні засоби на основі ООП;
3) розроблено об'єктні бібліотеки мовами C++ та Паскаль:
   - класів цілих та раціональних чисел необмеженої довжини;
   - параметризованих векторних, поліноміальних та матричних об'єктів;
   - алгоритмічних та проблемних класів.
4) розробка висунутих теоретичних положень доведена до практичної реалізації у вигляді навчальних посібників з чисельних методів та автоматики.

ОСОБИСТИЙ ВНЕСОК ЗДОБУВАЧА. У працях, опублікованих у співавторстві, автору належать такі результати:

1. Досліджено можливості об'єктно-орієнтованого підходу до побудови інформаційних моделей.

2. Розроблено факультативний курс фізики твердого тіла з використанням векторно-матричних бібліотек для структурних розрахунків.
3. Обґрунтовано необхідність застосування об'єктно-орієнтованих математичних бібліотек при проведенні електротехнічних розрахунків.
4. Створено програмно-методичний комплекс для комп'ютерної підтримки курсів алгебри та числових систем.

НА ЗАХИСТ ВИНОСЯТЬСЯ:

1. Об'єктний підхід є ефективним засобом дослідження об'єктів числової природи шляхом імітації їх суттєвих властивостей у класах мов програмування, що дозволяє розширити можливості мови програмування по оперуванню з такими типами даних, як вектори, поліноми, матриці і т.п. та наблизити запис програм до природної математичної нотації.
2. Застосування об'єктного підходу при навчанні чисельних методів сприяє активізації пізнавальної діяльності студентів та підвищує результативність процесу навчання.

ОБГРУНТОВАНІСТЬ ТА ВІРОГІДНІСТЬ результатів і висновків дисертаційного дослідження забезпечується методологічними основами дослідження, відповідністю методів дослідження його меті і завданням, аналізом значного обсягу теоретичного та емпіричного матеріалу, результатами статистичного аналізу даних, отриманих в ході масового педагогічного експерименту, широким впровадженням як в навчальній, так і в науковій діяльності.

АПРОБАЦІЯ РОБОТИ. Результати дослідження обговорювались на:

- Всеукраїнській конференції «Освітні стандарти та зміст шкільних і вузівських курсів інформатики» (Бердянськ, 1997);
- III Міжнародній науково-технічній конференції «Сучасні технології в аерокосмічному комплексі» (Житомир, 1997);

- Міжнародній науково-практичній конференції «ELBRUS’97. Новые информационные технологии и их региональное развитие» (Нальчик, 1997);
- Міжнародній науково-практичній конференції «Проблеми електронної промисловості у перехідний період» (Луганськ, 1998);
- Всеукраїнському семінарі «Нові інформаційні технології навчання» (Київ, 1998);
- Всеукраїнській конференції «Комп’ютерне моделювання та інформаційні технології в освітній діяльності» (Кривий Ріг, 1999);
- Всеукраїнському науково-методичному семінарі «Інформаційні технології в навчальному процесі» (Одеса, 1999);
- Міжвузівській науковій конференції «Математика, її застосування та викладання» (Кіровоград, 1999);
- VII Міжнародній конференції «Математика. Компьютер. Образование» (Дубна, 2000);
- Всеукраїнській конференції «Комп’ютерне моделювання та інформаційні технології в природничих науках» (Кривий Ріг, 2000).

Результати дослідження впроваджувались автором в процесі викладання курсів «Чисельні методи в об’єктній методології», «Методи математичного моделювання» у Криворізькому державному педагогічному університеті, «Методи обчислень» у Криворізькому технічному університеті, у виступах на міському постійно діючому науково-методичному семінарі «Комп’ютерне моделювання та інформаційні технології в освітній діяльності», при керівництві конкурсними, курсовими та дипломними роботами.

ПУБЛІКАЦІЇ. За матеріалами дослідження опубліковано 19 робіт, з них навчальних посібників – 2, методичних розробок – 2, статей – 8, матеріалів та тез конференцій – 7.

РОЗДІЛ І. ПСИХОЛОГО-ПЕДАГОГІЧНІ ОСНОВИ АКТИВІЗАЦІЇ ПІЗНАВАЛЬНОЇ ДІЯЛЬНОСТІ СТУДЕНТІВ В ПРОЦЕСІ НАВЧАННЯ ЧИСЕЛЬНИХ МЕТОДІВ ЗАСОБАМИ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ

§ 1.1. Активізація пізнавальної діяльності як психолого-педагогічна проблема

В педагогічній науці немає єдиного підходу до означення поняття *пізнавальної активності* студентів, а тому відсутні єдині характерні ознаки пізнавальної активності та критерії її визначення.

Тлумачний словник С.І. Ожегова наводить таке значення слова «активний» – діяльний, енергійний, діючий, той, що розвивається [130, с. 24]. Між активністю та діяльністю, в якій формується активність, існує діалектичний взаємозв'язок, проте пізнавальну активність не можна зводити до пізнавальної діяльності [58, 62]. Якби це було можливо, проблема активізації навчання розв'язувалася б сама собою: будь-яка діяльність студентів є активною. С.Л. Рубінштейн у [159] відмічає, що «… про будь-який психічний процес сприймання, мислення тощо ми кажемо, що він становить єдність змісту й процесу, та підкреслюємо його активний характер. Але мислення як процес – це для нас активність, а не діяльність. … Діяльність у власному розумінні слова – це предметна діяльність, практика».

Питання активізації пізнавальної діяльності було в центрі уваги Л.П. Арістової, М.М. Ацибора, Ю.К. Бабанського, М.Д. Бойправа, В.М. Вєргасова, А.Ф. Есаулова, М.І. Єнікеєва, Б.П. Єсипова, П.І. Зінченко, Л.А. Іванової, М.Я. Ігнатенка, І.Я. Ланіної, М.І. Махмутова, Р.А. Нізамова, І.Т. Огородникова, Н.О. Половникової, Т.І. Шамової, Г.І. Щукіної та інших вчених.

На думку Н.О. Половникової, пізнавальна активність – це готовність

до енергійного опанування знань при наполегливих систематичних вольових зусиллях [146, с. 25]. М.І. Махмутов розглядає пізнавальну активність як виявлення в навчальному процесі вольових, емоціональних та інтелектуальних якостей особистості [114, с. 44].

Л.П. Арістова [9, с. 31] вважає, що активність пізнавальної діяльності виявляється в перетворюючій діяльності людини – це виявлення перетворюючого, творчого ставлення індивіда до об'єктів пізнання, яке передбачає наявність таких компонентів активності:

– вибір підходів до об'єкту пізнання;
– постановка завдання, яке потрібно виконати;
– перетворення об'єкта в наступній діяльності.

За Б.П. Єсиповим, пізнавальна активність – це свідоме, цілеспрямоване виконання розумової або фізичної роботи, необхідної для опанування знань, умінь та навичок [67, с. 14].

М.П. Лебєдєв, визначаючи поняття пізнавальної активності, вказує на роль інтересу: «пізнавальна активність – це ініціативне, дійове ставлення учнів до засвоєння знань, а також виявлення ними інтересу, самостійності і вольових зусиль у навчанні» [101, с. 7].

В.С. Тюхтін [202] бачить ознаки пізнавальної активності в можливості виходу за межу відображення прихованих суттєвих (закономірних) зв'язків речей, яке досягається чуттєво. Це, зокрема, поєднання і пов'язування різних чуттєвих образів, наукове передбачення перебігу подій та процесів, свідома постановка не лише найближчої, а й більш віддаленої мети, втілення ідеальних моделей дій у практичні дії тощо.

Т.І. Шамова розуміє пізнавальну активність як «якість діяльності особистості, яка виявляється у відношенні учня до змісту та процесу діяльності, у його прагненні до ефективного оволодіння знаннями та способами діяльності за оптимальний час, в мобілізації морально-вольових зусиль на досягнення навчально-пізнавальної мети» [219, с. 48-49].

Л.А. Аврамчук на розвиток думки Т.І. Шамової під активною пізнавальною діяльністю пропонує розуміти таку навчально-пізнавальну діяльність, в процесі якої викладач мобілізує інтелектуальні сили студентів на досягнення конкретної мети навчання та виховання, і відбувається свідоме цілеспрямоване пізнання нового, проявляється здатність до самостійного творення [1, с. 123].

Г.І. Щукіна пізнавальну діяльність розглядає як особистісне утворення, що виражає інтелектуальний відгук на процес пізнання, живу участь, розумово-емоційну чуйність у пізнавальному процесі [228, с. 116]. Вона характеризується:

– пошуковою спрямованістю в навчанні;
– пізнавальним інтересом, прагненням задовольнити його за допомогою різних джерел як в навчанні, так і в позанавчальній діяльності;
– емоціональним піднесенням, благополуччям перебігу діяльності.

І.Ф. Харламов визначає пізнавальну активність як стан, що характеризується прагненням до навчання, розумовим напруженням і виявом вольових зусиль в процесі опанування знань [214, с. 31]. Р.А. Нізамов уточнює це означення, додаючи до визначених І.Ф. Харламовим характерних ознак напругу уваги та фізичних сил для досягнення поставленої мети [128, с. 34].

Згідно означення Г.Ц. Молонова [121, с. 55-56], пізнавальна активність – це складне духовне утворення, яке залежить від рівня підготовки студента та виявляється:

– в психічному плані – як здатність до діяльності;
– в інтелектуальному плані – як здатність до розв’язування пізнавальних задач;
– в моральному плані – як мотивоване відношення до предмету пізнання.

Отже, в наведених означеннях стверджується, що *пізнавальна активність характеризує індивідуальні особливості людини в процесі пізнавальної діяльності і є властивістю (якістю) особистості*.

Активізація пізнавальної діяльності студентів є процесом і результатом стимулювання їх пізнавальної активності. М.Я. Ігнатенко, розглядаючи активізацію пізнавальної діяльності учнів, пропонує таке її означення: «Активізація навчально-пізнавальної діяльності учнів – це мобілізація вчителем (за допомогою спеціальних засобів) інтелектуальних, морально-вольових та фізичних сил учнів на досягнення конкретної мети навчання, виховання і всебічного розвитку школярів, на посилену спільну навчально-пізнавальну діяльність вчителя та учнів, на спонукання до її енергійного цілеспрямованого здійснення, на подолання інерції, пасивності, стереотипних форм викладання і навчання». З.І. Слєпкань, беручи це означення у якості основного, розуміє під *активізацією навчально-пізнавальної діяльності студентів* цілеспрямовану діяльність викладача, спрямовану на розробку і використання такого змісту, форм, методів, прийомів і засобів навчання, які сприяють підвищенню пізнавального інтересу, активності, творчої самостійності студентів у засвоєнні знань, формуванні навичок і вмінь застосовувати їх на практиці [184, с. 46].

Основним завданням дидактики вищої школи З.І. Слєпкань вважає активізацію пізнавальної діяльності студентів не лише окремими способами, прийомами, а й активізацію всього навчально-виховного процесу, виявлення системи методів, способів, прийомів, організаційних форм і засобів, що сприяють підвищенню активності в процесі пізнання, виділяючи наступні психолого-педагогічні умови:

- забезпечення єдності освітньої, розвиваючої та виховної цілей процесу навчання;
- педагогічно правильне використання принципів дидактики вищої школи;

- забезпечення емоційності навчання і створення сприятливої атмосфери;
- динамічність, різноманітність методів, прийомів, форм і засобів викладання і навчання, їх спрямованість на розвиток активної дослідницької діяльності студентів, надання пріоритетів методам і формам активного навчання;
- орієнтація студентів на систематичну самостійну роботу, забезпечення регулярності і ефективності контролю і оцінки успішності студентів;
- комплексне, педагогічно доцільне використання технічних засобів навчання і нових інформаційних технологій;
- використання системи психологічних і педагогічних стимуляторів активної навчальної діяльності.

Відзначаючи відмінність та зв'язок активності та самостійності студентів, З.І. Слєпкань зауважує, що самостійність студента як систематична робота на матеріалом на заняттях і в позаурочний час сприяє активності, а прояв активності спрямовує особистість до самостійної роботи [184, с. 48].

Пізнавальна самостійність, що формується на базі активності, характеризується багатьма вченими як якість особистості, її властивість. Так, М.М. Скаткін визначав активність і самостійність як якість особистості та вказував, що пізнавальна самостійність формується на базі активності. Ознаками пізнавальної самостійності при цьому є прагнення та вміння самостійно мислити, здатність орієнтуватися в новій ситуації, знайти свій підхід до нової задачі, бажання не лише зрозуміти засвоювані знання, а й способи їх здобування, критичний підхід до суджень інших та незалежність власних суджень [183, с. 19].

Л.С. Коновалець розглядає пізнавальну самостійність як якість особистості, що поєднує у собі вміння набувати нові знання та творчо застосовувати їх в різних ситуаціях із прагненням до такої роботи. Цей феномен

являє собою єдність двох компонентів – мотиваційного та процесуального (змістовно-операційного). Перший відображає потребу у процесі пізнання, другий – знання даної предметної області та прийоми діяльності, що сприяють здійсненню цілеспрямованого пошуку [92, с. 47].

Диференціюючи поняття активності та самостійності, Л.П. Арістова зазначає, що «сутність самостійності – у здатності об'єкта діяти без сторонньої допомоги» [9, с. 34]. І.Я. Лернер вважає активність умовою самостійності (тому що не можна бути самостійними, не будучи активним), і головне завдання вбачає в тому, щоб підняти активність до рівня самостійності [104].

В.А. Крутецький встановлює такий зв'язок між самостійністю та активністю: «Відношення між поняттями «активне мислення», «самостійне мислення» та «творче мислення» можна позначити у вигляді концентричних кіл. Це різні рівні мислення, з яких кожен наступний є видовим по відношенню до попереднього, родового. Творче мислення буде самостійним та активним, проте не всяке активне мислення є самостійним і не всяке самостійне мислення є творчим» [96, с. 180]. Під стилем мислення Ю.В. Сенько розуміє «підхід учня до навчально-пізнавальної діяльності і її результатів, спрямований на досягнення пізнавальних і практичних цілей» [181, с. 180].

Головним у природі пізнавальної активності є *мотивація*. Це поняття включає в себе всі види спонукань: інтереси, мотиваційні установки, потреби, прагнення, цілі тощо. Мотивація пронизує основні структурні утворення особистості: емоційно-вольову сферу, здібності, спрямованість, характер, психічні процеси. Саме мотивація відіграє роль того вектора, який здатен спрямувати діяльність студента на досягнення поставлених викладачем цілей.

Розглядаючи пізнавальну активність як якість особистості, у якій виявляється перш за все відношення студента до предмету та процесу діяль-

ності, ми вважаємо формування у студентів позитивних мотивів учіння однією з головних умов пізнавальної активності.

Внутрішніми стимулами пізнавальної активності є *пізнавальні потреби*, які відрізняються від інших мотиваційних утворень саме тим, що вже містять у собі спонукання до активності – без такого спонукання активність не може відбутися. Мотив не є потребою, але безпосередньо пов'язаний з нею. Потреба і мотив – не тотожні утворення. Мотив – це те, що спонукає до діяльності, спрямовує її на задоволення певної потреби або декількох потреб. Пізнавальна потреба завжди спрямована на конкретні об'єкти, тобто на усвідомлені мотиви, активність над якими має задовольнити цю потребу.

З одного боку, мотив є джерелом діяльності людини, а з іншого – сам *мотив формується в процесі виконання дій по усвідомленню протиріччя між виниклою пізнавальною потребою та можливістю її задоволення* власними силами. Прояв самостійності в пізнавальній діяльності обов'язково пов'язаний з її мотивом.

На думку В.С. Ільїна, «пізнавальна потреба являє собою суб'єктивне відображення об'єктивної потреби суспільства в знаннях, це переживання нужди людини в пізнавальній діяльності, в функціонуванні психічних процесів нейродинамічних структур, за допомогою яких здійснюється пізнання» [80, с. 123]. Лише тоді, коли є необхідність, яка спонукає людину до діяльності, стимулюється і активність особистості.

З потребою тісно пов'язана *мета*, яка є функцією людини, оскільки будь-яка діяльність є реалізацією деякої мети. Приступаючи до певної діяльності, людина ставить перед собою конкретну мету, виходячи з потреб. Тому мета і визначається як усвідомлена потреба, ознаменування бажаного результату, на досягнення якого і спрямована активність особистості.

Мета є об'єктом, на який спрямована активність, але й активність є умовою реалізації мети – без активності неможливі як визначення мети,

так і діяльність по її досягненню. В той же час без мети неможлива активність, оскільки у постановці мети вже виявляється активність, тобто постановка мети теж є показником пізнавальної активності.

Постановка мети та її реалізація передбачають певні мотиви діяльності. У мотиві відбувається поєднання, синтез зовнішніх та внутрішніх сил, які визначають характер діяльності суб'єкта. Якщо потреба виражає необхідність, мета – конкретизовану потребу, то мотиви характеризують внутрішні причини цих процесів.

Система потреб та мотивів відбивається в інтересах. Формою виявлення пізнавальних потреб є *пізнавальний інтерес*, що детермінований мотивами діяльності, тому пізнавальний інтерес виявляє і потреби, і мотиви. Пізнавальні інтереси характеризуються параметрами стійкості, локалізації та усвідомленості.

І.Я. Ланіна під пізнавальним інтересом розуміє «… вибіркову спрямованість особистості, звернену до області пізнання, до її предметного боку та самого процесу оволодіння знаннями. Своєрідність пізнавального інтересу полягає в тенденції людини, що володіє пізнавальним інтересом, поглибитися в суть пізнаваного» [100, с. 4].

І.П. Павлов пов'язував вияв інтересу з безумовним рефлексом «що таке?». Цей рефлекс відповідає ситуативному інтересу, який може бути мотивом діяльності, головне в ньому – новизна інформації. Механізм пізнавального інтересу значно складніше, ніж просто відповідь на зовнішній подразник. Не все нове, з чим стикається людина у житті, стає предметом його інтересу. Коли ті чи інші поняття, предмети чи явища вона вважає важливими, життєво значимими, тоді вона із захопленням ними займається, намагається все це глибоко вивчити. У противному випадку інтерес буде мати випадковий, поверховий характер.

Пізнавальний інтерес – вибіркова спрямованість особистості на предмети і явища навколишньої дійсності, що характеризується постійним

прагненням до пізнання. Систематично зміцнюючи і розвиваючись, пізнавальний інтерес позитивно впливає не тільки на процес і результат діяльності, але і на протікання психічних процесів – мислення, уяви, пам'яті, уваги, що під впливом пізнавального інтересу здобувають особливу активність і спрямованість.

Активізація пізнавальної діяльності без розвитку пізнавального інтересу неможлива, тому в процесі навчання необхідно систематично збуджувати, розвивати і зміцнювати пізнавальний інтерес і як важливий мотив навчання, і як стійку рису особистості.

Пізнавальний інтерес спрямований не тільки на процес пізнання, але і на результат його, а це завжди пов'язано з прагненням до мети, з реалізацією її, подоланням труднощів, з вольовою напругою і зусиллям. В пізнавальний інтерес включені, отже, і вольові процеси, що сприяють організації, протіканню і завершенню діяльності.

Г.І. Щукіна виділяє такі три рівні розвитку пізнавального інтересу [228, с. 97]:

1. Елементарний рівень, що характеризується відкритим, безпосереднім інтересом до нових фактів та цікавих явищ.
2. Середній рівень, який характеризується інтересом до пізнання суттєвих властивостей предметів чи явищ, що вимагає пошуку, здогадки, активного оперування наявними знаннями та засвоєними засобами. На цьому рівні однаково фіксуються як зовнішні ознаки, так і суттєві властивості виучуваного явища.
3. Вищий рівень, що характеризується інтересом до причинно-наслідкових зв'язків, вияву закономірностей, встановленню загальних принципів явищ, що діють в різних умовах. Цей рівень іноді містить елементи дослідницької діяльності.

Т.І. Шамова виділяє такі компоненти пізнавальної діяльності [219, с. 128]:

– мотиваційний;
– орієнтаційний;
– змістовно-операційний;
– вольовий;
– оціночний.

На думку М.С. Голованя [42], до **структури** пізнавальної діяльності входять пізнавальна потреба, пізнавальний інтерес, а також потреби у самовираженні особистості, у самовдосконаленні, а також загальна спрямованість особистості на навчання, яка забезпечується конкретними мотивами учбової діяльності. До *мотиваційного компоненту* пізнавальної активності можна також віднести пізнавальну ініціативу та пізнавальну надситуативність (вихід за межі завдання, коли студент за власною ініціативою робить більше, ніж від нього вимагалося).

Прояв та функціонування пізнавальної активності на рівні її *змістовно-операційного компонента* відбувається в процесі перетворювальної діяльності з метою пізнання. Така діяльність може бути творчою, самостійною, а також навчальною, коли вчитель керує ходом діяльності. Активність може проявити себе в будь-якій з цих діяльностей, але найбільший її прояв припадає саме на творчу діяльність – вищий ступінь прояву пізнавальної активності.

Змістовно-операційний компонент включає в себе володіння студентом системою знань, умінь і способів навчання, адже стійке прагнення до поповнення знань можливо лише за тієї умови, коли студент вже володіє певним багажем знань і має уміння їх здобувати. Крім того, операційна сторона навчання тісно пов’язана з мотивацією, з пізнавальними інтересами. Самостійна пізнавальна діяльність свідчить про високий рівень розвитку пізнавальної активності.

Третій компонент пізнавальної активності – *емоційно-вольовий*. Потреби та інтереси відіграють ведучу роль, проте, якщо вони не набули до-

статнього розвитку, то на перший план виходить вольова складова. Активність не здійсниться без сформованих вольових рис, оскільки вона відбувається на післядовільному рівні регуляції, який передбачає сформованість вольової сфери.

Емоційно-вольовий компонент характеризується прагненням до подолання пізнавальних труднощів і наявність емоційного настрою, пов'язаного з успішністю навчання. Прагненням близькі такі характеристики, як потяг та бажання – вони є передумовами до стану прагнення. Від потягу та бажання прагнення відрізняється своєю усвідомленістю.

Морально-вольові процеси є основою розвитку прагнення; емоційні процеси підсилюють і збагачують як вольові, так і розумові процеси.

Таким чином, **пізнавальну активність можна розглядати як складне інтегративне утворення особистості, що складається з трьох компонентів: мотиваційного, змістово-операційного та емоційно-вольового**.

Структуру пізнавальної активності та основні складові кожного з її компонентів зображено на рис. 1.1.

Визначивши структуру пізнавальної активності, розглянемо існуючі в психолого-педагогічній літературі *класифікації рівнів розвитку пізнавальної активності*.

Так, Л.С. Коновалець [92, с. 47] у відповідності до ступеня готовності та прагнення студентів до самостійної роботи над завданнями відтворюючого, реконструктивно-варіативного, частково-пошукового та дослідницького характеру виділяє чотири рівня пізнавальної активності:

1) Завдання відтворюючого типу виконуються за допомогою вправ, що мають такі ознаки:
   - сформована в них проблема не вимагає виділення підпроблем;
   - завдання має єдиний шлях розв'язання;
   - усі необхідні дані присутні, немає зайвих та відсутніх;

Рис. 1.1. Структура пізнавальної активності

– розв’язання завдання передбачає використання відомих способів дій.

2) Реконструктивно-варіативний рівень характеризується тим, що:
   – для розв’язання завдання необхідно виділити підпроблеми;
   – завдання має єдиний шлях розв’язання;
   – умова завдання може містити зайві або відсутні дані;
   – завдання вимагає перенесення відомих способів дій в аналогічну внутрішньопредметну ситуацію.
3) В завданнях частково-пошукового типу студенти розв’язують проблеми, поставлені викладачем, виділяючи декілька підпроблем, при цьому не лише використовуючи накопичений досвід, а й виходячи за його межі. Ознаки таких завдань:
   – для розв’язання завдання необхідно виділити підпроблеми;
   – завдання може мати два та більше способів розв’язання;
   – умова завдання може містити зайві або відсутні дані;
   – завдання вимагає перенесення декількох відомих способів розв’язання та їх комбінування в межах внутрішньопредметної області.
4) Найвищий рівень пізнавальної активності – творчий. Він виявляється у ході виконання завдань дослідницького характеру, коли необхідно оволодіти методами та прийомами пізнання, які дозволяють побачити нову проблему у знайомій ситуації, знайти нові способи застосування засвоєних знань.

Д.Б. Богоявленська [18] залежно від характеру пізнавальної активності суб’єкта розглядає такі рівні інтелектуальної активності:

I. Репродуктивний. Нижня межа цього рівня характеризується пасивністю, інертністю, відсутністю інтелектуальної ініціативи: студенти, які перебувають у такій активності, залишаються в ра-

мках знайденого способу дій.

II. Евристичний. Характеризується прагненням удосконалювати дану діяльність, шукати нові способи розв'язання завдань, тобто це рівень самостійного оригінального розв'язання поставленої задачі.

III. Креативний, вищий за якістю рівень, який характеризується ініціативою в постановці задачі, у здійсненні цілепокладання, в умінні переходити до теоретичних узагальнень, коли емпіричні закономірності стають самостійними проблемами, тобто студент сам ставить творчу задачу або мету, сам намагається її розв'язати, а, отже, здійснює надситуативність.

Т.І. Шамова виділяє три рівні пізнавальної активності: репродуктивну, пошуково-виконавську, творчу [219, с. 52-53].

Г.І. Щукіна фіксує такі три рівні пізнавальної активності: репродуктивно-наслідувальну, пошуково-виконавську і творчу [228, 229].

Аналізуючи етап творчої пізнавальної активності, Н.О. Половникова у [146] показує, що творчість – антипод наслідування, проте наслідування – невід'ємна частина становлення пізнавальної самостійності, тому необхідна така система навчання, яка спонукає до творчого рівня учіння [107]. Серед прийомів, які застосовуються при такому навчанні, автор виділяє перенесення знань та вмінь у нову ситуацію, «відсторонення» знань, самостійне комбінування засвоєних прийомів в нових умовах, різностороннє бачення діалектики об'єкту тощо.

Г.І. Щукіна показниками творчої активності вважає наступні новизну, оригінальність, відсторонення, відхід від шаблону, ломку традицій, несподіваність, доцільність, цінність [228, с. 29]. Т.І. Шамова до зазначених вище показників додає інтерес до теоретичного осмислення виучуваних явищ та процесів [219, с. 53-54].

І.Я. Ланіна пропонує таку схему класифікації динаміки «захоплення учня навчальним предметом: від цікавості до подиву, від нього – до активної допитливості та бажання знати, від них – до міцного знання та наукового пошуку» [100, с. 5].

На першій стадії – подиву та цікавості – виникає ситуативний інтерес, який згасає та швидко зникає при зміні ситуації на занятті. Цікавість, як початкова стадія пізнавальної активності учня, характеризується тим, що її об'єктом є не зміст предмету, а лише зовнішні форми роботи на занятті.

У міру збагачення запасу конкретних знань в процесі учбової діяльності відбувається все більша об'єктивізація інтересу: все більшого значення набуває реальний зміст об'єкту інтересу. Цікавість переростає у допитливість – більш високий рівень розвитку пізнавальної активності, при якому на перший план виходить установка на пізнання. Стадія допитливості характеризується потягом до більш глибокого ознайомлення з предметом, на цій стадії студенти багато питають, сперечаються, намагаються самостійно знайти відповіді на свої питання та питання товаришів.

Наступна стадія – наявність пізнавального інтересу – виявляється у прагненні до оволодіння міцними знаннями з предмету, що пов'язано з вольовими зусиллями та напруженням думки, із застосуванням знань на практиці.

Л.А. Іванова дещо інакше трактує цю схему: цікавість, допитливість та стійкий інтерес вона відносить не до пізнавальної діяльності, а до пізнавального інтересу [77, с. 14], тому що розгляд цих компонентів як етапів розвитку пізнавальної активності звужує дане поняття до пізнавального інтересу.

На наш погляд, найбільш вдалою є інтегративна класифікація рівнів пізнавальної активності, запропонована у дисертаційному дослідженні

М.С. Голованя [42].

Нульовий рівень характеризується відсутністю пізнавальної активності студента – ним володіє пасивність, формальність у виконанні завдання. М.П. Руденко називає цей рівень рівнем пасивності [160].

Перший рівень розвитку характеризується тим, що в студента з'являється деяка активність в сприйнятті матеріалу, він починає активно стежити за ходом міркувань викладача. Таку активність позначають як ситуативну, вона не сформована як риса особистості, тому що не сформована навчальна діяльність. Але вона проявляється як стан: її можна охарактеризувати як імпульсивну пізнавальну активність, або штучну, викликану сильно діючими зовнішніми емоційними стимулами. Для стимуляції пізнавальної активності цього рівня важливим є майстерність пояснення викладача, його вміння пробудити в студентах ситуативну пізнавальну активність.

Другий рівень розвитку пізнавальної активності студента характеризується активною навчальною діяльністю. Ця діяльність ще може і не бути самостійною або творчою, але є вже активною, тобто розумово перетворювальною діяльністю, яка пробуджується завдяки вмінню викладача вводити студента в ситуацію прояву пізнавальної активності. На цьому рівні її розвитку повинен бути сформований змістовно-операційний компонент пізнавальної активності, пізнавальний інтерес, але пізнавальна потреба ще тільки формується. Цей рівень можна визначити як початок становлення пізнавальної активності як риси, що починає характеризувати особистість студента, проте ця риса ще не стійка і може блокуватися різними негативними утвореннями або індивідуальними особливостями, вона не є усталеною, оскільки ще не сформована навчальна діяльність.

Третій рівень розвитку пізнавальної активності характеризується сформованістю всіх структурних компонентів як пізнавальної активності, так і навчальної діяльності. Пізнавальна активність набуває статусу влас-

тивості особистості саме тому, що сформовані всі компоненти навчальної діяльності, головне ж – сформованим є мотиваційний компонент не лише навчальної діяльності, до якого належать внутрішні пізнавальні мотиви, а й мотиваційний компонент пізнавальної активності, до якого, крім мотиваційних утворень, входять спрямованість особистості на навчання, пізнавальна ініціатива, пізнавальна надситуативність, більш інтегровані особистісні утворення. Пізнавальна потреба при цьому серед інших потреб має домінуючий вплив на прояв активності. Навчальна діяльність на цьому рівні розвитку пізнавальної активності може бути не тільки самостійною, але й творчою. Оскільки процес розумової діяльності приносить задоволення, вольова саморегуляція її відбувається на післядовільному рівні.

Оскільки нульовий рівень активності не містить у собі пізнавальної активності, ми його не розглядатимемо. Таким чином, **виділено три рівні пізнавальної активності**.

Існують різні методики вимірювання рівнів пізнавальної активності.

М.С. Головань пропонує діагностувати структурні компоненти пізнавальної властивості, показані на рис. 1.1, за п’ятибальною шкалою:

1 бал – якість не розвинена, або вміння не сформоване;

2 бали – якість або вміння проявляють себе рідко, що безпосередньо позначається на діяльності;

3 бали – властивість або вміння помітні в діяльності, проявляються, проте не завжди якісно;

4 бали – вміння ефективне, але його ще можна вдосконалювати;

5 балів – вміння сформоване, ефективне в навчальній діяльності та рефлексується.

Т.І. Шамова пропонує для оцінки ефективності впливу системи засобів активізації учіння такі критерії:

1) *рівень свідомості засвоєння знань*: знання вважаються засвоєні

студентом свідомо, якщо він їх пам'ятає (або знає, як поновити у пам'яті) і, головне, вміє ними оперувати;

2) *міцність знань та навичок* – перевіряється шляхом включення у тестовий (контрольний) матеріал розділів, вивчених раніше;
3) *сформованість вмінь опрацьовувати навчальну інформацію*: порівнювати, виділяти головне, узагальнювати, конкретизувати;
4) *сформованість учбових вмінь*: планувати навчальну діяльність та здійснювати її самоконтроль [219, с. 128].

На основі виділених Т.І. Шамовою критеріїв нами в ході констатуючого експерименту було розроблено діагностичну таблицю оцінки рівня розвитку пізнавальної активності студентів III курсу при вивченні курсу чисельних методів (табл. 1.1).

Таблиця 1.1

Критерії та рівні активності пізнавальної діяльності студентів при вивченні курсу чисельних методів

Рівень / *Критерій*	***Низький***	***Середній***	***Високий***
Свідомість засвоєння знань	Теоретичні знання з предмету мають уривчастий характер, слабо алгоритмізовані, можуть бути застосованими на практиці лише за допомоги викладача	Теоретичні знання з предмету мають систематичний характер, слабо алгоритмізовані, можуть бути застосованими на практиці лише у стандартних ситуаціях	Теоретичні знання з предмету мають систематичний характер, алгоритмізовані, застосовуються на практиці як у стандартних ситуаціях, так і у змінених ситуаціях

Рівень / *Критерій*	***Низький***	***Середній***	***Високий***
Міцність знань та навичок	Відсутнє розуміння зв'язку між різними розділами чисельних методів, вивчені раніше методи не запам'ятовуються та не застосовуються	Розуміння зв'язку між різними розділами чисельних методів присутнє, вивчені раніше методи запам'ятовуються та частково застосовуються	Глибоке розуміння зв'язку між різними розділами чисельних методів присутнє, вільне володіння вивченим матеріалом та самостійне опрацювання нового
Вміння опрацьовувати навчальну інформацію	Допускаються помилки в процесі виконання елементарних інтелектуальних операцій, самостійний вибір раціональних прийомів програмування відсутній	Елементарні задачі розв'язуються правильно, за допомоги викладача – і більш складні, вибір раціональних прийомів програмування обмежується одним-двома способами	Як елементарні, так і складні задачі розв'язуються легко і правильно, при програмуванні методу самостійно обираються найбільш раціональні прийоми
Учбові вміння	Вольові зусилля нестійкі, ініціатива та самоконтроль відсутні, навчальна діяльність постійно коригується викладачем	Вольові зусилля та ініціатива проявляються за наявності стимулів, самоконтроль застосовується нерегулярно, навчальна діяльність потребує нерегулярної корекції викладачем	Вольові зусилля та ініціатива сталі, регулярно застосовується самоконтроль, діяльність спрямована на досягнення пізнавальних цілей

В ході дослідження ми дійшли висновку, що рівень пізнавальної активності студентів є недостатнім для успішного засвоєння навчального матеріалу. Так, констатуючий експеримент показав, що високий рівень розвитку пізнавальної активності мають 6% студентів, середній – 45%, а низький – 49%. Всебічному розвитку студентів в найбільшій мірі відповідають високий та середній рівні розвитку пізнавальної активності. Низький рівень пізнавальної активності майже у половини студентів вказує на суттєве відставання у розвитку мотиваційного, змістовно-операційного та емоційно-вольового компонентів пізнавальної активності, а отже – і відповідних якостей особистості майбутнього вчителя математики.

Ю.С. Рамський підкреслює, що курс чисельних методів має «значні потенційні можливості в підвищенні рівня теоретичної підготовки та інформаційної культури вчителя математики, які поки що не використовуються в повній мірі» [154, с. 25]. Вивчення цього курсу за відповідної методики навчання «… сприяє розвитку у студентів творчого, теоретичного мислення, а також формуванню операційного мислення, направленого на пошук оптимальних розв’язків. Студенти набувають не тільки певні знання з предмету, але й вміння думати, шукати і знаходити власні розв’язки, відношення до творчого застосування набутих знань, що так важливо для випускника вищої школи» [154, с. 46].

Виявлена розбіжність між високим потенціалом цього предмету по формуванню інформаційної культури педагога та низьким рівнем пізнавальної активності при його вивченні вимагає цілеспрямованої роботи викладача по формуванню і розвитку пізнавальної активності студентів, яка є запорукою підвищення якості засвоєння навчального матеріалу, розвитку мислення та творчих здібностей студентів.

§ 1.2. Еволюція та сучасний стан курсу чисельних методів у вищий школі

Чисельні методи мають досить поважний вік. За влучним виразом Е.Д. Бута, «отримуючи результати у формі чисел, сучасний обчислювач є безпосереднім спадкоємцем печерної людини, яка підраховувала кількість своїх жінок приведенням їх у взаємно однозначну відповідність з пальцями своєї руки» [24].

Відомо, що, основною діяльністю вчених Вавилона було складання математичних таблиць. Прикладом може слугувати табличка, яка збереглася до наших часів, датована 2000 р. до н.е., що містить квадрати чисел від 1 до 60. На іншій табличці 747 р. до н.е. наведені дати затемнень, так що астрономічні розрахунки також входили у сферу діяльності цих ранніх обчислювачів. Стародавні єгиптяни також були діяльними обчислювачами. Вони побудували таблиці, які дозволяють подати складні дроби у вигляді суми простих дробів з одиничним чисельником та винайшли метод хибного положення для розв'язування нелінійних алгебраїчних рівнянь.

Переходячи до грецьких математиків, слід відмітити Архімеда, який біля 220 р. до н.е. наближував число π, визначивши його як число, менше за $3\frac{1}{7}$, але більше, ніж $3\frac{10}{71}$. Герон Старший біля 100 р. до н.е. використав ітеративний метод $\sqrt{a} \approx \frac{1}{2}\left(x_n + \frac{a}{x_n} \right)$, який звичайно приписують Ньютону, а школа Піфагора розглядала сумування рядів (1+2+3+…). Діофант (III ст. до н.е.), крім своєї відомої роботи про невизначені рівняння, створив також метод чисельного розв'язування квадратних рівнянь.

Індійці були творцями наших сучасних арифметичних позначок, які звичайно називають арабськими, та розробили метод перевірки правильності арифметичних обчислень, відомий зараз під назвою «перевірки за мо-

дулем 9». Арабському математику Мухаммеду ібн-Муса аль-Хорезмі ми зобов'язані систематизацією обчислювальних методів. Він визначив значення π як 62832/20000 і приймав діяльну участь у складанні математичних таблиць. Абу-л-Вафа (960 р.) розробив метод обчислення таблиць синуса і, як показав М.І. Мєдовой у [118], дав значення $\sin\frac{1}{2}^{\circ}$ з дев'ятьма правильними десятковими знаками; крім того, він використовував функцію tg x та обчислив таблицю її значень.

Переходячи одразу до XVII ст., цікаво відмітити, що перші таблиці логарифмів Непера були створені до введення експоненціальної функції і що його логарифми відрізняються від усіх використовуваних зараз, оскільки Неперів логарифм x дорівнював $10^7\log_e(10^7/x)$. В 1614 р. Непер опублікував свої *Mirifici logarithmorum canonis descriptio*, а у 1619 р. були посмертно опубліковані його *Mirifici logarithmorum canonis constructio*. Дещо пізніше, у 1624 р., Бріггс розробив свою *Arithmetica logarithmica*, яка містить чотирнадцятизначні таблиці логарифмів чисел 1–20000 та 90000–100000, а у 1628 р. Влакк склав чотирнадцятизначні таблиці логарифмів чисел 1–100000. Приблизно в той самий час, у 1620 р., Гюнтер опублікував перші таблиці логарифмів тригонометричних функцій, що заслуговують на довіру.

Починаючи з XVII ст., математики все частіше звертаються до обчислень (за допомогою таблиць, спеціальних приладів тощо), які за самою своєю природою були наближеними. Значний вклад у розробку методів та методології таких обчислень внесли І. Ньютон, В. Лейбніц, Л. Ейлер, К. Гаусс та інші видатні математики.

Таким чином, *чисельні методи* (методи наближених обчислень, обчислювальна математика, прикладна математика) мають багатовікову історію, проте як окремий розділ математики вони існують не такий вже довгий час. В 1906 р. академік О.М. Крилов прочитав курс лекцій про наближені обчислення слухачам вільного математичного факультету при гімна-

зії К. Мая, яким керував професор Н.М. Гюнтер. Того ж самого року курс було відлітографовано, а у 1911 р. видано типографським способом. Книга О.М. Крилова «Лекції про наближені обчислення» [97] стала першим у світі систематизованим курсом наближених обчислень. Як відзначає відомій німецький математик Л. Коллатц, «без перебільшення можна сказати, що «Лекції про наближені обчислення» академіка Крилова є витоком або початком нового розділу сучасної математики, який називають *обчислювальною математикою*» [90, с. 5].

Дійсно, у цій книзі, поряд з викладенням та глибоким аналізом «методів доведення до числових результатів розв'язків математичних задач» ставиться принципове питання про те, що задачі у кінці кінців немає потреби розв'язувати точно, що ступінь наближеності їх розв'язку потрібно узгоджувати з похибками вихідної інформації тощо. На думку автора, створений ним курс має своєю ціллю «показати дійсно застосовні практичні прийоми і способи обчислень… Головна проблема при цьому була у тому, щоб показати, *як та коли використовувати той чи інший прийом, а не у теоретичній строгості обґрунтування самого прийому чи способу*» [97, с. 7].

Курс О.М. Крилова виявився настільки вдалим, що практично без змін перевидавався майже півстоліття: останнє, шосте його видання вийшло у 1954 р. як навчальний посібник для вузів, і саме воно закінчило епоху методів, орієнтованих на ручні та напівручні розрахунки. Розвиток обчислювальної математики (та відповідних навчальних курсів) з початку 50-х рр. вівся у напрямку оптимізації існуючих методів для обчислень на електронно-обчислювальних машинах (ЕОМ) та створення нових методів, орієнтованих на обчислення саме на ЕОМ. Це зумовило появу нової назви методів наближених обчислень – тепер вони стали *методами машинних обчислень*. Саме у 1954 р. створюється перша спеціалізована мова для програмування обчислювальних та науково-технічних задач – Fortran [15], яка

і досі залишається найбільш широковживаною мовою серед науковців-практиків. Наприкінці 70-х рр. розвиток обчислювальної техніки призвів до появи нового класу чисельних методів, орієнтованих на паралельні [29] та об'єктні [143] обчислення, і можна впевнено сказати, що ці технології є визначальними для усього подальшого розвитку чисельних методів.

Таким чином, у своєму розвитку чисельні методи пройшли такі три основні етапи:

I. До початку XX ст. – етап накопичення досвіду наближених обчислень.
II. Перша половина XX ст. – безмашинний, або докомп'ютерний етап.
III. Друга половина XX ст. – машинний, або комп'ютерний етап.

На першому етапі відбувалося формування основних навичок практичних вимірювань та обчислень, які призвели до виникнення та розвитку математики як науки. Другий етап характеризувався розвитком чисельних методів як галузі математичної науки, орієнтованої на виконання науково-технічних розрахунків при якомога меншій кількості дій за допомогою механічних пристроїв. Останній, третій етап, характеризується розвитком обчислювально ємних методів, орієнтованих на виконання розрахунків за допомогою електронно-обчислювальних машин різної архітектури.

Виданий у 1988 р. математичний енциклопедичний словник означав обчислювальну математику як «розділ математики, що включає коло питань, пов'язаних з використанням ЕОМ» [113]. Проте, як зазначають І.П. Гаврилюк та В.Л. Макаров [38], зміст цього терміну не можна вважати усталеним: поступове взаємопроникнення математичних методів, орієнтованих на машинні обчислення, та методів програмування, орієнтованих на математичні обчислення, призвело до того, що чисельні методи у їх сучасному вигляді не можна вважати ані розділом математики, ані розділом інформатики – це *галузь знань, що знаходиться на стику математики та інформатики*, природно їх інтегруючи.

Розвиток чисельних методів вимагав постійного оновлення відповід-

них курсів, що читалися у вищих навчальних закладах з цього предмету. Як складова частина вузівської математичної освіти, чисельні методи впливали і на зміст шкільної математичної освіти, на підвищення рівня обчислювальної культури школярів. Велику роботу у цьому напрямку проводили В.М. Брадіс [20], Н.І. Сирньов [191], П.В. Стратилатов [138] та інші. У 60-70-х рр., під час реформи шкільної математичної освіти, створювалися факультативні курси для старшокласників з елементів теорії ймовірностей, вищої математики та чисельних методів, серед яких особливо слід виділити курси А.Б. Бакушинського та В.К. Власова [11], С.П. Пулькіна [149, 150], М.І. Жалдака, Б.С. Ковбасенко та Ю.С. Рамського [71], В.М. Монахова [122], М.М. Рассудовської [155], М.І. Жалдака та Ю.С. Рамського [74].

З метою виявлення основних закономірностей та внутрішньої логіки курсів чисельних методів прослідкуємо, як проходила їх еволюція, для чого звернемося до змісту відповідних підручників, і почнемо, звичайно, з «Лекцій про наближені обчислення» О.М. Крилова [97], які слугуватимуть нам надалі порівняльним орієнтиром.

У вступі автор на емпіричному рівні вводить поняття точності обчислень, відносної й абсолютної похибки та на прикладах різних типів розрахунків (обчислення суми, добутку, логарифмування тощо) вводить загальні правила наближених обчислень. Коротко зупиняючись на знаходженні коренів рівнянь, автор переходить до обговорення необхідності застосування визначених інтегралів для знаходження площ і об’ємів та методів їх обчислення.

Методам наближеного обчислення визначених інтегралів (у тому числі й кратних) автор приділяє особливу увагу та присвячує їм дві глави. У першій главі розглядаються методи прямокутників, трапецій, формули Сімпсона, Котеса, Чебишова, Гаусса та інтерполяційна формула Лагранжа, у другій – механічні прилади для обчислення визначених інтегралів (пла-

німетри, інтегратори та інтеграфи).

Наступна глава підручника присвячена розкладу функцій у тригонометричні ряди, зокрема – прискоренню швидкості збіжності ряду. Розглядаються випадки аналізу графічно заданих функцій та спеціальні механічні прилади для розв'язання цієї задачі – гармонічні аналізатори.

Методи інтерполювання викладені у главі «Формули, що виражають зв'язок між сумою та інтегралом через різниці та похідні». Вводиться формула Ейлера та приклади її застосування, формули для поданння інтегралів та похідних через різниці, інтерполяція за Гауссом та її застосування до обчислення інтегралів та похідних.

У передостанній главі викладено методи наближеного інтегрування диференціальних рівнянь: Пікара, Ейлера, Рунге, Адамса, Штернера та інші. Ця глава є найбільшою як за обсягом, так і за концентрацією викладеного матеріалу; численні приклади (рух сферичного маятника, поїзда, обчислення траєкторії снаряду, форми краплі та багато інших) допомагають краще усвідомити викладений матеріал. Останні параграфи глави присвячено обчисленню фундаментальних функцій в задачах математичної фізики (згодом, у 1913 р., ці декілька сторінок стануть основою знову ж таки першого систематичного курсу з методів математичної фізики). Заключна глава, що з'явилася у виданні 1934 р., присвячена методу найменших квадратів.

Таким чином, у цьому підручнику визначена більша частина змісту сучасних курсів чисельних методів. Незважаючи на те, що частина матеріалу є застарілою, його й досі можна використовувати як один з кращих курсів для початкового ознайомлення з чисельними методами. Слід відзначити, що побудова курсу та матеріал, обраний О.М. Криловим, був для свого часу нетрадиційним. Переважна більшість книжок, присвячених обчисленням, були на дуже низькому математичному рівні, що зумовлено тогочасним ставленням до обчислювальної математики (або, як її тоді ще на-

зивали, «практичної математики») як математики для малоосвічених (!) людей. Зараз це може викликати лише посмішку, проте ще наприкінці XIX сторіччя деякі відомі математики (див., наприклад, [134]) вважали наближені обчислення «брудними», на відміну від «чистої» математики.

Праця О.М. Крилова викликала до життя декілька невеликих за обсягом матеріалу, але важливих праць, присвячених різним аспектам наближених обчислень. Популяризація наближених обчислень мала у той час дуже високу актуальність; щоб пересвідчитись у цьому, наведемо фрагмент з передмови Ю.А. Шиманського до книги «Принципи числових розрахунків. Теорія та практика раціонального ведення числових розрахунків» [221], що вийшла у 1909 р: «Відсутність систематизованого збірника правил раціонального розрахунку і повне ігнорування, а інколи й перекручене тлумачення цього питання в середніх та вищих навчальних закладах призводять до того, що одні з майже забобонним страхом відносяться до можливості будь-якого скорочення під час розрахунків та ведуть їх «про всяк випадок» із сумлінням, яке вимагає кращого застосування, інші, більшменш зрозумівши сутність розрахунків, а інколи й виробивши собі власні прийоми скорочених обчислень, не застосовують їх на практиці через побоювання бути запідозреними у недбалості розрахунку особами, до яких він попаде у руки». Сам О.М. Крилов неодноразово наводив приклади, коли йому потрібно було розглядати подані у Морський технічний комітет проекти, у яких 9/10, а інколи навіть 34/35 обчислювальної роботи було затрачено на виписування та обчислення зайвих цифр. Виконана на високому рівні робота Шиманських [221] розглядає теорію похибок майже у сучасному її трактуванні. Разом з працею Н.В. Оглобліна «Визначення ступеня точності при логарифмічних обчисленнях» [129] (першою працею такого плану, виданою на Україні), вона стала вдалим доповненням до відповідної глави «Лекцій про наближені обчислення».

У 20-30-х рр. виходять чимало підручників з математики, орієнтова-

них на слухачів різного рівня підготовки, від школярів до інженерів. Це посібник для фізико-математичних факультетів С. Придатко «Практичні обчислення» (1924 р., [148]), посібник Я.С. Безіковича «Математика» (1926 р., [14]), перекладний підручник А. Декіна «Прикладна математика для шкіл, курсів та самоосвіти» (1926 р., [61]), збірники задач (1930 р., [206]) і посібник (1931 р., [205]) для шкіл та ФЗУ І. Файнермана, «Технічна математика» І.Г. Єрофєєва та С.Р. Ляшука (1931 р., [63]), «Прикладна математика» П.К. Шмулевича (1931 р., [226]), посібники та збірники задач з прикладної математики для студентів, аспірантів та викладачів втузів (1932 р., [79]) Б.І. Ізвєкова, «Математика для інженерів» Г.М. Фіхтенгольца (1933 р., [209]), «Чисельні методи математичного аналізу» Д.Г. Скарборо (1934 р., [182]) та інші.

Широко досліджувалися питання виробничої спрямованості навчання в школі (див., наприклад, [199]), що зумовило суттєві зміні у курсах чисельних методів. Так, С. Придатко [148] у своєму курсі обмежується діями над наближеними числами (множення, ділення та добування кореня із заданою точністю), аналізом похибок при обчисленнях за таблицями логарифмів та прийомами розрахунків за допомогою рахівниці та логарифмічної лінійки. Я.С. Безікович, адресуючи свій посібник [14] співробітникам Головної Палати міри та вагів, у стислій формі викладає відомості про скорочені обчислення, формули, таблиці, функції та їх графіки, прямі та обернені залежності, конічні поверхні, поверхні другого порядку, логарифмічну лінійку, диференціальне числення, визначення похибок, наближене розв’язування рівнянь, знаходження мінімуму та максимуму функції, метод найменших квадратів, інтегрування функцій, визначений інтеграл, вимірювання площин та об’ємів. Значення цього посібника у тому, що, поряд з об’єднанням основ вищої математики та чисельних методів (як це пізніше було зроблено у [11]), Я.С. Безікович заклав підвалини такого розділу чисельних методів, як безумовна оптимізація функцій.

Бурхливий розвиток математичного апарату чисельних методів у 20-50-х рр. в працях Д.В. Агеєва, Н.І. Ахієзера, Г.М. Баженова, Я.С. Безіковича, М.Ш. Бірмана, В.П. Вельміна, Д.А. Вентцель, О.С. Вентцель, М.К. Гавуріна, Ф.Р. Гантмахера, І.М. Гельфанда, Я.Л. Геронімуса, А.М. Данілевського, Д.М. Загадського, В.К. Іванова, Л.В. Канторовича, Ф.М. Когана, М.Г. Крейна, В.І. Крилова, А.Г. Куроша, Н.Н. Лузіна, Л.А. Люстерніка, Б.К. Млодзеєвського, І.П. Натансона, С.М. Нікольського, К.А. Семендяєва, В.І. Смірнова, Д.К. Фаддєєва, В.Н. Фаддєєвої, Г.М. Фіхтенгольца, В.А. Фока, А.Я. Хінчіна, Г.М. Шапіро, Ю.А. Шрейдера, М.Р. Шура-Бури та інших знаходив, однак, слабке відображення у навчальних курсах. Більшість підручників, що виходили в ті роки (Л.В. Канторовича, В.І. Крилова, Я.С. Безіковича та інш.) суттєво не розширювали набір методів, викладених у підручнику О.М. Крилова, хоча деякі з них і відрізнялись більш високим математичним рівнем, ніж згадувані вище [14] та [148]. Це зумовлювалося всебічною орієнтацією на потреби виробництва, а, відповідно, – відсутністю необхідності суттєвої модифікації чи заміни курсу О.М. Крилова, який давав те, що, на думку автора курсу, «…буває потрібно техніку та інженеру» ([97], с. 8), основними інструментами якого є рахівниця, логарифмічна лінійка та арифмометр [95].

Перші підручники з чисельних методів, орієнтованих на машинні обчислення за допомогою ЕОМ, з’явилися на початку 50-х рр. Починаючи з 1950 р., виходять роботи вітчизняних та зарубіжних авторів, присвячених різним аспектам застосування ЕОМ до розв’язання математичних проблем, а у 1956 р. у видавництві іноземної літератури виходить підручник відомого математика А.С. Хаусхолдера «Основи чисельного аналізу» [215], причому майже одразу після його видання мовою оригіналу. Розглянемо структуру цього підручника та основні відмінності її від видань докомп’ютерного періоду. Книга складається з таких глав:

I. **Мистецтво обчислень**: *Похибки та прорахунки. Утворення похибок. Перехідні похибки. Значущі цифрі. Аналіз повної похибки. Статистичні оцінки похибок.*

II. **Матриці та лінійні рівняння**: *Ітераційні методи. Прямі методи. Порівняння різних методів.*

III. **Нелінійні рівняння та системи рівнянь**: *Метод Лобачевского. Метод Бернуллі. Функціональні ітерації. Системи рівнянь. Комплексні корені та методи обчислення множників.*

IV. **Власні значення та власні вектори матриці**: *Ітераційні методи. Прямі методи.*

V. **Інтерполяція**: *Інтерполяція многочленами. Тригонометричне та показникове інтерполювання.*

VI. **Більш загальні методи апроксимації**: *Кінцеві лінійні методи. Чебишовські розклади.*

VII. **Чисельне інтегрування та диференціювання**: *Проблема квадратур у цілому. Чисельне диференціювання. Операторні методи.*

VIII. **Метод Монте-Карло**: *Чисельне інтегрування. Випадкові послідовності.*

Порівняно з курсом О.М. Крилова, відбулися такі зміни:

1. Розв’язання систем лінійних рівнянь ґрунтується на алгебрі матриць, тому вводяться операції знаходження оберненої матриці, множення матриць, знаходження алгебраїчного доповнення тощо. Показано зв’язок між прямими та ітераційними методами розв’язування систем лінійних алгебраїчних рівнянь.
2. Метод Лобачевского поширено на системи нелінійних рівнянь, умови збіжності яких викладено з урахуванням їх можливої машинної реалізації. Введено поняття функціональних ітерацій.

3. Вперше систематично викладені основні результати роботи вчених у 30-50 рр. над проблемою власних значень та власних векторів матриці. Особлива увага звертається на ітераційні методи розв'язування, зокрема – розроблені автором. Досліджуються можливості застосування різних методів до знаходження власних значень та власних векторів різнотипних матриць великого розміру.
4. На основі єдиного підходу розглядаються поліноміальні, експоненціальні та тригонометричні наближення. Серед інтерполяційних многочленів розглядаються передусім зручні для машинних розрахунків (а, скажімо, про метод Лагранжа лише коротко згадується).
5. Чисельне диференціювання табульованих функцій розглядається як некоректна задача.
6. Вперше систематично викладено метод Монте-Карло.

На останньому пункті зупинимося більш детально. Підручник А.С. Хаусхолдера містив здебільшого матеріал, який вже був відомий раніше радянським математикам, проте основний акцент у ньому робився саме на методах, призначених для розв'язання обчислювально важких задач. Автор концентрується на загальних підходах до розв'язування чисельних проблем за допомогою ЕОМ, хоча і використовує їх не систематично. Проте основною заслугою цієї книги все ж таки можна вважати *введення методу Монте-Карло у практику обчислень* як першого чисельного методу, застосування якого без допомоги ЕОМ практично неможливо. На відміну від інших, глава, присвячена цьому методу, дуже коротка (4 сторінки без бібліографічних зауважень), але вона відкрила собою новий етап у розвитку курсів чисельних методів.

Як зазначає у тому ж 1956 р. Е.Д. Бут, «… в 40-х рр. відбулася революція та відродження чисельного аналізу. Були розвинені нові методи, а проблеми, які раніше не могли бути розв'язані навіть роботою протягом усього життя, зараз розв'язуються за декілька годин» [24].

Підручник А. Анго «Математика для електро- та радіоінженерів» [7], що вийшов у 1957 р., містить величезний матеріал з вищої математики та чисельних методів. Особливо цікавим в ньому є розділ, присвячений аналітичному операційному методу, програмна реалізація якого у [141] стала одним із зразків гармонійного поєднання аналітичних (символьних) та чисельних методів у сучасних технологіях програмування.

Починаючи з 1957 р., чисельні методи, спочатку у вигляді спецкурсів, а потім і як рівноправний предмет, починають викладатися у педагогічних вузах, тому поява у 1958 р. посібника П.Ф. Фільчакова [208], призначеного для фізико-математичних факультетів педагогічних інститутів УРСР, була не випадковою. Зміст цього курсу майже повністю збігається з курсом О.М. Крилова, що дозволяє віднести його до попереднього етапу розвитку чисельних методів. Значимість його у тому, що це є перший підручник з чисельних методів для педагогічних вузів, до того ж виданий українською мовою. Майже аналогічне видання, «Елементи обчислювальної математики» за редакцією С.Б. Норкіна [230], вийшло російською мовою у 1960 р.

Першим повним вітчизняним курсом чисельних методів з орієнтацією на машинні обчислення стало фундаментальне двотомне видання І.С. Березіна та М.П. Жидкова «Методи обчислень» (1959 р., [16]). Таким чином, всього через три роки після виходу першого перекладного видання був підготований та виданий якісно новий підручник, що не лише не поступався кращим закордонним аналогам, але й перевершував їх.

Автори курсу у передмові відмічають, що вони «… ставили своєю задачею викласти з можливою строгістю … методи чисельного розв’язування … математичних задач. Розвиток обчислювальної техніки за останні роки наклав свій відбиток на обчислювальну математику. Автори намагалися відобразити це у своєму курсі. Але тут зустрілися великі труд-

нощі, викликані двома причинами. З одного боку, потрібно було дати не дуже широкий систематичний виклад найважливіших чисельних методів особам, які не знайомі зі специфікою обчислювальної роботи. З іншого боку, багато напрямків сучасної обчислювальної математики ще не є остаточно усталеними. … Враховуючи широке використання цифрових обчислювальних машин у практиці розрахунків у теперішній час, ми робили основний натиск на чисельні методи розв'язування задач і зовсім мало торкалися аналітичних методів наближеного розв'язування математичних задач».

У першій главі викладено основні правила дій з наближеними числами та правила оцінки їх точності. У главах 2–5 викладено основні способи наближення функцій (інтерполювання, рівномірне та середньоквадратичне наближення функцій) та їх застосування. У главі 3 викладено чисельні методи диференціювання та інтегрування. В главах 6 та 8 описано чисельні методи розв'язування основних задач лінійної алгебри: розв'язування систем лінійних алгебраїчних рівнянь, обернення матриць, обчислення власних значень та власних векторів матриць. У главі 7 викладено способи чисельного розв'язування алгебраїчних рівнянь вищих степенів та трансцендентних рівнянь. Нарешті, глави 9 та 10 присвячено чисельним методам розв'язування звичайних диференціальних рівнянь, диференціальних рівнянь у частинних похідних та інтегральних рівнянь.

Зауважимо, що, хоча обчислення на ЕОМ у цьому курсі передбачалися, але чимала частина машиноорієнтованих чисельних методів (зокрема, метод Монте-Карло) у нього не увійшла. Крім того, обрана авторами строгість викладу базується на застосуванні ними ідей функціонального аналізу, що вимагало досить серйозної математичної підготовки слухачів і орієнтувало цей курс здебільшого на математичні факультети університетів. (Гарний огляд інших підручників такого плану було дано німецьким математиком Л. Коллатцом у його книзі «Функціональний аналіз та обчис-

лювальна математика» [90], що вийшла у 1969 р.) Це зумовило необхідність створення більш компактного та простого курсу, засвоєння якого вимагало б лише знання основ вищої математики. Навчальний посібник для втузів Б.П. Демідовича та І.А. Марона «Основи обчислювальної математики» (1960 р., [56]) цим вимогам цілком задовольняв, що на багато років зробило його чи не найпопулярнішим підручником з чисельних методів. Пізніше вийшли ще дві книги: Б.П. Демідович, І.А. Марон, Е.З. Шувалова «Чисельні методи аналізу» (1963 р., [57]) та Н.В. Копченова, І.А. Марон «Обчислювальна математика у прикладах та задачах» (1972 р., [93]), які є логічним продовженням та доповненням [56].

Серед позитивних рис цього курсу особливо слід виділити важливі для розвитку методики викладання чисельних методів, а саме:

– показано використання апарату ланцюгових дробів для розкладу функцій у ряди;
– систематизовано викладання методів наближеного обчислення значень функцій різних типів;
– приділено велику увагу алгебрі матриць та лінійним векторним просторам;
– показано застосування ортонормальних перетворень до розв’язування систем лінійних рівнянь;
– розглянуто застосування методу Монте-Карло до розв’язування різних класів задач.

Система задач, що входить у склад курсу, разом з авторськими зауваженнями дозволяють скласти уявлення про авторську концепцію його викладання.

Так, курс чисельних методів згідно [93] має структуру, наведену у табл. 1.2.

Таблиця 1.2

Наближене тематичне планування курсу чисельних методів за І.А. Мароном та іншими

Тема	Назва теми	К-сть занять
1	*Правила наближених обчислень та оцінка похибок при обчисленнях*	**5**
2	*Обчислення значень функцій*	**5**
3	*Чисельне розв'язування систем лінійних алгебраїчних рівнянь*	**11**
4	*Чисельне розв'язування систем нелінійних рівнянь*	**4**
5	*Інтерполювання функцій*	**6**
6	*Чисельне диференціювання*	**3**
7	*Наближене обчислення інтегралів*	**7**
8	*Наближене розв'язування звичайних диференціальних рівнянь*	**10**
9	*Крайові задачі для звичайних диференціальних рівнянь*	**6**
10	*Чисельне розв'язування рівнянь у частинних похідних та інтегральних рівнянь*	**10**
	Всього занять:	**67**

Кожна тема курсу складається з визначеної у табл. 1.2 кількості занять, а кожному заняттю відповідає один параграф з [93]. На початку кожного параграфа подаються стислі теоретичні відомості: постановка задачі, робочі формули, обчислювальні схеми, оцінки похибки, порівняння окремих методів з точки зору їх трудомісткості, ступеня точності, який можна досягти, зручності реалізації на ЕОМ тощо. Далі наводиться докладне розв'язування типових прикладів, які ілюструють відповідні алгоритми. Наприкінці параграфа пропонуються задачі для самостійних вправ; майже всі вони мають відповіді.

У другому виданні [56], яке вийшло у 1963 р., автори рекомендують свій курс також для студентів фізико-математичних факультетів педагогічних інститутів, оскільки діючий тоді підручник П.Ф. Фільчакова [208] було видано лише українською мовою, а посібник з обчислювального практикуму І.К. Андронова, А.К. Окунева та Н.І. Сирньова [8], що вийшов обмеженим накладом у 1963 р., не відповідав орієнтації на машинні обчислення.

Приблизно у той самий час, коли створювався курс Б.П. Демидовича та І.А. Марона, у США виходить курс відомого спеціаліста у галузі чисельних методів та програмування Р.В. Хеммінга [216], який він читав студентам Стенфордського університету. Книга має назву «Чисельні методи для наукових робітників та інженерів», що ще більше підкреслює її прикладну спрямованість. Як підкреслює редактор перекладу, в основу книги покладено дві тези: «*мета розрахунків – розуміння, а не числа*» та «*перед тим, як розв'язувати задачу, подумай, що робити з її розв'язком*». Автор особливо зауважує, що книга передбачає знайомство читача з мовою програмування Алгол [203] чи Фортран [161].

Підручник Р.В. Хеммінга складається з 33 глав, по 10-12 сторінок кожна – фактично 33 лекції з чисельних методів, які складають річний лекційний курс. Від інших видань цей підручник відрізняє незвичний, проте дуже логічний поділ матеріалу на чотири основні розділи: «Дискретне числення скінчених різниць» (*Числення різниць. Похибки округлення. Числення сум. Обчислення нескінченних рядів. Рівняння у скінчених різницях. Скінченні ряди Фур'є*), «Наближення многочленами – класичний чисельний аналіз» (*Вступ до многочленних наближень. Інтерполяція многочленами: дані з довільними проміжками. Інтерполяція многочленами: рівновіддалені вузли. Єдиний метод знаходження інтерполяційних формул. Про знаходження залишкового члена формули. Формули для визначених інтегралів.*

Невизначені інтеграли. Вступ до диференціальних рівнянь. Загальна теорія методів прогнозу та корекції. Спеціальні методи інтегрування звичайних диференціальних рівнянь. Метод найменший квадратів: теорія. Метод найменших квадратів: практика. Многочлени Чебишова. Раціональні функції), «Немногочленні наближення» (*Періодичні функції: апроксимація Фур'є. Збіжність рядів Фур'є. Неперіодичні функції: інтеграл Фур'є. Лінійні фільтри: зглажування та диференціювання. Інтеграли та диференціальні рівняння. Експоненціальна апроксимація. Особливості*), «Алгоритми та евристичні методи» (*Знаходження нулів. Системи лінійних алгебраїчних рівнянь. Обернення матриць та власні значення. Деякі приклади моделювання. Випадкові числа та метод Монте-Карло. Мистецтво обчислювати для інженерів та вчених*).

Книга Р.В. Хеммінга містить положення, суттєві для усього подальшого розвитку курсів чисельних методів та методики їх викладання:

1. **Різниця між арифмометром та ЕОМ не у тому, що можна працювати з великими задачами** (як на цьому наголошував за 10 років до того А.С. Хаусхолдер)**, а у тому, що з'являється *зовсім інший підхід* до них.**
2. При навчанні чисельних методів **важливо показати, як можна об'єднати різні частинні результати у рамках *загальних ідей та методів*.**
3. **Матеріал треба подавати у формі, зручній для тих, хто** більше **зацікавлений у *використанні потужних обчислювальних засобів***, ніж у «красі виведення формул».

Завершуючи огляд книги Р.В. Хеммінга, наведемо тези з останньої глави його книги «Мистецтво обчислювати для інженерів та вчених», в якій автор формулює **загальні рекомендації з технології проведення чисельних розрахунків**:

1) Метою чисельного розрахунку практичної задачі є не стільки відповідь у її числовій формі, скільки розуміння того, що вона якісно означає.
2) Приступаючи до обчислень, слід спробувати відповісти на питання: «Що ми збираємося робити з відповіддю?», тобто які нові відомості ми збираємося отримати з обчислень – це допоможе спланувати процес обчислень та скоротити їх обсяг.
3) Наступне питання, на яке треба дати відповідь: «Що нам відомо?», тобто якою інформацією ми володіємо? які вхідні дані? чи включено до них усю відому інформацію? – дослідження вхідних даних може повернути нас до етапу постановки чисельної задачі та навіть перегляду самої проблеми.
4) При плануванні процесу обчислень треба врахувати якомога більше початкових даних, а сам план обчислень повинен містити перевірки як програмування, так і результатів.
5) При програмуванні чисельного алгоритму розв'язування задачі треба оцінити такі обставини:
   – Чи впливатимуть похибки округлення і, якщо так, то наскільки?
   – Чи задовольняє взятий інтервал?
   – Якщо маємо ітераційний процес, то скільки приблизно треба буде ітерацій?
   – Скільки часу віднімуть програмування та налагодження?
   – Як перевірити, чи правильні результати?
   – Скільки треба буде машинного часу?
   – Коли будуть отримані кінцеві результати?
6) Зміни у обчислювальний алгоритм слід вносити обережно, їх слід так само обміркувати, як початковий план, і зрозуміти, чому сталася помилка при плануванні; важливо з'ясувати, чи дає зміна нові відомості про використану модель? чи треба намагатися ще отримати подібні резуль-

тати? чи треба з цього приводу вставити нові перевірки у модель? чи можна що-небудь зрозуміти з самої невдачі чи з нового плану?

Ідеї, викладені Р.В. Хеммінгом, були пізніше розвинені вітчизняними авторами Н.С. Бахваловим, Н.П. Жидковим та Г.М. Кобельковим у навчальному посібнику для студентів вузів «Чисельні методи» (1987 р., [13]).

Таким чином, Р.В. Хеммінг у своєму курсі послідовно проводить ідею *про нероздільність чисельних методів та методів програмування*. Як підкреслює у своїй доповіді на Міжнародному математичному конгресі відомий радянський математик, академік Г.І. Марчук, «взаємозв'язок засобів обчислювальної техніки, методів обчислювальної і прикладної математики, теорії автоматичного програмування та мов стає настільки тісною, що … центр уваги все більш зміщується до питань оптимізації усього обчислювального процесу» [111, с. 328].

Першим вітчизняним підручником, що містив не лише алгоритми чисельних методів, а й їх програмну реалізацію, був двотомний курс Р.С. Гутера та інших «Програмування та обчислювальна математика» (1971 р., [53]), призначений для спеціальності «Прикладна математика» математичних технікумів. У першій частині викладаються основи мови Алгол та мови змістовних позначень для триадресних машин типу М-20, у другій – чисельні методи та їх програмування. Курс розрахований на два роки навчання: 1-ий рік – алгоритмізація та основи програмування, 2-ий рік – чисельні методи та їх програмування. Загальний обсяг курсу складає 90 годин, з них 45 відводиться на чисельні методи, а матеріал розподілений таким чином:

Заняття 1–3. *Обчислення елементарних функцій.*

Заняття 4–10. *Чисельне розв'язування алгебраїчних та трансцендентних рівнянь.*

Заняття 12–19. *Системи рівнянь.*

Заняття 20–27. *Інтерполяція.*

Заняття 28–35. *Чисельне інтегрування.*

Заняття 36–45. *Чисельне розв'язування диференціальних рівнянь.*

Незважаючи на обмежений обсяг матеріалу, недостатній для вузівського курсу чисельних методів, та низьку наочність програм, значимість підручника [53] полягає у *поєднанні викладання чисельного методу та його програмної реалізації* в одному курсі. Крім того, вперше курс чисельних методів було перенесено у середні навчальні заклади; перспективність такого підходу обґрунтовано, зокрема, у докторській дисертації В.М. Монахова «Введення в школу застосувань математики, пов'язаних з використанням ЕОМ» (1973 р., [122]) та кандидатській дисертації М.М. Рассудовської «Проблеми обчислювальної математики на факультативних заняттях в 9-му та 10-му класах середньої школи» (1973 р., [155]).

Необхідність ознайомлення учнів з елементами обчислювальної математики зумовило появу підручників С.П. Пулькіна «Обчислювальна математика: Посібник для вчителів з факультативного курсу» (1972 р., [150]) та «Обчислювальна математика: Посібник для учнів 9–10-х кл. з факультативного курсу» (1974 р., [149]), які є авторською переробкою підручника [151] для студентів заочних відділень фізико-математичних факультетів педагогічних інститутів, М.І. Жалдака, С.Б. Ковбасенко, Ю.С. Рамського «Обчислювальна математика. Спец. курс факультативних занять у 9-х і 10-х кл.» (1973 р., [71]) та М.І. Жалдака, Ю.С. Рамського «Чисельні методи математики: Посібник для самоосвіти вчителів» (1984 р., [74]). Останні два посібники, створені представниками української школи програмування та обчислювальної математики, заслуговують на особливу увагу. На відміну від [149] та [150], вони не дублюють один одного, а дійсно викладають матеріал для учня й для вчителя. Так, у посібнику для учнів [71] розглядаються алгоритми обчислювальних процесів, вперше у посібниках такого

типу виділені у окремий розділ та систематизовані, дається поняття про числа з фіксованою та нефіксованою десятковою точкою, наближені числа, таблиці функцій, методи наближеного розв'язування трансцендентних та алгебраїчних рівнянь і їх систем, чисельне диференціювання та інтегрування, методи інтерполювання. Посібник для вчителів [74] побудовано структурно приблизно так само (за винятком додаткового розділу, присвяченого задачам математичного програмування), проте рівень подання матеріалу обрано значно вищий. Книга орієнтована на сучасного вчителя математики з посиленою фундаментальною та професійною підготовкою, який володіє основними поняттями математичного аналізу, алгебри та геометрії. Автори викладають чисельні методи, широко використовуючи елементи аналізу в метричних просторах з метою виявлення зв'язків між вищою та елементарною математикою, глибокою природою чисельних методів та їх зовнішньою простотою. Обраний підхід подання матеріалу дозволяє використовувати цю книгу не лише як посібник для самоосвіти вчителів, а й як підручник для фізико-математичних факультетів педагогічних вузів.

У 1972 р. українською мовою виходить підручник О.Ф. Калайди та А.Т. Янішевського «Елементи програмування та обчислювальної математики» [82], що наслідує концепцію, викладену у [53]. У 1978 р. виходить підручник для фізико-математичних факультетів педагогічних інститутів Т.П. Іванової та Т.В. Пухової «Обчислювальна математика та програмування» [78], в якому в досить стислій формі викладено архітектуру ЕОМ, мову програмування Алгол-60 та чисельні методи, серед яких особливу увагу приділено методам лінійного програмування. Нарешті, у 1983 р. у видавництві «Радянська школа» вийшов посібник для факультативних занять у 10 класі А.С. Козіна та М.Я. Лященко «Обчислювальна математика» [89], що включає в себе два розділи: «Обчислювальний практикум» (*Елек-*

тронні клавішні обчислювальні машини. Загальні відомості про організацію розв'язування задач на сучасних обчислювальних машинах) та «Обчислювальна математика» (*Чисельні методи розв'язування рівнянь та їх програмування. Розв'язування систем лінійних рівнянь. Інтерполювання. Чисельне інтегрування*). Мовою програмування цієї книги, на відміну від двох попередніх, обрано більш прогресивний Фортран.

Серед виданих до 1991 р. підручників з чисельних методів слід особливо виділити «Обчислювальні методи вищої математики» В.І. Крилова, В.П. Бобкова та П.І. Монастирського (1975 р., [98]), «Методи обчислювальної математики» Г.І. Марчука (1973 р., [111]), «Довідник алгоритмів мовою Алгол. Лінійна алгебра» Дж. Уілкінсона та К. Райнша (1976 р., [203]), «Машинний підхід до розв'язування математичних задач» Ю. Нівельгерта, Дж. Фаррара та Е. Рейнголда (1977 р., [127]), «Машинні методи обчислень» Н.А. Пєтухової, А.Н. Жернака та О.А. Пєтухова (1979 р., [136]), «Машинні методи математичних обчислень» Д. Форсайта, М. Малькольма та К. Моллера (1980 р., [210]), «Чисельні методи» Є.А. Волкова (1982 р., [35]), «Методи обчислень на ЕОМ» В.В. Іванова (1986 р., [76]), «Комп'ютерна алгебра: Символьні та алгебраїчні обчислення» Б. Бухбергера, Ж. Каллє та Е. Калтофена (1986 р., [91]), «Безпомилкові обчислення: Методи та застосування» Р.Т. Грегорі та Є.В. Крішнамурті (1988 р., [44]), «Обчислювальні методи та застосування ЕОМ» В.Т. Малікова та Р.Н. Квєтного (1989 р., [110]), «Обчислювальна математика та програмування» Ю.П. Боглаєва (1990 р., [17]), «Прикладні чисельні методи в фізиці та техніці» Т.Є. Шупа (1990 р., [227]), «Системи та алгоритми алгебраїчних обчислень» Дж. Девенпорта, І. Сіре та Е. Турн'є (1991 р., [60]), у яких поглиблювався підхід на використання та створення чисельних алгоритмів, орієнтованих на обчислення за допомогою ЕОМ.

У підручнику [17], призначеному для студентів технічних вузів, чисельні методи розглядаються разом з мовою програмування Фортран. Основну увагу при цьому автор приділяє використанню деякої стандартної бібліотеки математичних підпрограм, призначених для проведення чисельних розрахунків, а тому наводить не тексти програмних реалізацій розглядуваних чисельних методів, а лише приклади їх використання. Такий підхід обмежує використання цього підручника тими навчальними закладами, які мають описувані автором тип ЕОМ та математичне забезпечення, що є неприйнятним у навчальному процесі. Інші підручники, в яких використовується мова Фортран, – «Чисельні методи для ПЕОМ мовами Бейсік, Фортран та Паскаль» А.Є. Мудрова (1991 р., [126]), «Чисельні методи та програмування на Фортрані для персонального комп'ютера» А.Б. та А.С. Самохіних (1996 р., [164]), «Практичне керівництво з методів обчислень з доданням програм для персональних комп'ютерів» В.І. Ракітіна та В.Є. Пєрвушіна (1998 р., [153]), – є застарілими як змістовно, так і за підходами до викладання чисельних методів. Так, наприклад, у підручнику «Методи обчислень: Практикум на ЕОМ» В.Л. Бурківської та інших (1995 р., [119]), програмну реалізацію чисельних методів винесено у додаток, що утруднює співставлення алгоритму чисельного методу та його програмної реалізації. Незважаючи на задекларований більшістю авторів принцип ієрархічності, згідно з яким у кожній підпрограмі повинні максимально використовуватися інші, тексти підпрограм, як правило, не є універсальними, містять велику кількість помилок, не витримується єдиний стиль програмування і таке інше.

На жаль, вітчизняні підручники з чисельних методів, видані у останні роки ([12, 38, 45, 55, 109, 119] та інші), з одного боку, високим математичним рівнем нагадують взірцеві підручники 60-70-х рр., та з іншого, відображають досягнення обчислювальної математики та програмування 20-

30-річної давнини. Недосконалі спроби побудови математичних бібліотек для підтримку курсу чисельних методів та неврахування сучасних тенденцій розвитку чисельного математичного забезпечення знижують можливу ефективність цього курсу. Навіть у найвдаліших курсах основною тенденцією є *механічне поєднання* засобів обчислювальної математики та мов програмування, що суперечить *органічній єдності* чисельних методів, орієнтованих на ЕОМ, та методів програмування, орієнтованих на ефективні обчислення. Це протиріччя виявляється у всіх курсах, в яких чисельні методи реалізуються на ЕОМ засобами мов програмування, і змушує таких дослідників, як М.І. Жалдак, Ю.С. Рамський, С.А. Раков та інші поставити питання про альтернативні шляхи вивчення цього курсу з використанням спеціалізованих ППЗ, зокрема Derive, GRAN-1, Matlab, Mathematica та інших [68].

Не заперечуючи такого підходу, відмітимо, що програмування прикладних задач вимагає перш за все програмування обчислювальних методів, причому у переважній більшості випадків не спеціалізованими мовами математичних пакетів, а мовами загального призначення C++ чи Pascal, що вимагає вивчення чисельних методів з використанням однієї з цих мов. На наш погляд, вимагає удосконалення не стільки зміст курсу чисельних методів, що складає його математичну частину, скільки зміна технології програмування чисельних методів, а саме: *перехід у викладанні курсу чисельних методів від процедурної методології програмування до об'єктно-орієнтованої*. Обґрунтування застосування об'єктного підходу до викладання курсу чисельних методів наведено у наступних параграфах дослідження.

§ 1.3. Принципи застосування об'єктного підходу до розробки математичного програмного забезпечення

«Наука про те, як замість лише швидкодії машини використовувати чисельні методи та бібліотечні програми, переживає період дитинства і є однією з найважливіших областей дослідження у майбутньому» [216, с. 198]. Ці слова Р.В. Хеммінга, сказані майже 40 років тому, і досі не втрачають своєї актуальності. Тенденція до створення та систематичного використання математичних бібліотек, що виникла на початку 60-х рр., наприкінці 80-х рр. стає домінуючою. Проте розвиток чисельних методів зумовлений сьогодні також розвитком засобів обчислювальної техніки та технології програмування. Так, поява паралельних ЕОМ породила новий клас чисельних методів, орієнтованих на паралельні обчислення, а розвиток об'єктно-орієнтованого програмування – новий напрямок: *Object-oriented numerics* (OON), або *об'єктні обчислення* [2, 237, 267], який зараз є провідним напрямком у чисельних методах [260].

Об'єктно-орієнтоване програмування (ООП), що отримало широке розповсюдження як потужна програмна технологія, є у наш час вагомою альтернативою традиційним процедурним методам програмування [112, 239]. Популярність ООП у чималій мірі визначається концептуальною цілісністю та більш сильною формою структуризації програмного забезпечення (ПЗ), що створюється на його основі [34]. Використання ООП прискорює процес розробки програм, даючи при цьому можливість гнучкої та природної модифікації існуючого ПЗ [147]. Найбільш рельєфно можливості ООП проявляються при створенні досить складних програмних продуктів, до яких, зокрема, відносяться проблемно-орієнтовані бібліотеки [201].

Перш ніж розглядати ООП, доцільніше спочатку розглянути його підґрунтя – об'єктний підхід, який є більш загальною технологією дослі-

дження та пізнання, як це пропонує А.П. Єршов. У свої роботі «Про об'єктно-орієнтовану взаємодію з ЕОМ» [65] він надає об'єктно-орієнтованому програмуванню більш широкий зміст, ніж програмуванню лише з використанням об'єктно-орієнтованих мов. У якості одного з прикладів об'єктно-орієнтованої взаємодії програмуючого користувача з ЕОМ А.П. Єршов посилається на Е-практикум як реальну систему автоматизованого конструювання програм, особливо підкреслюючи при цьому тезу про перспективність та універсальність об'єктно-орієнтованої взаємодії. Отже, розглянемо основні передумови впровадження об'єктного підходу у практику викладання чисельних методів, і почнемо з психологічних передумов.

Основне утруднення, що постає перед розробником складного програмного продукту, полягає в неможливості одночасно утримувати в пам'яті усі необхідні деталі. Г.Р. Міллер та його послідовники стверджують (див., наприклад, [21, 250]), що максимальна кількість об'єктів, з якою здатен одночасно оперувати людський мозок, не перевищує 7±2 (так званий «гаманець Міллера»). Це «магічне число», скоріше за все, пов'язане з обсягом короткострокової пам'яті у людини. Ще одним обмежуючим фактором тут виступає швидкість опрацювання мозком нової інформації: йому потрібно приблизно 5 секунд на сприймання кожного нового об'єкту. Як бачимо, природна здатність людського мозку до роботи із складними системами є низькою.

Проте, як услід за Е. Дейкстрою зазначає Б. Страуструп [259], ще з давніх давен людству відомий простий та ефективний спосіб управління складними системами: «Розділяй та володарюй». Тому при проектуванні складної системи (програми) необхідно складати її з окремих невеликих підсистем (підпрограм) – у цьому випадку ми не виходимо за межі можливостей людини: при розробці будь-якого рівня системи необхідно одночасно утримувати в пам'яті інформацію лише про деякі її частини.

Такий підхід називається *алгоритмічної декомпозицією* і забезпечує психологічне підґрунтя для процедурного програмування, визначаючи головну вимогу до написання підпрограми: «усі дії, що виконуються в підпрограмі, повинні усвідомлюватися *одночасно*», і якщо ця вимога не виконується, підпрограму слід поділити на дрібніші блоки. До того ж у шкільному віці «гаманець Міллера» має менші розміри, тому при вивченні основ алгоритмізації поняття допоміжного алгоритму (процедури) та алгоритмічної декомпозиції, на наш погляд, повинні вводитися одними з перших, а самі процедури не повинні містити більше 4–5 усвідомлюваних дій [19, 50, 69].

Проте число подій, що одночасно може опрацювати людина, *не залежить від обсягу інформації*, що міститься у кожній події, і це дає людині надзвичайно ефективний механізм опрацювання складних повідомлень – *абстрагування*. Не маючи можливості відтворити у всіх деталях складний об’єкт, ми ігноруємо несуттєві для нас деталі і, таким чином, маємо справу з узагальненою, ідеалізованою моделлю об’єкта. І хоча при цьому ми, як і раніше, змушені охоплювати одночасно значну кількість властивостей об’єкту, та завдяки абстракції ми використовуємо узагальнені властивості суттєво більшого семантичного обсягу. Це особливо вірно, коли ми розглядаємо світ з позицій об’єктно-орієнтованої взаємодії, оскільки об’єкти як абстракції реального світу являють собою насичені зв’язні інформаційні одиниці. При цьому ми також обмежені кількістю об’єктів, яку можемо сприйняти у кожний окремий момент, все одно, використовуючи абстрактні поняття, ми отримуємо можливість працювати із складними системами, а, отже, і створювати складні програмні продукти.

Складні системи можна досліджувати, концентруючи основну увагу або на об’єктах, що фігурують у системі, або на процесах, що протікають в ній. Проте доцільніше розглядати систему як впорядковану сукупність об’єктів, які в процесі взаємодії один з одним забезпечують функціонуван-

ня системи як єдиного цілого. Об'єкти, що складають систему, можуть утворювати *ієрархії*. При такому підході основним способом дослідження складної системи є *об'єктна декомпозиція*.

Таким чином, з'являється можливість розширити межі когнітивних можливостей людини, використовуючи методи декомпозиції, виділення абстракцій та створення ієрархій. Саме ці методи покладено в основу *об'єктного підходу*, який утворює концептуальний базис *об'єктно-орієнтованої методології*. (Тут під методологією ми розуміємо сукупність методів, що застосовуються при розробці програм і об'єднаних одним загальним філософським підходом.)

Реалізацією об'єктно-орієнтованої методології дослідження складних систем є *об'єктно-орієнтоване програмування* – методологія програмування, заснована на представленні програми у вигляді сукупності об'єктів, кожен з яких є реалізацією деякого класу, а класи утворюють ієрархію за принципами наслідуваності [190].

Таким чином, **об'єктно-орієнтоване програмування є найбільш природною методологією програмування, яка, враховуючи особливості психічних процесів, дає можливість створювати чітко структуровані та осяжні складні програмні продукти**.

Об'єктний підхід відомий ще з давніх часів. Так, давнім грекам належить ідея про те, що світ можна розглядати як у термінах об'єктів, так і подій. А у XVII ст. Декарт підкреслював, що люди зазвичай мають об'єктно-орієнтований погляд на світ. У 60–70-х р.р. XX ст. ця думка була розвинена в одній з течій когнітивної філософії – об'єктивістській епістемології (Е. Ранд), а на початку 80-х р.р. М. Мінскі запропонував модель людського мислення, у якій розум людини розглядається як спільнота агентів, що по-різному мислять. На його думку, лише спільні дії таких агентів приводять людину до осмисленої поведінки.

Основні ідеї об'єктного підходу (абстрагування, типізація, ієрархія

тощо) у тому чи іншому вигляді були присутні у практиці програмування, починаючи з перших мов високого рівня [245, 246]. Одну з можливих класифікацій мов високого рівня наведено у табл. 1.3.

Таблиця 1.3

Класифікація мов програмування високого рівня

(за П. Вегнером, [26, с. 44])

Генерація	*Основні мови*	*Особливості, нові конструкції*
Перша (1954–1958)	Fortran I	Математичні формули
	Algol–58	Математичні формули
	Flowmatic	Математичні формули
	IPL V	Математичні формули
Друга (1959–1961)	Fortran II	Підпрограми, роздільна компіляція
	Algol–60	Блочна структура, типи даних
	Cobol	Опис даних, робота з файлами
	Lisp	Опрацювання списків, вказівників, збирання сміття
Третя (1962–1970)	PL/1	Fortran+Algol+Cobol
	Algol–68	Більш строгий спадкоємець Algol–60
	Pascal	Більш простий спадкоємець Algol–60
	Simula	Класи, абстрактні дані

У кожній наступній генерації змінювалися підтримувані мовами механізми абстракції. Мови першої генерації орієнтувалися виключно на науково-інженерне застосування, і словник цієї предметної галузі був майже виключно математичним. Такі мови, як Fortran I, були створені для спрощення програмування математичних формул, щоб звільнити програміста від труднощів асемблера та машинного коду. Перша генерація мов високого рівня була кроком, що наближує програмування до предметної галузі та віддаляє його від конкретної машини.

Характерною тенденцією мов першої та початку другої генерації став розвиток алгоритмічних абстракцій. У цей час потужність комп'ютерів швидко росла, що дозволяло розширити межі їх застосування. Можна відмітити, що для таких мов, як Fortan та Cobol, основним будівельним блоком є підпрограма. Програми, написані такими мовами, мають відносно просту структуру, що складається лише з глобальних даних та підпрограм. У процесі розробки можна логічно упорядкувати різнотипні дані, проте механізми цих мов практично не підтримують такого упорядкування. Помилка у будь-якій частині програми може мати глобальні наслідки, тому що область даних відкрито усім підпрограмам. В процесі розробки програм цими мовами вже через короткий час виникає плутанина, викликана великою кількістю перехресних зв'язків між підпрограмами, заплутаними схемами управління, що знижує надійність та ясність програми.

Починаючи з середини 60-х рр. поступово усвідомлюється роль підпрограм як програмної (процедурної) абстракції. Погляд на підпрограми як механізм абстрагування дозволив розробити мови, що підтримували різні механізми передавання параметрів. Були закладені основи структурного програмування, що відобразилося у мовній підтримці механізмів вкладеності підпрограм та науковому дослідженні структур управління та областей видимості, виникли методи структурного проектування, які стимулювали розробляти програми, використовуючи підпрограми як готові будівельні блоки.

Наприкінці 60-х рр. з появою транзисторів, а потім інтегральних схем, вартість комп'ютерів різко знизилася, а їх потужність виросла майже експонентно. З'явилася можливість розв'язувати все більш складні задачі, проте це вимагало вміння обробляти найрізноманітніші типи даних. Такі мови, яка Algol–68, а далі Pascal стали підтримувати абстракцію даних у явному вигляді. Виразом потреби у незалежній розробці окремих частин задачі став окремо компільований модуль, який спочатку був просто

більш-менш випадковим набором даних та підпрограм. У такі модулі збиралися підпрограми, які, як уявлялося, скоріше за все будуть змінюватися сумісно, і мало хто розглядав їх як нову техніку абстракції. У більшості мов третьої генерації, хоча і підтримувалося модульне програмування, проте не вводилося жодних правил, що регламентували б узгодження інтерфейсів модулів. Програміст, наприклад, написавши підпрограму в одному з модулів, міг очікувати, що її будуть викликати із трьома параметрами: дійсним числом, десятиелементним масивом та логічною змінною. Але у якомусь іншому модулі ця підпрограма могла помилково викликатися з фактичними параметрами у вигляді: ціле число, п’ятиелементний масив, дійсне число. На жаль, оскільки більшість мов надавали у кращому випадку рудиментарну підтримку абстрактних даних та типів, такі помилки виявлялися лише при виконанні програм.

Абстрагування, що досягається за допомогою процедур, добре підходить для опису абстрактних дій, проте не годиться для опису абстрактних об’єктів. Усвідомлення цього дало два важливих напрямки, що розвивалися у 70-х рр. По-перше, виникають методи проектування на основі потоків даних, які вносять упорядкування в абстракцію даних у мовах, орієнтованих на алгоритми. По-друге, з’являється теорія типів, яка втілюється у таких мовах, яка Pascal. У 70-ті рр. було створено більше двох тисяч різних мов та їх діалектів, та лише деякі з них збереглися до нашого часу. Проте багато принципів, розвинених у попередніх мовах програмування, знайшли своє адекватне відображення у нових. Так, наприклад, мова Smalltalk є спадкоємцем мови Simula, Ada – спадкоємець Algol–68 та Pascal з елементами Simula, Alphard та Clu, Clos об’єднав Lisp, Loops та Flavors, C++ виник від поєднання C та Simula, Eiffel – від Simula та Ada [54]. Ці мови отримали назву об’єктних та об’єктно-орієнтованих, і саме цей клас мов являтиме для нас подальший інтерес.

Таким чином, *класифікацію генерацій мов програмування*, подану у

табл. 1.3, можна переформулювати *з позицій підтримуваних ними абстракцій*: ***математичні, алгоритмічні, орієнтовані на дані та об'єктно-орієнтовані***.

У свою чергу, об'єктно-орієнтовані мови також можна класифікувати за різними ознаками. Дж. Саундерс [257] поділяє їх на 7 таких категорій:

1. *Actor-подібні* – мови, що підтримують механізм делегування.
2. *Паралельні* – мови, що реалізують паралелізм.
3. *Розподілені* – мови, націлені на опрацювання розподілених об'єктів.
4. *Фреймові* – мови, що реалізують теорію фреймів.
5. *Гібридні* – об'єктно-орієнтовані надбудови над звичайними мовами.
6. *Smalltalk-подібні* – мова Smalltalk та її діалекти.
7. *Ідеологічні* – мови, орієнтовані на сферу застосування.
8. *Інші* – мови, які не відносяться до жодної з категорій.

У відповідності до введеного нами визначення не всі мови програмування є об'єктно-орієнтованими. Автор об'єктно-орієнтованої мови програмування C++ Б. Страуструп у [259] зазначає, що «… якщо термін *об'єктно-орієнтована мова* взагалі щось означає, то він повинен означати мову, що має засоби гарної підтримки об'єктно-орієнтованого стилю програмування… Забезпечення такого стилю у першу чергу означає, що у мові зручно використовувати такий стиль. Якщо написання програм у стилі ООП вимагає спеціальних зусиль чи воно неможливо зовсім, то ця мова не відповідає вимогам ООП». Теоретично можлива імітація об'єктно-орієнтованого програмування навіть мовою асемблера [81], проте це надзвичайно важко. За Вегнером, мова програмування є об'єктно-орієнтованою тоді і лише тоді, коли виконуються такі умови:

– підтримуються об'єкти, тобто абстракції даних, що мають інтерфейс у

вигляді іменованих операцій та власні дані, з обмеженням доступу до них;

– об’єкти відносяться до відповідних типів (класів);
– типи (класи) можуть наслідувати атрибути супертипів (суперкласів).

Таким чином, можна дати два такі означення:

Об’єктними називають мови, які підтримують абстракцію даних та класи.

Об’єктно-орієнтованими називають ті об’єктні мови, які підтримують наслідування та поліморфізм.

Побудову спеціалізованих математичних бібліотек, зокрема – бібліотеки класів для підтримки курсу чисельних методів, починають з етапу проектування.

Об’єктно-орієнтоване проектування – це методологія проектування, що поєднує у собі процес об’єктної декомпозиції та прийоми подання логічної та фізичної, а також статичної та динамічної моделей проектованої системи [249].

Саме об’єктно-орієнтована декомпозиція відрізняє об’єктно-орієнтоване проектування від структурного; у першому випадку логічна структура системи відображається абстракціями у вигляді класів і об’єктів, у другому – алгоритмами [99, 105, 188]. Об’єктно-орієнтований аналіз спрямований на створення моделей реальної дійсності на основі об’єктно-орієнтованого світогляду.

Об’єктно-орієнтований аналіз – це методологія, при якій вимоги до системи сприймаються з точки зору класів та об’єктів, виявлених у предметній галузі [195, 197].

Між всіма цими визначеннями існує тісний взаємозв’язок: на результатах об’єктно-орієнтованого аналізу формуються моделі, на яких ґрунтується об’єктно-орієнтоване проектування; у свою чергу, об’єктно-орієнтоване проектування створює фундамент для фінальної реалізації си-

стеми з використанням методології ООП.

У літературі з програмування існують різні підходи до розв'язання питання про класифікацію стилів програмування. Ми дотримуємося класифікації Д. Боброва та М. Стефіка [258], що виявили п'ять основних різновидів стилів програмування, які перераховані нижче разом з властивими їм видами абстракції:

– процедурно-орієнтований	алгоритми
– об'єктно-орієнтований	класи та об'єкти
– логіко-орієнтований	цілі, часто виражені у термінах числення предикатів
– орієнтований на правила	правила «якщо–то»
– орієнтований на обмеження	інваріантні співвідношення

Кожний стиль програмування має свою концептуальну основу; для об'єктно-орієнтованого стилю цією основою є об'єктний підхід, основні елементи якого (принципи об'єктного підходу) були сформульовані Г. Бучем у [25]:

Абстрагування – виділення таких суттєвих характеристик об'єкту, які відрізняють його від усіх інших об'єктів і, таким чином, чітко визначають особливості даного об'єкту з точки зору його подальшого розгляду.

Абстрагування концентрує увагу на зовнішніх особливостях об'єкта та дозволяє відділити найсуттєвіші особливості його поведінки від несуттєвих. Такий поділ називається *бар'єром абстракції*, який будується на принципі мінімізації зв'язків, коли інтерфейс об'єкту містить лише суттєві аспекти поведінки і нічого більше. **Вибір правильного набору абстракцій для заданої предметної галузі є основною задачею об'єктно-орієнтованого проектування**. Всі абстракції володіють як статичними, так й динамічними властивостями. Наприклад, матриця як об'єкт потребує певного обсягу пам'яті та має певний зміст. Ці атрибути являють собою статичні властивості. Конкретні ж значення кожної з перелічених власти-

востей динамічні та змінюються у процесі використання об’єкта: матрицю можна збільшити чи зменшити, змінити її елементи. У процедурному стилі програмування дії, що змінюють динамічні характеристики об’єктів, складають сутність програми. Стиль програмування, орієнтований на правила, характеризується тим, що за певних умов активізуються певні правила, які, в свою чергу, активізують інші правила, і т.д. Об’єктно-орієнтований стиль програмування пов’язаний із впливом на об’єкти (з передаванням об’єктам повідомлень). Так, операція над об’єктом породжує деяку реакцію цього об’єкту. Операції, які можна виконати по відношенню до даного об’єкту, та реакція об’єкта на зовнішні впливи визначають поведінку цього об’єкту.

Обмеження доступу (інкапсуляція) – процес захисту окремих елементів об’єкту, що не зачіпає суттєвих характеристик об’єкту як цілого.

Абстрагування та інкапсуляція є взаємодоповнюючими операціями: абстрагування фокусує увагу на зовнішніх особливостях об’єкта, а інкапсуляція не дозволяє об’єктам-користувачам розрізняти внутрішню будову об’єкта, виконуючи роль обмеження доступу, або *захисту інформації*. Найчастіше інкапсуляція реалізується за допомогою приховування інформації, тобто маскуванням усіх внутрішніх деталей, що не впливають на зовнішню поведінку. Практично це означає наявність у класі двох частин: інтерфейсу та реалізації. *Інтерфейс* відображує зовнішню поведінку об’єкта, описуючи абстракцію поведінки усіх об’єктів даного класу. Внутрішня *реалізація* описує подання цієї абстракції та механізми реального досягнення бажаної поведінки об’єкта. Поділ на інтерфейс та реалізацію є природним: у інтерфейсній частині зібрано усе, що стосується взаємодії даного об’єкту з будь-якими іншими об’єктами, а реалізація приховує від інших об’єктів усі деталі, що не мають відношення до процесу взаємодії об’єктів.

Модульність – властивість системи, пов’язана з можливістю її декомпозиції на ряд тісно пов’язаних модулів.

У деяких мовах програмування (зокрема, у Smalltalk) модулів нема, і

класи складають єдину фізичну основу декомпозиції. У інших мовах, включаючи Object Pascal, C++, Ada, CLOS, модуль – це самостійна мовна конструкція. В цих мовах класи та об'єкти утворюють логічну структуру системи, вони розміщуються у *модулі, які утворюють фізичну структуру системи*. У більшості мов, що підтримують принцип модульності як самостійну концепцію, інтерфейс модуля відділено від його реалізації. Правильний поділ програми на модулі є майже такою ж складною задачею, як вибір правильного набору абстракцій. Для більшості програм (окрім найпростіших) найкращим рішенням буде таке:

1) особливості системи, що можуть змінюватися, слід сховати у окремих модулях;
2) у якості міжмодульних слід використовувати лише ті компоненти, ймовірність зміни яких мала;
3) усі структури даних повинні бути інкапсульовані в модулі;
4) доступ до структур даних модуля можливий для всіх процедур цього модуля і не можливий для всіх інших;
5) доступ до даних з модуля слід здійснювати лише через процедури цього модуля.

Іншими словами, слід намагатися побудувати модулі так, щоб об'єднати логічно пов'язані абстракції та мінімізувати взаємозв'язки між модулями. Виходячи з цього, ми можемо уточнити означення модульності: ***модульність*** *– властивість системи, яка була розкладена на внутрішньо зв'язні, але слабо пов'язані між собою модулі*. Таким чином, принципи абстрагування, інкапсуляції та модульності є взаємодоповнюючими. Об'єкт логічно визначає межі деякої абстракції, а інкапсуляція та модульність фізично їх закріплюють.

Ієрархія – рангована чи впорядкована система абстракцій.

Кількість абстракцій в системі завжди, окрім найпростіших випадків, перевищує розумові можливості людини. Інкапсуляція у деякій мірі дозво-

ляє усунути цю перепону, усунувши з поля зору внутрішній зміст абстракцій. Модульність до того ж спрощує задачу, об'єднуючи логічно пов'язані абстракції в групи. Проте утворення з абстракцій ієрархічної структури дозволяє значно спростити розуміння складних задач. Ієрархія – це впорядкування абстракцій, розташування їх за рівнями. Основними видами ієрархічних структур є структура класів та структура об'єктів. Основним видом ієрархії структури класів є згадувана вище концепція *наслідування*, яке означає таке відношення між класами, коли один клас запозичує структурну чи функціональну частину одного чи декількох інших класів (відповідно, *одиночне* та *множинне наслідування*). Іншими словами, наслідування створює таку ієрархію абстракцій, у якій підкласи наслідують будову від одного чи декількох суперкласів, добудовуючи чи переписуючи їх компоненти. Основним видом структури об'єктів є *ієрархія агрегації*, у якій клас-спадкоємець знаходиться на більш високому рівні абстракції, ніж будь-який використаний при його реалізації (на відміну від ієрархії класів, у якій суперклас є узагальненням, а його спадкоємці – спеціалізаціями). Агрегація дозволяє фізично згрупувати логічно пов'язані структури, а наслідування з легкістю копіює ці загальні групи у різні абстракції.

Типізація – обмеження, що пред'являється до класу об'єктів, перешкоджаючи взаємозаміні різних класів (звужуючи можливість такої взаємозаміни).

Центральне місце у понятті типізації займає ідея *узгодження типів*. Наприклад, будь-який поліном можна подати у вигляді вектора його коефіцієнтів, будь-який вектор можна подати у вигляді рядкової або стовпцевої матриці, проте зворотний процес можливий далеко не завжди. Це – приклади сильної (строгої) типізації, коли прикладна область накладає правила та обмеження на використання та сполучення абстракцій. Строго типізовані мови програмування мають такі переваги над слабо типізованими:

– наявність контролю типів веде до зникнення незрозумілих збоїв у програмах під час їх виконання;
– раннє виявлення помилок, пов'язаних з конфліктами типів, значно полегшує процес написання та налагодження програми;
– декларація типів надає програмам елементи самодокументованості.

Від строгої типізації слід відрізняти статичну. Строга типізація слідкує за відповідністю типів, а статична типізація (статичне, або раннє зв'язування) визначає час, коли імена зв'язуються з типами. Статичне зв'язування означає, що типи всіх змінних та виразів відомі під час компіляції, динамічне (пізнє) зв'язування – що типи невідомі до моменту виконання програми. Концепції типізації і зв'язування є незалежними, тому в мові програмування можуть бути: типізація – сильна, зв'язування – статичне (Ada), типізація – сильна, зв'язування – динамічне (C++, Pascal), або відсутні і типи, і зв'язування динамічне (Smalltalk).

Паралелізм – властивість, що відрізняє активні об'єкти від неактивних.

Стійкість – властивість об'єктів існувати у часі та/або у просторі.

Абстрагування, інкапсуляція, модульність та ієрархія є головними у тому розумінні, що без будь-якого з них мова програмування не може вважатися об'єктно-орієнтованою. Типізація, паралелізм та стійкість є додатковими елементами об'єктного підходу – вони корисні, проте не обов'язкові.

Генеалогію об'єктних та об'єктно-орієнтованих мов програмування подано на рис. 1.2, а у табл. 1.4 наведено порівняльний аналіз основних мов програмування та зроблено висновок про належність мови до одного з класів.

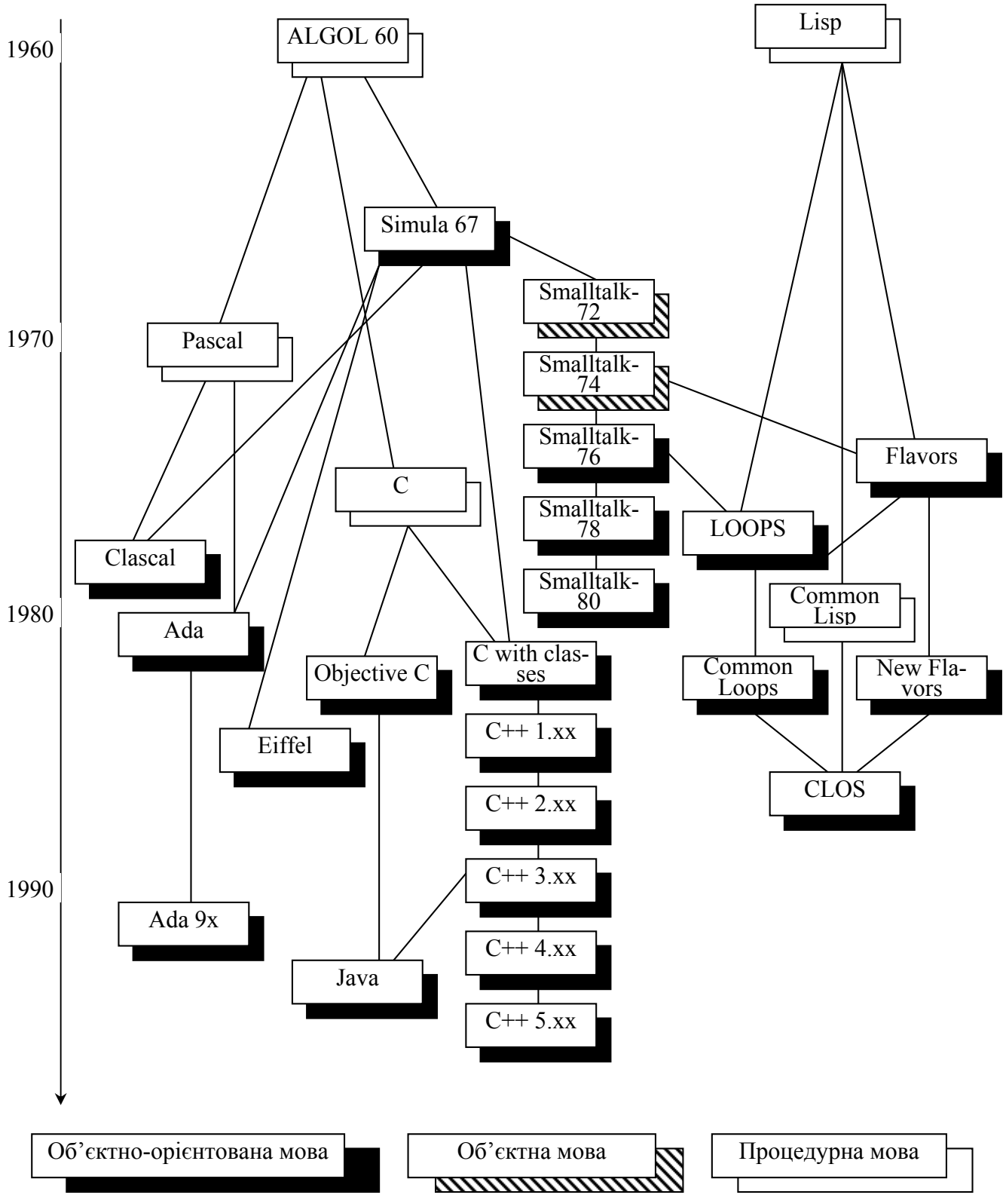


Рис. 1.2. Генеалогія об’єктних та об’єктно-орієнтованих мов програмування

Таблиця 1.4

Порівняльний аналіз основних мов програмування з позицій ООП

	Smalltalk	**Object Pascal**	**C++**	**Common Lisp Object System (CLOS)**	**Ada**	**Eiffel**
Абстракції						
Змінні об'єкту	Так	Так	Так	Так	Так	Так
Методи об'єкту	Так	Так	Так	Так	Так	Так
Змінні класу	Так	Ні	Так	Так	Ні	Ні
Методи класу	Так	Ні	Так	Так	Ні	Ні
Інкапсуляція						
Змінних	Приватні	Загально-доступні	Загально-доступні, захищені, приватні	Читання, запис, доступ	Загальнодо-ступні, приватні	Приватні
Методів	Загально-доступні	Загально-доступні	Загально-доступні, захищені, приватні	Загально-доступні	Загальнодо-ступні, приватні	Загально-доступні, приватні
Модульність						
Різновиди модулів	Ні	Модуль (інтерфейс – реалізація)	Файл (заголовок – тіло)	Пакет (монолітний)	Пакет (специфікація – тіло)	Блок (інтерфейс – реалізація)
Ієрархії						
Наслідування	Одиночне	Одиночне	Множинне	Множинне	Ні (входить у Ada9x)	Множинне
Шаблони	Ні	Ні	Так	Ні	Так	Так
Метакласи	Так	Ні	Ні	Так	Ні	Ні
Типізація						
Сильна типізація	Ні	Так	Так	Можлива	Так	Так
Поліморфізм	Так (одиночний)	Так (одиночний)	Так (одиночний)	Так (множинний)	Ні (входить у Ada9x)	Так
Паралелізм						
Багатозадачність	Непряма (за допомогою класів)	Ні	Непряма (за допомогою класів)	Так	Так	Ні
Стійкість						
Довгоживучі об'єкти	Ні	Ні	Ні	Ні	Ні	Ні
Тип мови	Чисто об'єктно-орієнтована	Спрощена об'єктно-орієнтована	Гібридна об'єктно-орієнтована	Гібридна об'єктно-орієнтована	Об'єктна (об'єктно-орієнтована у Ada9x)	Об'єктно-орієнтована

Проведений аналіз дозволяє зробити висновок про те, що найбільш придатною мовою для навчання програмуванню у об'єктній методології є мова C++ [30, 59, 108]. Ця мова є гібридною, її ядро – широко відома мова програмування C, що дозволяє організувати неперервне навчання студентів процедурній та об'єктній методології у межах однієї мови. Спочатку вивчається та підмножина C++, що містить основні конструкції мови C, а далі на вивченому базисі надбудовуються компоненти, що надають цій мові об'єктно-орієнтованих властивостей. Такий підхід, зокрема, застосовано у навчальному посібнику [140].

У табл. 1.5 узагальнено дані про кількість синтаксичних елементів у різних мовах програмування.

Таблиця 1.5

Порівняльний аналіз синтаксису основних мов програмування (за [145])

Мова	*Лексеми*	*Нетермінали*	*Термінали*	*Службові слова*
Algol–60	1085	119	92	25
Pascal H. Вірта	1012	110	84	35
Turbo Pascal 2	1184	124	87	42
Turbo Pascal 5	1331	127	87	48
Turbo Pascal 5.5	1410	135	87	52
Turbo Pascal 6	1488	143	89	55
Object Pascal	1825	180	90	83
Modula–2	887	70	88	39
Oberon	765	62	90	32
Oberon–2	726	43	91	34
C	917	49	123	27
C++	1662	126	131	47
Java	1771	174	121	48
Ada	2206	226	102	63

Розглядаючи синтаксис двох найбільш поширених мов програмування – C та Pascal, можна зробити наступні висновки:

1. Навіть у варіанті Н. Вірта кількість службових слів у мові Pascal більше, ніж у мові C (35 проти 27). В останній необ'єктній його реалізації (Turbo Pascal 5) ключових слів уже 48, що перевищує кількість ключових слів в об'єктно-орієнтованому C++ [139].
2. Object Pascal, що використовується у середовищі візуального програмування Delphi, містить 83 ключові слова, у той час як мова C++ в аналогічному середовищі C++ Builder містить всього 47 ключових слів.
3. Кількість лексем у Turbo Pascal 5 на 45% більше, ніж у мові C, в Object Pascal – на 10% більше, ніж у C++.
4. Для опису синтаксису Turbo Pascal 5 використовується 127 правил (терміналів), для опису Object Pascal – 180 правил. У той же час весь синтаксис C може бути заданий всього 49 правилами, а синтаксис C++ – 126. Таким чином, описати синтаксис об'єктно-орієнтованої мови C++ виявляється легше, ніж описати синтаксис необ'єктного Pascal.
5. Чим більше в мові нетерміналів, тим коротші програми на цій мові, тим компактніший запис програм цією мовою. З табл. 1.5 видно, що найбільше «багатослівна» мова – це Pascal у нотації Н. Вирта, а найкоротша – навіть не C, а його «спадкоємець» – C++.

Порівняння синтаксису мови дає нам об'єктивний показник його складності – по суті, кількості часу, затрачуваного на вивчення його синтаксису. Враховуючи синтаксичну простоту мови C++ у порівнянні з Object Pascal, повну підтримку у цій мові парадигм об'єктного підходу, чітку структуризацію системи захисту та наявність трьох рівнів доступу до програмного коду й даних, що дозволяє будувати логічно завершені системи класів та їх модулів, при програмуванні чисельних методів нами було обрано мову C++ як базову мову програмування [243], а Object Pascal – як додаткову.

В.А. Семенов у [166, 167, 168] виділяє такі головні можливості, що їх надає об’єктний підхід при розробці математичного програмного забезпечення:

1. Інкапсуляція чисельних методів у класах математичних об’єктів є більш сильною формою структуризації обчислювальних модулів, що підвищує концептуальну наочність застосовуваних чисельних підходів та регламентує коректну дисципліну роботи з даними об’єктів без порушення їх цілісності.
2. Розробник прикладного ПЗ виступає у якості користувача бібліотечних класів і замість підтримки внутрішніх даних може сконцентрувати зусилля на предметному аспекті розроблюваних програм (приймаючи до уваги складність динамічно розміщуваних даних математичних об’єктів, зокрема, розріджених матричних об’єктів, зазначена перевага є більш ніж помітною).
3. Механізм захисту даних, що забезпечує об’єктна технологія, сприяє підвищенню надійності розроблюваних програм.
4. Наслідування математичних об’єктів та поліморфізм чисельних методів, інкапсульованих ними, надає можливість гнучкої модернізації і розвитку математичного забезпечення як з урахуванням проблемної орієнтації, так і у відповідності до детальної класифікації об’єктів за їх математичними та обчислювальними ознаками і властивостями.
5. Реалізація чисельних методів у вигляді наслідуваних методів споріднених математичних класів дозволяє природним чином виразити міру спільності математичних понять та ступінь універсальності застосування тих чи інших чисельних підходів у кожному конкретному випадку.

Таким чином, математична об’єктно-орієнтована бібліотека може розглядатися у якості базового інструментального середовища для програмування чисельних методів.

При цьому, по-перше, відпадає необхідність починати нову програму

з нуля: базові типи векторів, поліномів, матриць тощо, оформлені у вигляді відповідних класів, можуть бути використані поряд із вбудованими числовими типами. Інструментальний характер базових класів тим більше усвідомлюється за необхідності перевизначення методу чи групи методів або породження нового, більш спеціалізованого типу даних.

По-друге, нові типи даних, що виражають потужні концептуальні поняття, утворюють своєрідну мову, максимально наближену до природної математичної, використовуючи яку можна програмувати чисельні методи у загальноприйнятих термінах та позначеннях [240].

По-третє, побудована система класів може бути доповнена та розширена у відповідності до предметної галузі, в якій її планується застосовувати. Специфічна для проблемної області система понять, властивостей, відношень знаходить своє відображення в системі класів, побудованій за принципом ієрархічної деталізації.

Так, розроблена нами бібліотека математичних класів знайшла застосування не лише в курсі чисельних методів [142], а й в курсах автоматики [141], фізики твердого тіла [46, 213], електротехніки [211, 212] та інших.

С. Шлєєр та С. Меллор [224, 225] пропонують діалектичну схему об’єктно-орієнтованого проектування та аналізу, що складається з п’яти етапів:

1. Вибір об’єктів та означення їх класів.
2. Ідентифікація атрибутів об’єктів та означення даних класів.
3. Ідентифікація властивостей об’єктів та операцій над ними і означення методів класів.
4. Виділення спільнот об’єктів, встановлення між ними відношень наслідування та організація ієрархій класів з використанням поліморфізму методів.
5. Аналіз взаємодії між об’єктами, тестування сценарію роботи з систе-

мою класів, можливе їх критичне переосмислення і, як наслідок, повернення до попередніх стадій розробки на новому рівні.

Застосування об’єктного підходу до розробки математичного програмного забезпечення передбачає перш за все проведення попереднього об’єктно-орієнтованого аналізу розглядуваної предметної галузі. Результатом такого аналізу повинна стати система узагальнень, що виражає основні предметні поняття, їх властивості та встановлює між ними необхідні класифікаційні відношення.

Складовою частиною об’єктно-орієнтованого аналізу є виявлення усіх можливих співвідношень наслідування між об’єктами, за яких множина класів може бути подана ієрархією споріднених об’єктів. В основі побудови такої ієрархії лежить та чи інша система ознак, що ідентифікує кожен об’єкт у відповідності до обраної класифікації. Оскільки способи виділення таких ознак можуть варіюватися у широких межах, при побудові ієрархій математичних класів будемо виходити з конструктивних властивостей об’єктів – в такому випадку будь-який обчислювальний алгоритм, який можна застосувати до об’єктів деякого математичного класу, також може бути застосований і до об’єктів будь-якого його нащадка, причому з високою ефективністю на множинах еквівалентних задач, що різняться лише типами математичних об’єктів.

Разом з тим, кожного разу, коли наявність спеціальних математичних властивостей об’єкта – класифікаційних критеріїв – допускає застосування більш ефективного обчислювального методу, відповідний метод загального класу може бути перевизначений. Це стає тим більш виправданим, якщо математичні особливості частинного об’єкта дозволяють замінити обчислювальну процедуру відповідним аналітичним перетворенням.

Таким чином, наслідування і поліформізм чисельних методів, інкапсульованих у відповідних математичних класах, забезпечує компроміс між необхідністю мати надійне, функціонально повне і уніфіковане алгоритмі-

чне ядро та можливість заміщення універсальних методів на частинні реалізації.

На думку В.А. Семенова, об'єктна тріада «математичний об'єкт – обчислювальний алгоритм – чисельна проблема» є фундаментальною (рис. 1.3). Дійсно, оскільки ці види об'єктів виражають основні конструктивні поняття обчислювальної математики, відповідне її відображення у класах мови C++ забезпечує бажану спільність математичних методів та програмних засобів розв'язування обчислювальних задач [123–125].

Рис. 1.3. Об'єктна тріада обчислювальної математики

Під *математичним об'єктом* будемо розуміти самостійну сутність, яка виражає деяку математичну категорію і є об'єктом обчислень. Математичними об'єктами є цілі, дійсні та комплексні числа, множини, вектори, матриці, функції, послідовності, ряди, області просторів, криві, поверхні тощо. Кожен математичний об'єкт володіє набором математичних ознак, які є основою для класифікаційних побудов. Самі по собі математичні об'єкти ще не складають обчислювальної задачі і є лише інструментальним засобом для її постановки та розв'язування.

Під *обчислювальними алгоритмами* будемо розуміти, власне, методи обчислювальної математики та деяку допоміжну інформацію, що визначає умови їхнього алгоритмічного використання. Будемо вважати, що кожний алгоритм призначений для розв'язування лише однієї проблеми, хоча й може використовуватися опосередковано при розв'язуванні задач різних типів. Наприклад, будь-який метод інтерполяції може бути алгоритмічною

основою для побудови сім'ї квадратурних формул. Проте для чисельного інтегрування, крім власне інтерполяційного методу, необхідно визначити методи оцінювання похибки наближення та стійкості, адаптивну стратегію вибору кроку та багато іншого. Тому кожного разу обчислювальний алгоритм пов'язуватимемо з однією єдиною проблемою. Обернене, взагалі, неправильно, оскільки для розв'язування чисельних проблем одного класу можуть використовуватися різні підходи. Крім параметрів чисельного методу, алгоритмічні об'єкти містять необхідну інформацію про задану точність, наявні обчислювальні ресурси, що виражаються звичайно граничною кількістю ітерацій, максимально можливою кількістю обчислень функцій та їх похідних тощо. Ця інформація визначає конкретні умови організації обчислювального процесу.

Під *чисельною проблемою* будемо розуміти класичну задачу обчислювальної математики, подану в стандартній уніфікованій формі. Прикладами чисельних проблем можуть бути системи лінійних рівнянь, проблема власних значень, задачі лінійного та квадратичного програмування. Серед нелінійних алгебраїчних проблем найбільш важливими є системи рівнянь, задачі умовної та безумовної оптимізації. Серед диференціальних рівнянь виділимо задачі Коші, крайові задачі для звичайних диференціальних рівнянь, інтегральні рівняння та рівняння у частинних похідних. Звичайно, наведені приклади складають лише найзагальніший перелік обчислювальної проблематики і потребують більш детальної класифікації.

Таким чином, виділено три основні види обчислювальних об'єктів і, відповідно, три пов'язаних з ними способи інкапсуляції чисельних методів як методів математичних, алгоритмічних та проблемних класів, що дозволяє нам побудувати курс чисельних методів у об'єктно-орієнтованій технології програмування. Методику активізації пізнавальної діяльності студентів при вивченні такого курсу викладено у наступних параграфах дослідження.

Висновки до першого розділу

Аналіз психолого-педагогічної, навчальної та методичної літератури з теми дослідження дозволив зробити висновки, покладені нами в основу методології і методики дослідження з даної теми.

Проблема активізації пізнавальної діяльності студентів є однією з актуальних проблем не лише педагогічної науки, а й вузівської практики, тому що в ній містяться витоки багатьох проблем: навчання мисленню, формування пізнавальних інтересів, розвитку самостійності тощо.

Пізнавальна активність – це складне інтегративне утворення особистості, що характеризує її індивідуальні особливості в процесі пізнавальної діяльності та складається з трьох компонентів: мотиваційного, змістовно-операційного та емоційно-вольового. Визначальними характеристиками пізнавальної активності є пізнавальна потреба, пізнавальний інтерес, пізнавальна ініціатива, пізнавальну надситуативність. Пізнавальна активність має мотиваційно-вольову природу, оскільки пов’язана з вольовими рисами особистості: старанністю, організованістю, наполегливістю, цілеспрямованістю, самокритичністю, самостійністю.

Пізнавальна активність формується шляхом розвитку її структурних компонентів. Нижній рівень розвитку пізнавальної активності вимагає формування змістовно-операційного компонента, на основі якого формуються мотиваційний та емоційно-вольовий компоненти. Середній рівень вимагає формування усіх трьох компонентів, особливо – мотиваційного. Вищий рівень пізнавальної активності вимагає формування органічної єдності усіх трьох компонентів, приділяючи при цьому особливу увагу розвитку емоційного-вольового компонента.

Активізація пізнавальної діяльності студентів, що є процесом і результатом стимулювання їх пізнавальної активності – це цілеспрямована

діяльність викладача, спрямована на розробку і використання такого змісту, форм, методів, прийомів і засобів навчання, які сприяють підвищенню пізнавального інтересу, активності, творчої самостійності студентів у засвоєнні знань, формуванні навичок і вмінь застосовувати їх на практиці.

Основними психолого-педагогічними умовами активізації пізнавальної діяльності студентів є:

– забезпечення єдності освітньої, розвиваючої та виховної цілей процесу навчання;
– педагогічно правильне використання принципів дидактики вищої школи;
– забезпечення емоційності навчання і створення сприятливої атмосфери;
– динамічність, різноманітність методів, прийомів, форм і засобів викладання і навчання, їх спрямованість на розвиток активної дослідницької діяльності студентів, надання пріоритетів методам і формам активного навчання;
– орієнтація студентів на систематичну самостійну роботу, забезпечення регулярності і ефективності контролю і оцінки успішності студентів;
– комплексне, педагогічно доцільне використання технічних засобів навчання і нових інформаційних технологій;
– використання системи психологічних і педагогічних стимуляторів активної навчальної діяльності.

Розглянуто передумови впровадження об’єктного підходу у практику викладання чисельних методів та реалізацію об’єктного підходу у об’єктно-орієнтованому програмуванні – найбільш природній методології програмування, яка, враховуючи особливості психічних процесів, дає можливість створювати чітко структуровані та осяжні складні програмні продукти. На основі синтаксичних ознак та принципів об’єктного підходу (абстрагування, інкапсуляції, модульності, ієрархії, типізації, паралелізму,

стійкості) проведено порівняльний аналіз основних мов програмування та виділено мову C++ як таку, що відповідає принципам об’єктного підходу та є найбільш синтаксично простою.

Інкапсуляція чисельних методів у класах математичних об’єктів є більш сильною формою структуризації обчислювальних модулів, що підвищують концептуальну наочність застосовуваних чисельних підходів. Наслідування математичних об’єктів та поліморфізм чисельних методів, інкапсульованих ними, надає можливість гнучкої модернізації і розвитку математичного забезпечення як з урахуванням проблемної орієнтації, так і у відповідності до детальної класифікації об’єктів за їх математичними та обчислювальними ознаками і властивостями.

Виділено фактори, згідно яких математична об’єктно-орієнтована бібліотека може розглядатися у якості базового інструментального середовища для програмування чисельних методів:

– базові типи векторів, поліномів, матриць тощо можуть бути використані разом із вбудованими числовими типами;
– нові типи даних утворюють своєрідну мову, максимально наближену до природної математичної, використання якої дає можливість програмувати чисельні методи у загальноприйнятих термінах та позначеннях;
– система класів, що складає основу об’єктно-орієнтованої математичної бібліотеку, може бути доповнена та розширена у відповідності до предметної галузі, в якій її планується застосовувати.

В результаті аналізу можливостей застосування об’єктного підходу до розробки математичного програмного забезпечення ми дійшли висновку, що на даному етапі для розвитку пізнавальної активності студентів при вивченні чисельних методів найдоцільніше використовувати бібліотеки математичних об’єктів – векторів, поліномів та матриць.

РОЗДІЛ ІІ. МЕТОДИЧНА СИСТЕМА АКТИВІЗАЦІЇ ПІЗНАВАЛЬНОЇ ДІЯЛЬНОСТІ СТУДЕНТІВ ПРИ ВИВЧЕННІ ЧИСЕЛЬНИХ МЕТОДІВ У ОБ'ЄКТНО-ОРІЄНТОВАНІЙ ТЕХНОЛОГІЇ ПРОГРАМУВАННЯ

§ 2.1. Методичні основи навчання чисельних методів у об'єктно-орієнтованій технології програмування

Структура методики будь-якої навчальної дисципліни, зокрема, чисельних методів, організується трьома основними питаннями: «навіщо вчити?», «чому вчити?» та «як вчити?». Основне призначення методики полягає у теоретичному та практичному розв'язанні цих питань. До системного розгляду процесу навчання в дидактиці вже розроблялися компоненти навчального процесу, які розташовувались звичайно у лінійному порядку, як це показано на рис. 2.1.

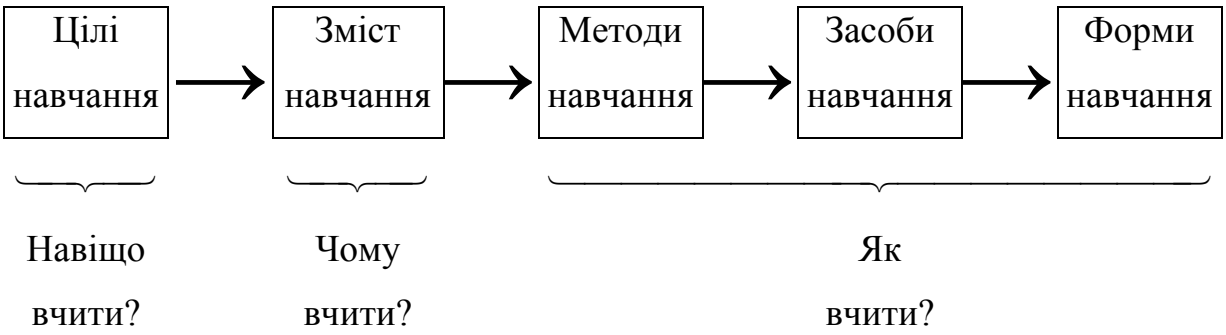


Рис. 2.1. Лінійна структура методики навчання.

Вперше системний підхід на рівні методики був застосований А.М. Пишкало [152]. У зв'язку з розробкою методики початкового навчання математики він запропонував підхід, у якому всі компоненти навчального процесу утворюють єдине ціле із визначеними внутрішніми зв'язками. Структуру такої системи зображено на рис. 2.2.

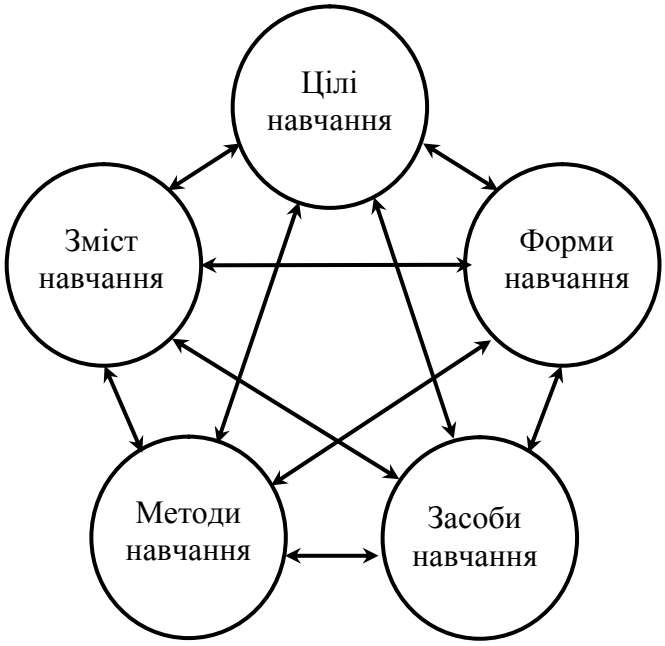


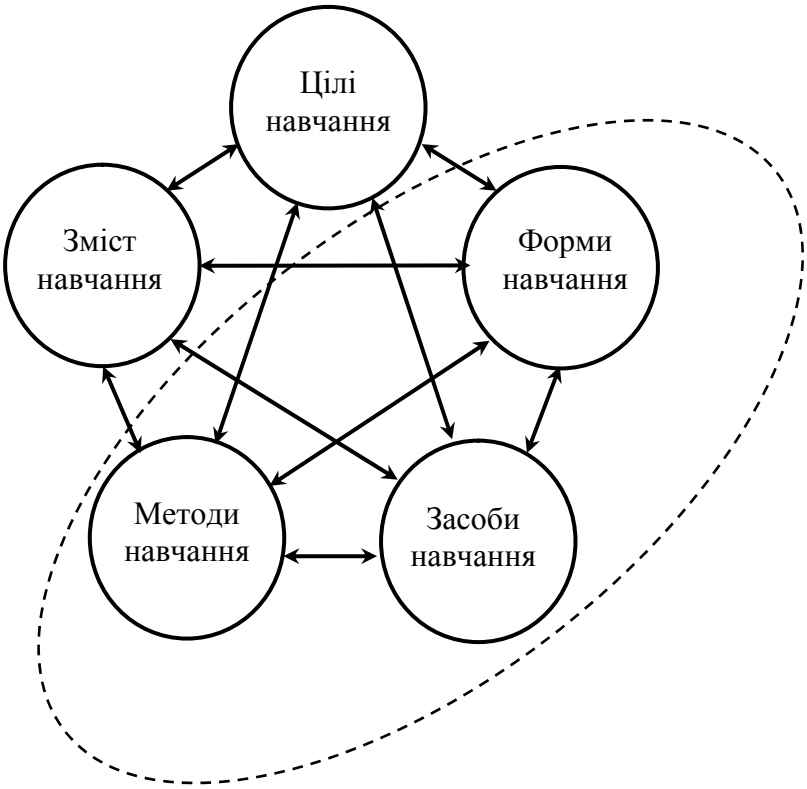
Рис. 2.2. Структура методичної системи навчання (за А.М. Пишкало)

Л.О. Черних, розглядаючи сукупність тих компонентів традиційної методичної системи, що відповідають на питання «як вчити?», вважає, що вони утворюють деяку підсистему єдиної системи: «*Технологією навчання* будемо називати таку підсистему методичної системи, яка включає в себе методи, засоби, форми навчання і має відповісти на питання «як вчити?»» [218, с. 18]. Схематичне подання структури методичної системи зображено на рис. 2.3.

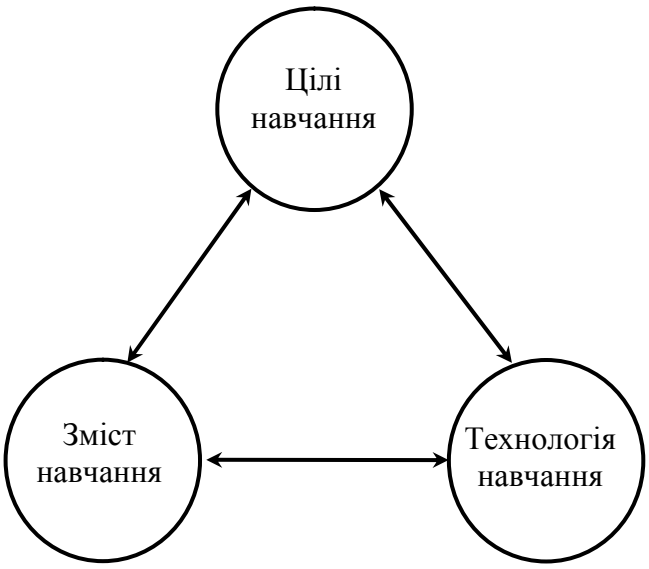
Виходячи з такої структури, визначимо цільовий, змістовний та технологічний компоненти методичної системи навчання чисельних методів у об’єктно-орієнтованій технології програмування.

Як було показано у § 1.2 нашого дослідження, цілі та зміст курсу чисельних методів на різних етапах його розвитку суттєво відрізнялись. На першому етапі – етапі накопичення досвіду наближених обчислень – чисельних методів як окремого розділу математики не існувало. Незважаючи на те, що окремі елементи цього предмету вивчалися у різних курсах, сис-

тематичного його викладу не існувало до виходу книги [97], яка відкрила собою другий етап розвитку чисельних методів.



а)



б)

Рис. 2.3. Структура методичної системи з виділеною підсистемою «техно-

логія навчання» (за Л.О. Черних)

На думку О.М. Крилова, ціллю курсу чисельних методів є «… показати: дійсно застосовувані практичні прийоми та способи обчислення коренів чисельних рівнянь, обчислення визначених інтегралів, використання тригонометричних рядів та наближеного розв'язування диференціальних рівнянь» [97, с. 7]. І.С. Березін та М.П. Жидков ціллю курсу чисельних методів вважали «… виклад методів чисельного розв'язування найважніших математичних задач» [16, с. 7], Б.П. Демідович та І.А. Марон – «… дати в певній мірі систематичне та сучасне викладення найважливіших методів та прийомів обчислювальної математики» [56, с. 9], Р.В. Хеммінг – «зв'язний виклад основних ідей обчислювальної математики» [216, с. 14].

А.А. Самарський, П.М. Вабіщевич, І.П. Гаврилюк та В.Л. Макаров вважають цілями навчання чисельних методів набуття навичок побудови математичних моделей, їх чисельного дослідження та програмування обчислювального експерименту [29, 84, 162, 163, 38]. До цієї ж думки схиляються В.Х. Прес, С.А. Текульський, В.Т. Веттерлінг та Б.П. Фленнері [225].

Ю.С. Рамський, визначаючи предмет курсу чисельних методів як сукупність методів, технічних прийомів і теоретичних результатів, необхідних для розв'язування на ЕОМ математичних моделей задач науки і техніки, ціллю навчання курсу вважає навчання техніці побудови, дослідження та реалізації на ЕОМ методів чисельного розв'язування типових математичних задач (моделей).

Виходячи з наведених означень та враховуючи *головну мету* нашої методичної системи навчання чисельних методів – активізацію пізнавальної діяльності студентів засобами об'єктно-орієнтованої технології програмування, – були виділені наступні ***цілі курсу***: ознайомлення з основними принципами побудови та дослідження математичних моделей; система-

тичний виклад найважливіших методів та прийомів обчислювальної математики, орієнтованих на машинні обчислення; навчання навичок побудови та використання об'єктно-орієнтованих математичних бібліотек як специфічного засобу програмування обчислювальних задач; формування культури дослідницької роботи з використанням обчислювального експерименту.

Зміст курсу містить сукупність двох взаємопов'язаних складових: теоретичної та практичної. Теоретична складова спрямована на формування в студентів наукового теоретичного мислення, здатності до коректної постановки задач, передбачення наслідків прийнятих рішень і дій, свідоме і обґрунтоване використання засобів НІТ в навчанні та трудовій діяльності. Практична складова пов'язана з набуттям студентами умінь: конструювання і добору найбільш ефективних алгоритмів та готових бібліотечних підпрограм для розв'язування задач; виконання об'єктно-орієнтованого аналізу чисельних методів та проектування об'єктно-орієнтованих бібліотек чисельних проблем, алгоритмів та об'єктів; проведення обчислювального експерименту – вибору математичної моделі, створення на її основі дискретної моделі, складання алгоритму розв'язання обчислювальної проблеми з використанням чисельних методів, створення, налагодження та тестування програми, розрахунку за програмою, обробки й аналізу знайдених чисельних результатів.

Навчальний матеріал, що входить до *змістовного компоненту* методичної системи, викладено у авторському навчальному посібнику «Методи обчислень в класах мови С++» [143]. Структуру цього посібника та засоби активізації пізнавальної діяльності студентів на різних етапах вивчення курсу розглянуто у наступних двох параграфах дослідження.

Технологічна підсистема розглядуваної методичної системи містить у собі три взаємопов'язаних компонента: організаційні форми навчання, методи навчання та засоби навчання, що пов'язані як між собою, так і з

усіма іншими компонентами методичної системи.

У вищий школі функціонують різноманітні ***організаційні форми*** навчання. З.І. Слєпкань пропонує розділити їх на дві основні групи за своїм функціональним призначенням:

1) форми організації засвоєння знань, формування навичок та вмінь і пошуку нових знань: лекції, семінарські, практичні, лабораторні заняття, консультації, екскурсії, експедиції, навчальні конференції, самостійна науково-дослідна робота студентів, навчальні та виробничі практики, курсові, дипломні роботи (проекти) тощо;
2) форми організації контролю знань, навичок і вмінь: колоквіум, залік, контрольна робота, екзамен (курсовий, державний), захист курсових і дипломних робіт, рубіжний контроль тощо [184, с. 89].

Викладання чисельних методів у об’єктно-орієнтованій технології програмування на початку педагогічного експерименту не мало відображення в методичній літературі. За цих умов функцію основного джерела навчальної інформації виконували *лекції*, основною метою яких було передавання знань від викладача студентам, створення орієнтувальної основи для подальшого засвоєння студентами навчального матеріалу, цілеспрямованому формуючому впливі на свідомість і почуття студента.

На початку кожної лекції пропонувався план і рекомендована література, причому близько 25% навчального матеріалу виносилося на самостійне опрацювання (форма контролю – міні-колоквіум після закінченню кожної теми). Основне утруднення, яке постало в процесі читання лекцій – майже повна відсутність у бібліотечних фондах матеріалу з чисельних методів, що інтенсивно розвиваються в останні десятиріччя. Наведемо лише декілька прикладів.

1) Нехай нам необхідно провести гладку криву через задані точки, координати яких відомі з достатньою точністю. Часто з цією метою рекомендують застосовувати сплайни, проте сучасні програми застосовують

здебільшого ерміттові кубічні криві та криві Без'є, що не описані у класичних підручниках.

2) Програми опрацювання матриць можна організувати таким чином, що вони будуть ефективні як на скалярних, так і на векторних комп'ютерах. Для цього відповідні алгоритми необхідно реалізувати за допомогою типових операцій нижнього рівня (так званих базових підпрограм лінійної алгебри – BLAS), опис яких поки що не перекладено з мови оригіналу (англійської).
3) Для обчислення визначених інтегралів рекомендують адаптивні методи, однак такі програмні пакети, як MATLAB, що включають в себе відсутні у класичних курсах формули Кронрода, дають значно кращі результати.
4) У сучасному програмному забезпеченні для чисельного інтегрування звичайних диференціальних рівнянь методи Рунге-Кутти поступово витісняються багатозначними методами.

З метою подолання цього методичного утруднення нами було розроблено електронний підручник «Інформатика-96» у форматі Windows Help, який, зокрема, містив наявний лекційний матеріал та методичні рекомендації до виконання лабораторних робіт (рис. 2.4).

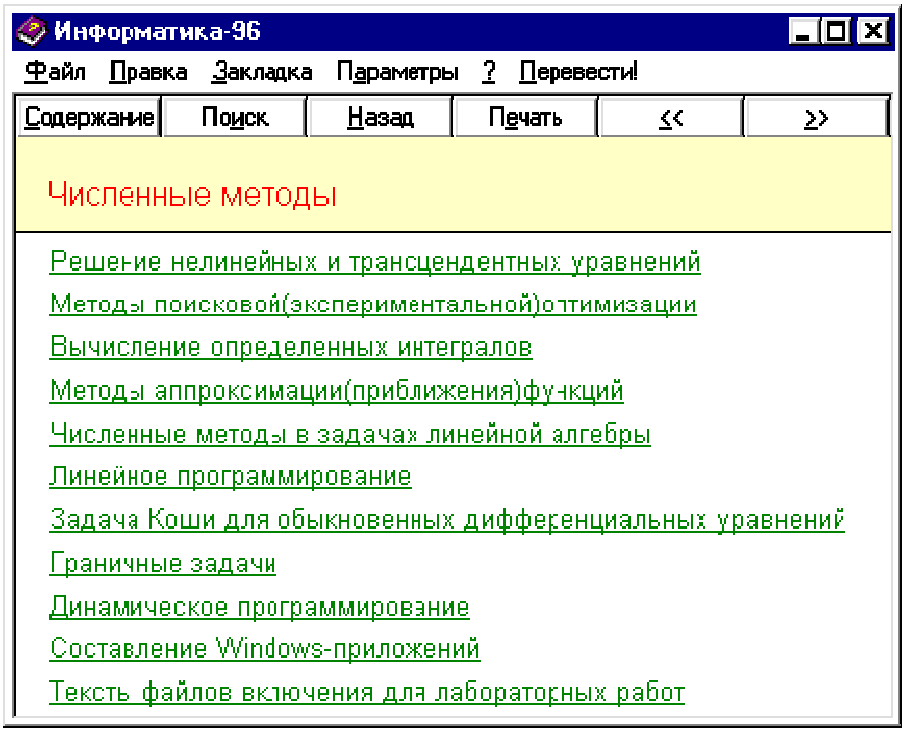


Рис. 2.4. Головне вікно електронного підручника «Інформатика-96»

Зауважимо, що цей електронний підручник не має змоги проконтролювати знання студентів, проте ця вада компенсується низкою переваг. По-перше, «Інформатика-96» має дуже низькі апаратні вимоги: вона функціонує навіть на AT-286 з 1 Мб пам’яті. По-друге, вона не має обмежень на глибину вкладеності і обсяг матеріалу, який організовано у вигляді гіпертекстової системи з великою кількістю перехресних посилань [41]. По-третє, навчальним матеріалом може бути не тільки багатошрифтовий текст та графіка, а й будь-які OLE-об’єкти: звук, відео, анімація і навіть електронні таблиці.

«Інформатика-96» не є закритою системою – вона дозволяє будь-кому доповнити та переробити її на свій смак, навіть не маючи програмістського досвіду. Для цього треба лише оволодіти будь-яким Windows-редактором, який дозволяє зберігати результати праці у збагаченому текстовому форматі RTF та засвоїти декілька спеціальних команд форматування тексту (методику цієї роботи описано у самій системі в одному з її численних розділів). Це призвело до появи цілої низки курсових робіт з інформатики, розроблених як додаткові модулі до цієї системи.

Основні результати використання цього електронного підручника у навчальному процесі Криворізького державного педагогічного університету доповідалися на Всеукраїнській конференції молодих науковців «Інформаційні технології в науці та освіті» (м. Черкаси, 1997) та описані у [174, 175].

Наявність такого підручника дозволила розвантажити лекції, зняти необхідність подання викладачем усього програмового матеріалу та приділити більше уваги міжпредметним зв'язкам, додати лекціям емоційного забарвлення історичними довідками з життя відомих математиків, відвести більше часу на розгляд прикладів, а, головне – ввести лекції узагальнення та систематизації вивченого матеріалу.

Наступним значним кроком в удосконаленні розроблюваної методики викладання стало видання авторського навчального посібника [143], що містив у собі більшість необхідного лекційного матеріалу. З урахуванням досвіду викладання курсу алгебри та числових систем на кафедрі математики за навчальним посібником [204] було вирішено запровадити *лекції-семінари*, на яких у формі диспуту відбувався розгляд теоретичного матеріалу, самостійно вивченого студентами за навчальним посібником.

Метою лекцій-семінарів є формування у студентів уміння аналізувати факти і події, порівнювати, узагальнювати і систематизувати набуті знання й уміння. Головною перевагою такою форми роботи є те, що лекції-семінари дозволяють розкрити необхідність і корисність засвоєння нових знань шляхом надання кожному виучуваному чисельному методу прикладного характеру. Це сприяє кращому розвитку мотиваційний компонент пізнавальної активності.

Якщо лекції були переважно присвячені розгляду теоретичних основ та алгоритмів чисельних методів, то на *практичних заняттях* перевага віддавалася розгляду особливостей програмної реалізації чисельних методів у обраній технології програмування. В процесі побудови об'єктно-

орієнтованої математичної бібліотеки студентам під керівництвом викладача пропонувалося виконати об'єктно-орієнтований аналіз чисельних методів, визначити структуру та зміст проблемних, алгоритмічних та математичних класів, спроектувати власні бібліотеки. Ефективним засобом активізації пізнавальної діяльності студентів при цьому є диференціація завдань на етапі об'єктивізації наявного математичного знання з метою вирівнювання знань студентів.

Досвід викладання показав, що оцінювання роботи студентів на лекціях-семінарах та практичних заняттях, яке виконується під час заняття, сприяє контролю й активізації пізнавальної діяльності студентів. Активізації навчально-пізнавальної діяльності студентів як на лекціях, так і на практичних заняттях сприяє також розв'язування вправ і задач професійної спрямованості, а також вправ міжпредметного характеру. Такі задачі мають бути математичною моделлю практичних задач, які будуть зустрічатися у майбутній професійній діяльності й у суміжних предметах [184, с. 92].

На *лабораторних роботах* студентами самостійно виконувалась програмна реалізація чисельних методів у комп'ютерному класі. Викладач при цьому виступає як організатор студентських груп, консультант, не лише розв'язувач, а й генератор проблемних ситуацій. Саме під час лабораторних робіт у студентів виникають нові ідеї, що згодом знаходять своє відображення у конкурсних, курсових та дипломних роботах. Саме тут найбільш доцільним є використання системи додаткових самостійних завдань для студентів з високим рівнем пізнавальної активності з метою рівномірного засвоєння матеріалу студентами з різним її рівнем. Викладачу при цьому слід, з одного боку, контролювати процес виконання робіт за визначеним планом, а з іншого – дозволяти свідомо відхилятись від нього, експериментувати, шукати нові шляхи, якщо це не шкодить засвоєнню предмету.

Яскравим прикладом такої діяльності є робота, виконана однією зі студентських груп. Основна мета їхньої діяльності полягала в розробці комп'ютерних інтерпретацій алгебраїчних структур і нових типів даних, що розширюють можливості мови C++ стосовно дослідження теоретико-числових проблем і проведення розрахунків високої точності. Виконавши об'єктно-орієнтований аналіз, студенти дійшли таких висновків:

1) неможливість виконання засобами машинної арифметики операцій над числами необмеженої довжини (точності) веде до необхідності створення відповідних бібліотек для роботи з такими об'єктами;
2) наявність загальних підходів до конструктивної побудови алгебраїчних систем веде до надбудови однієї алгебри над іншою так само, як і наслідування властивостей одного класу від іншого.

Така природна спільність класів мови C++ і алгебраїчних структур дала можливість побудувати комп'ютерні інтерпретації основних числових систем у вигляді розгалуженої ієрархії математичних класів, що дозволило використовувати в програмі нові типи даних – «довгих» цілих і раціональних чисел так само, як і вбудовані машинні типи даних обмеженої довжини, з можливістю їх спільного використання. Повернувшись до основної задачі, студенти застосували створені числові об'єкти при побудові параметризованого класу арифметичних векторів, компонентами якого можуть бути числові об'єкти будь-якої природи, зокрема, побудовані класи необмежених цілих і раціональних чисел. Це дозволило змоделювати такі числові об'єкти, як кільце комплексних чисел з раціональними компонентами, тіло кватерніонів і т.п., знову вийшовши за границі поставленої задачі. Основні результати цієї роботи доповідалися на Всеукраїнській конференції молодих науковців «Комп'ютерне моделювання та інформаційні технології в природничих науках» (м. Кривий Ріг, 2000 р.) та описані у [28].

Важливою умовою ефективності проведення лабораторних робіт є попередня перевірка викладачем готовності студента («допуск» до робо-

ти). З метою удосконалення пропонованої методики нами було розроблено автоматизований засіб навчання та тестування TUTOR, описаний у [175].

Самостійна робота студентів завершує завдання всіх інших видів навчальної роботи, вона не лише формує навички і уміння самостійного здобування знань, що важливо для здійснення неперервної освіти протягом всієї майбутньої трудової діяльності, а й має важливе виховне значення, оскільки формує пізнавальну самостійність. Пізнавальна самостійність, що формується на базі активності, характеризується багатьма вченими як якість особистості, її властивість. Ознаками пізнавальної самостійності при цьому є прагнення та вміння самостійно мислити, здатність орієнтуватися в новій ситуації, знайти свій підхід до нової задачі, бажання не лише зрозуміти засвоювані знання, а й способи їх здобування, критичний підхід до суджень інших та незалежність власних суджень.

І.Я. Лернер вважає активність умовою самостійності, і головне завдання вбачає в тому, щоб підняти активність до рівня самостійності [104].

Пізнавальна самостійність вважається багатьма дослідниками найвищим рівнем розвитку активності. Так, Д.Б. Богоявленська [18], виділяючи залежно від характеру пізнавальної активності суб'єкта рівні інтелектуальної активності, особливу увагу звертає на креативний, вищий за якістю рівень, який характеризується ініціативою в постановці задачі, у здійсненні цілепокладання, в умінні переходити до теоретичних узагальнень, коли емпіричні закономірності стають самостійними проблемами, тобто студент сам ставить творчу задачу або мету, сам намагається її розв'язати, а, отже, здійснює надситуативність.

Основним стимулом для самостійної роботи студентів в обговорюваному курсі є лекції, на яких ставляться проблеми, пропонуються конкретні завдання, рекомендується література та ресурси мережі Інтернет, вказується час виконання роботи, види та терміни контролю, можливості одержання консультації. Формою контролю самостійної роботи є індивідуа-

льна співбесіда чи колоквіум.

Самостійна робота студентів тісно пов'язана з науково-дослідною, головними завданнями якої є:

– опанування студентами науковим методом пізнання, поглиблення та творче засвоєння навчального матеріалу;

– формування у студентів навчально-дослідницьких навичок і вмінь;

– розвиток здібностей дослідної роботи, аналіз літературних та інших джерел знань;

– розвиток творчого мислення у вирішенні практичних питань;

– розширення теоретичного кругозору та наукової ерудиції майбутнього спеціаліста.

До форм науково-дослідної роботи студентів, що здійснюються в позаурочний час, відносяться виконання курсових і дипломних робіт; робота в студентських наукових гуртках; участь у виконанні госпдоговірних, держбюджетних наукових досліджень, договорів про творчу співдружність з галузевими організаціями, а також індивідуальна робота на кафедрах; робота в навчально-наукових педагогічних комплексах тощо. Результати науково-дослідної роботи висвітлюються в таких організаційно-масових заходах, як конкурси наукових робіт студентів, предметні олімпіади та олімпіади з спеціальностей, наукові конференції, виставки наукової та науково-технічної творчості студентів тощо.

За останні три роки відбулося більше 20 конференцій, присвячених об'єктно-орієнтованим обчисленням. Найбільш інформативними є Міжнародний симпозіум з наукових обчислень в об'єктно-орієнтованих паралельних середовищах ISCOPE (США, Санта-Клара, 2000; США, Сан-Франциско, 1999; США, Санта-Фе, 1998; США, Маріна-дель-Рей, 1997; США, Санта-Фе, 1996), Конференція з паралельних об'єктно-орієнтованих наукових обчислень POOSC (Португалія, Лісабон, 1999; Бельгія, Брюссель, 1998), Конференція з використання мови Java для наукових та інженерних

обчислень ACM (США, Пало Альто, 2000, 1999, 1998, 1997), SciTool – семінар з сучасних програмних засобів для наукових обчислень (Норвегія, Осло, 2000, 1998, 1996). Матеріали цих конференцій доступні у мережі Інтернет та можуть бути використані студентами у самостійній науково-дослідній роботі.

Другим компонентом технологічної підсистеми розглядуваної методичної системи є ***методи навчання***. У дидактиці вищої школи існують різні трактування цього поняття.

Ю.І. Машбиць, О.О. Гокунь, М.І. Жалдак та інші автори посібника [41] підходять до навчання як до управління учбовою діяльністю і розглядають метод навчання як спосіб управління та істотну детермінанту навчальної діяльності, яка реалізується у системі навчальних впливів, у способі включення учнів (студентів) у процес відтворення педагогом фрагменту учбової діяльності, у «полі самостійності» учнів (воно характеризується відхиленням від нормативного способу розв'язання учбових задач, при яких учням не надається допомога), у організаційних формах навчання і у модальності обміну інформацією між учнем (студентом) і вчителем (навчаючим засобом). А.М. Алексюк також визначає метод навчання як спосіб організації і управління з боку викладача пізнавальною діяльністю студентів [5, с. 46].

Р.А. Нізамов зазначає, що навчання не можна редукувати до управління. Воно включає в себе такі функціональні види діяльності:

а) викладача – організацію діяльності студентів із засвоєння знань, формування навичок і вмінь; виклад сутності наукових знань, складних теоретичних положень; контроль знань і вмінь; стимулювання пізнавальної діяльності студентів;

б) студента – засвоєння знань; формування вмінь; добування нових знань [128, с. 124].

З.І. Слєпкань під методом навчання розуміє способи роботи викла-

дача і студентів, за допомогою яких досягається оволодіння знаннями, навичками й уміннями, формується світогляд студентів, розвиваються їхні здібності [185, с. 105].

Виходячи з останніх двох означень, методи навчання можна поділити на такі дві групи:

1. Методи викладання – це система прийомів, що використовується викладачем з метою ефективного викладу знань, формування навичок і умінь, наукового світогляду, розвитку здібностей студентів, способи організації і управління пізнавальною діяльністю студентів. Основні методи викладання – лекція, розповідь, демонстрація, пояснення, бесіда.

2. Методи учіння – це система способів, що використовуються студентами у власній навчально-пізнавальній діяльності. Основні методи учіння – спостереження, експеримент, слухання-осмислення, вправи, вивчення джерел, моделювання.

Кожен метод викладання складається із сукупності прийомів: виділення головного, порівняння, доведення, складання плану, опорного конспекту тощо.

При вивченні чисельних методів у об’єктно-орієнтованій технології програмування *лекція* виступає і як форма організації навчальної діяльності, і як метод викладання. Характерною особливістю лекції як методу викладання є те, що в ній систематично та послідовно викладається великий за обсягом навчальний матеріал, зміст наукових проблем. Р.А. Нізамов показав, що найбільше активізує пізнавальну діяльність студентів проблемна лекція та її різновиди: лекція проблемного викладу, лекція проблемного засвоєння та комбінована лекція.

Наявність у студентів навчального посібника, що містить лекційний матеріал, та електронного підручника дозволяє приділити більше навчального часу лекціям-семінарам, які містять у собі елементи бесіди та лекції проблемного засвоєння. Такий тип лекції є найбільш ефективним для акти-

візації пізнавальної діяльності студентів, проте можливий лише за умови попереднього ознайомлення студентів із навчальним матеріалом за посібником з курсу.

Пояснення – найчастіше використовуваний нами метод. Це детальне, доступне тлумачення окремих понять, методів, змісту та елементів роботи на лабораторних та практичних заняттях.

Евристична бесіда – це метод запитання-відповідей, який активізує мислення студентів та процес пізнання в цілому і може бути використаний як елемент лекції-семінару, на колоквіумах, заліку, екзамені, під час індивідуального консультування студентів, що виконують творчі, конкурсні, курсові та інші види робіт. Найбільш повний аналіз дидактичних можливостей цього методу навчання наведено у кандидатському дослідженні та монографії М.З. Грузмана [50, 51].

До методів учіння, що найбільше впливають на розвиток пізнавальної активності студентів, відносяться *моделювання та обчислювальний експеримент*, що найчастіше застосовуються разом. Особливістю методу моделювання в курсі чисельних методів є те, що воно виступає не лише як метод навчання, а й як зміст та засіб навчання. Приклади моделювання різних об’єктів, наведені у наступних параграфах цього розділу, показують основні етапи його застосування як методу навчання.

Вивчення підручників, навчальних посібників, першоджерел та інших матеріалів – це метод самостійної пізнавальної діяльності студентів. Одним з джерел інформації про чисельне об’єктно-орієнтоване програмне забезпечення, що може бути ефективно використано для самостійної роботи, є Web-сторінка The Object-Oriented Numerics Page, що розташована за адресою http://oonumerics.org. Інформацію на ній згруповано тематично за такими розділами:

1. Список розсилки Object-Oriented Numerics List – це форум для обговорення наукового програмування у об’єктно-орієнтованому середовищі

та можливих стандартів для наукових об’єктно-орієнтованих обчислювальних компонент.

2. Вільно поширювані бібліотеки лінійної алгебри (PyMat, JAMA, MTL, SL++, GNUSSL, LAPACK++, ARPACK++, IML++, MV++, SparseLib++, ISIS++, ARPACK++, TNT, LinAlg, Newmat, CLHEP, BPKIT), обробки масивів (POOMA II, Blitz++, A++P++, AIPS++, CPPIMA, valarray<Troy>, IUE, WAILI), нейронних мереж та генетичних алгоритмів (PDP++, EO, CONICAL, GALib), багатознакових та розширених типів даних (EXTNUM, CLN, LiDIA, Apfloat, hfloat, doubledouble), диференціальних рівнянь (MOUSE, ODE++, Godess, PETSc, Diffpack, TIDE, BoxLib, OVERTURE), автоматичних диференціаторів та інтеграторів (FADBAD-TADIFF, PROFIL/BIAS), візуалізації (vtk), універсальні (MPI-2, CNCL, GTL, CTL, Tech-X, OOMF, CPPF77, MLC++, STL, FFTPACK++, FXT, Bench++, COOOL, CalcPlus, EFLIB).

3. Вільно поширювані програми.

4. Комерційні бібліотеки та програмне забезпечення.

5. Споріднені проекти.

6. Матеріали, на які є посилання.

7. Конференції.

8. Інші ресурсі мережі Інтернет.

9. Бібліографічні посилання.

Останнім компонентом технологічної підсистеми розглядуваної методичної системи є ***засоби навчання***. До засобів навчання, що використовуються в курсі чисельних методів у об’єктно-орієнтованій технології програмування, відносяться:

– підручники з чисельних методів, програмування, об’єктно-орієнтованого підходу;

– авторський навчальний посібник з курсу;

– науково-популярна, довідкова література;

– матеріали з мережі Інтернет, присвячені об’єктно-орієнтованим обчисленням;

– засоби наочності;

– технічні засоби навчання.

Підручники з виучуваного курсу охарактеризовані в § 1.1 дослідження, *авторський навчальний посібник* – у § 2.2–2.3, *матеріали з мережі Інтернет* – у поточному параграфі.

До *засобів наочності* належать моделі, рисунки, схеми, графіки, діаграми, інструменти, прилади, які демонструються з метою полегшення і покращення засвоєння програмового матеріалу.

Основним *технічним засобом навчання*, що застосовується нами, є комп’ютер, що виконує три основні функції: інформаційну, контролюючу та навчаючу.

До технічних засобів, що виконують інформаційну функцію, відноситься електронний підручник «Інформатика-96» (рис. 2.4), до засобів автоматизованого навчання та тестування – система TUTOR, розроблена автором спільно з О.П. Поліщуком.

При розробці системи TUTOR ми виходили з того, що одним з напрямів підвищення ефективності комп’ютерної підтримки учбового процесу є надання викладачу простої та зручної у користуванні інтерактивної інструментально-виконавчої системи з набором навчальних матеріалів та тестових завдань для перевірки знань [175, с. 213]. Така система повинна:

– обслуговувати функції створення, знищення, коректування учбових матеріалів і тестових завдань;

– бути гарно документованою і мати контекстно-залежну систему допомоги;

– бути відкритою і здатною до обміну даними з іншими програмами;

– мати інтуїтивно зрозумілі засоби керування.

Наявність у такій системі підсистеми навчання та підсистеми тесту-

вання й обліку успішності дозволяє викладачеві вирішити проблему одночасного опитування при проведенні занять у комп'ютерному класі. Студент, в свою чергу, отримує можливість самостійного повторного тестування для виправлення не задовольняючих його оцінок з вже опрацьованих тем.

Розроблена нами система TUTOR в одному з своїх варіантів є Windows-програмою, що може функціонувати на IBM-сумісних PC-386 та вище під керуванням Windows 3.1 та вище.

Система пропонує користувачу 2 основних режими роботи: «Розробка» та «Експлуатація» (навчання та тестування). Режим розробки призначений для викладача і захищений від випадкового та несанкціонованого використання кнопкою паролю, що перевіряє наявність у ключової дискети, а у разі її відсутності запитує пароль з клавіатури. Запуск системи здійснюється з ДОС (з побіжним автоматичним запуском Windows) або безпосередньо з Windows. Як показав досвід, для успішного початку курсу лабораторних робіт з чисельних методів у Windows студентам, зовсім з нею не знайомим, досить одного-двох занять.

Після запуску системи встановлюється режим експлуатації. В системі відсутні багаторівневі меню вибору режимів та різних установок, такі звичні для складних систем – з метою спрощення всі органи керування зосереджені в головному діалоговому вікні у вигляді панелі керування з набором кнопок та текстових субвікон (рис. 2.5). Такого роду інженерній підхід – створення пульту або панелі керування – дозволяє постійно мати перед очима режим роботи, вибрану предметну область, групу та ім'я студента, що, як показав досвід користування, прискорює роботу, на відміну від меню-орієнтованих систем.

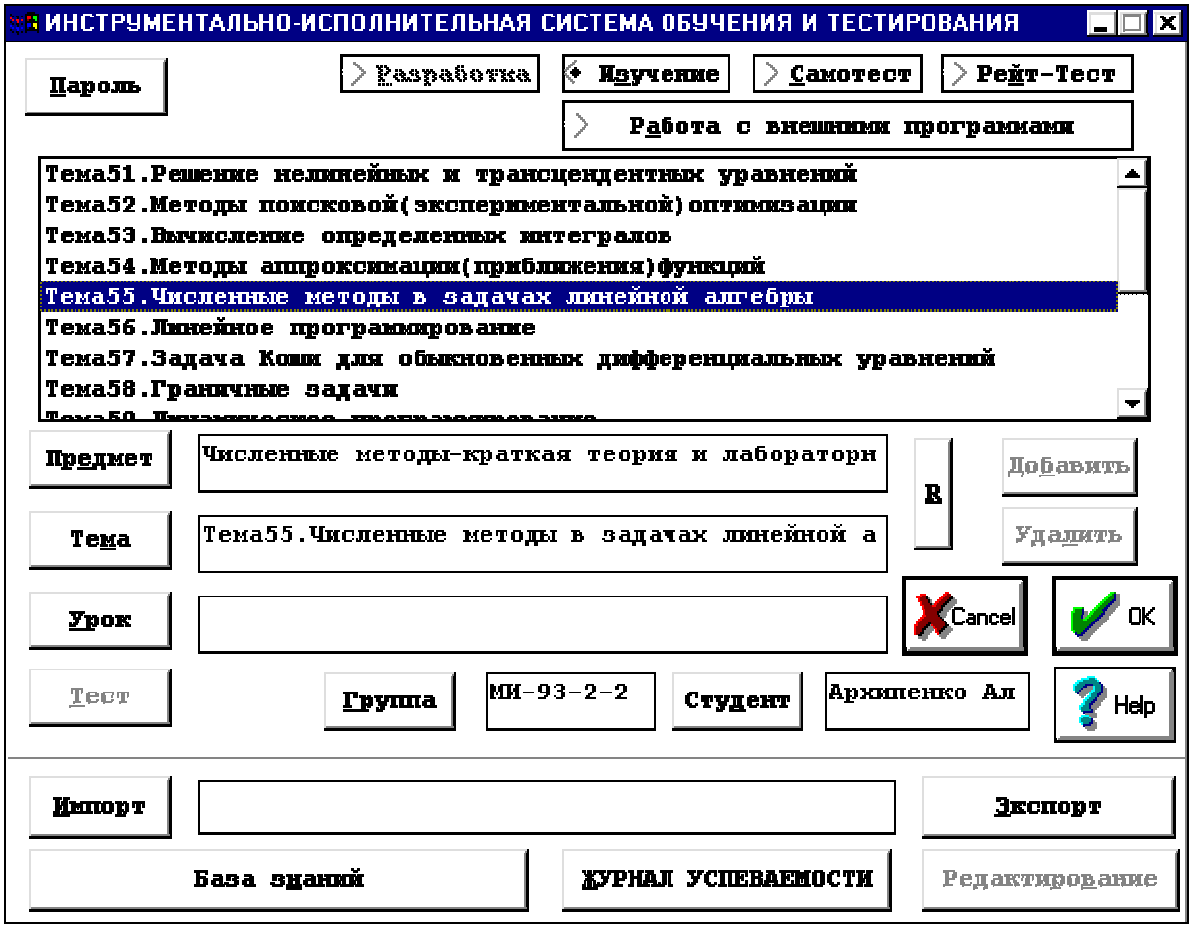


Рис. 2.5. Головне вікно системи TUTOR.

Для вибору виду робіт в режимі експлуатації у верхній частині панелі розташовано набір кнопок, що дозволяють обрати один з чотирьох видів діяльності, а у середній її частині міститься велике вікно для відображення різних режимно-залежних списків (з можливістю вибору елемента списку) або журналу успішності для перегляду. Режим навчання реалізується після вибору предметної області, який здійснюється за трирівневою схемою «Предмет–тема–урок». На будь-якому рівні натискання кнопки «ОК» викликає перехід у вікно демонстрації навчального матеріалу одного або послідовно усіх уроків теми чи предмету. В режимі навчання доступні усі стандартні функції перегляду (редагування навчального матеріалу в цьому режимі заборонено), виділення та переносу блоків тексту через буфер обміну Windows, який є одним із засобів обміну даними між різними програмами (рис. 2.6).

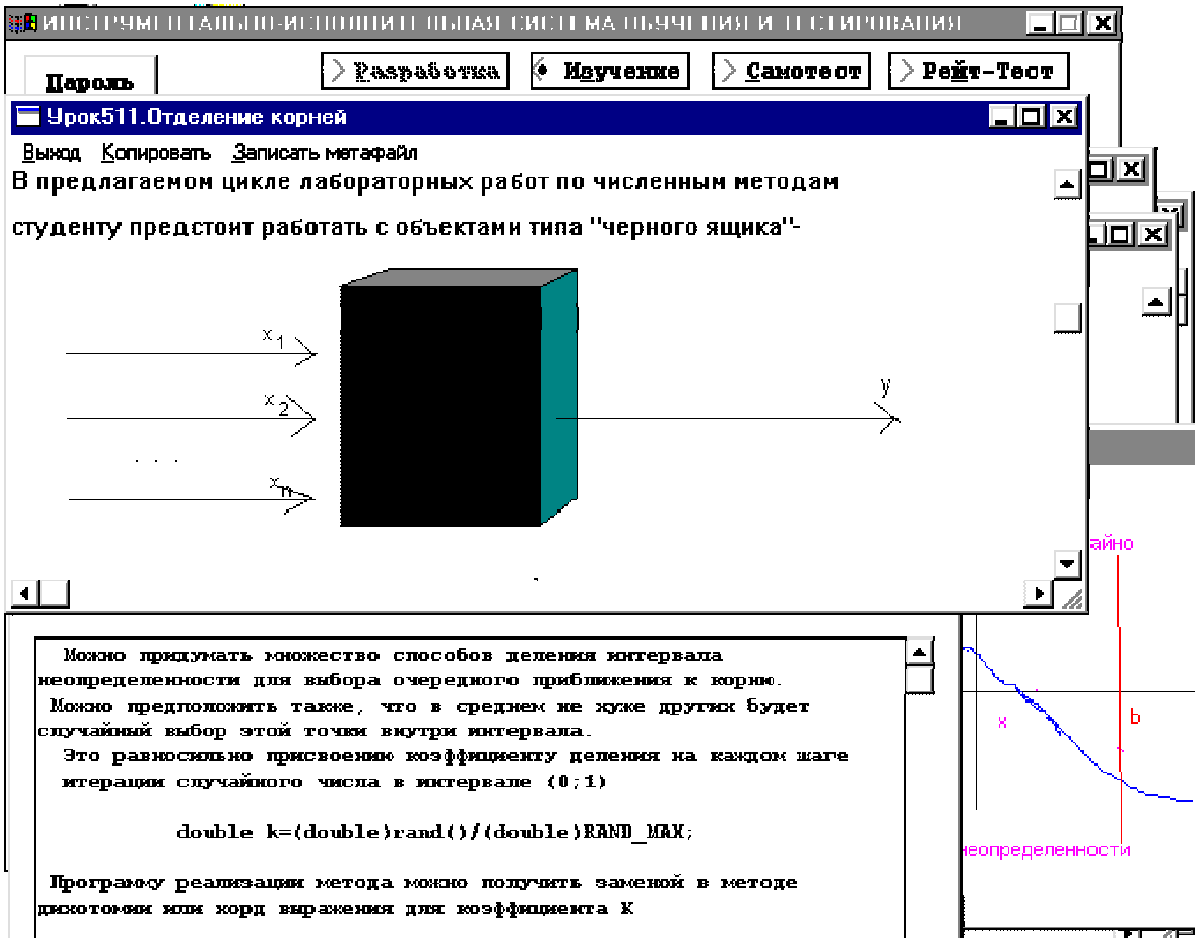


Рис. 2.6. Зовнішній вигляд системи у режимі перегляду навчального матеріалу.

Режим «Самотестування» призначений для анонімної перевірки знань без занесення оцінки до журналу успішності (рис. 2.7). В цьому режимі після вибору предметної області здійснюється перехід в діалогову панель тестування. Набір тексту відповіді здійснюється з клавіатури або копіюванням з буферу обміну – туди може бути занесена важка для запам’ятовування частина уроку або весь урок в режимі навчання. Система оцінювання тестового завдання демократична – заощаджений час на підготовку до відповіді призводить до підвищення оцінки з заданим коефіцієнтом, а прострочений – до незаліку тесту; за підказку доводиться розраховуватися деяким зниженням оцінки, яка виставляється після проходження обумовленої кількості тестів для даного уроку (не менше трьох).

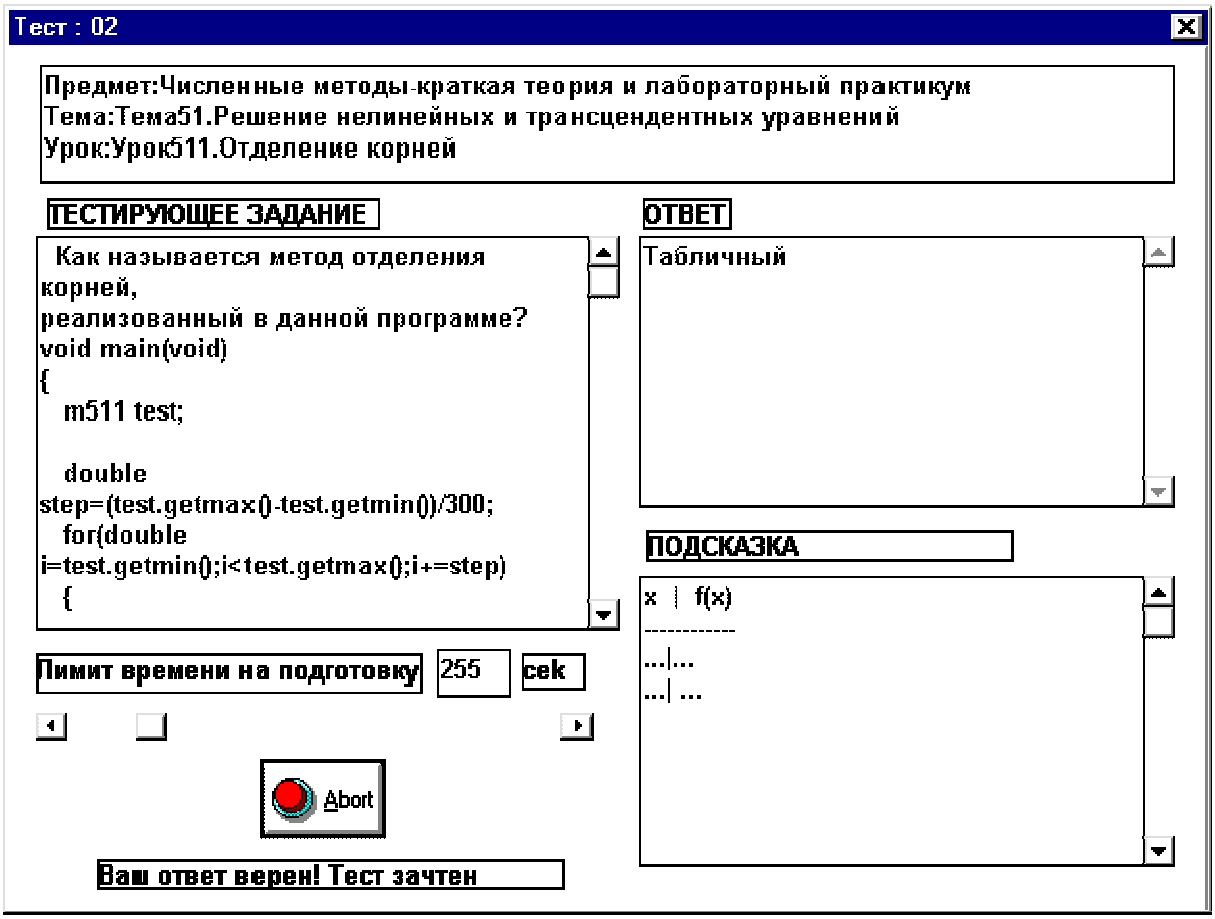


Рис. 2.7. Головне вікно підсистеми тестування.

Режим рейтинг-тестування з занесенням оцінок до журналу реалізується після вибору групи та прізвища у списку, які за необхідності поповнює викладач. Режим «Робота з імпортованими програмами» призначений для об'єднання «під одним дахом» усіх необхідних для вивчення курсу чисельних методів допоміжних інструментальних засобів. Це можуть бути навчальні програми з окремих розділів, електронні підручники, транслятори з мов програмування тощо.

Відмінність режиму розробки від режиму експлуатації полягає у доступності функцій коректування та редагування списків предметів, тем, уроків, тестів, груп та їх складу, імпортованих програм, текстів уроків і тестових завдань. Для формування уроків можуть використовуватися вже готові текстові файли: ця можливість виявляється не тільки корисною, а й незамінною у тих випадках, коли курс лекцій вже набрано раніше. В сис-

темі також передбачена можливість експортування навчального матеріалу у файли в ANSI чи OEM кодуванні (тільки текст) чи у форматі Windows Metafile (текст + графіка). Отримання твердої копії з файлу залишається на розсуд користувача.

Так само, як і виучуваний курс, інструментально-виконавча система навчання та тестування TUTOR є об'єктно-орієнтованою. Доступ до об'єктів класів «Урок» та «Тест», що зберігаються на диску, є прямим завдяки невеликим за обсягом асоціативним словникам, що виконують роль індексних файлів.

Апробація системи під час експериментального викладання курсу чисельних методів, окрім усунення ряду помилок і недоробок, показала такі основні результати:

1) Засвоєння органів керування досягається за 10-20 хвилин за умови хоча б мінімального досвіду спілкування з Windows; у разі його відсутності зайняття розпочинаються з підручника до операційного середовища Windows.

2) Системний журнал успішності відіграє ключову роль в активному використанні студентами навчальних матеріалів і відвідувань додаткових занять у комп'ютерному класі, що особливо ефективно для студентів, які мають проблеми із засвоєнням курсу. Спостерігаються спроби багаторазового тестування, конспектування еталонних відповідей при незаліках тестів з метою майбутнього використання, активний пошук відповідей в текстах уроків; студент мимоволі втягується у процес «самонатаскування» з погано засвоєних розділів.

3) Інший ефект – підвищення продуктивності – виникає за рахунок копіювання фрагментів програм з наведених в уроках методичних матеріалів до інтегрованого середовища розробки, що дає можливість підвищити складність завдань з програмування шляхом «збирання» програм з рекомендованих фрагментів.

4) За самою своєю побудовою система передбачає індивідуальний темп навчання, при якому студент може повернутися до будь-якої методичної рекомендації та прикладу, ще раз пройти погано засвоєний матеріал, та диференційоване тестування, що дозволяє встановити рівень вимог студента щодо оцінки.

Результати впровадження інструментально-виконавчої системи навчання та тестування TUTOR у навчальний процес Криворізького державного педагогічного університету дозволяють стверджувати, що ця система є ефективним засобом активізації пізнавальної діяльності студентів як при поданні програмового матеріалу, так і під час контролю його засвоєння.

Виділення цілей, змісту, методів, засобів та організаційних форм навчання чисельних методів у об'єктно-орієнтованій технології програмування та встановлення взаємозв'язків між ними дозволяють нам зробити висновок, що побудовано *методичну систему навчання чисельних методів у об'єктно-орієнтованій технології програмування, спрямовану на розвиток пізнавальної активності студентів*. Особливості активізації пізнавальної діяльності студентів на різних етапах вивчення курсу розкрито у наступних параграфах дослідження.

§ 2.2. Активізація пізнавальної діяльності на етапі об'єктивізації наявного математичного знання

Вивчення курсу чисельних методів у об'єктно-орієнтованій технології програмування відбувається у два етапи.

На першому етапі відбувається об'єктивізація математичного знання шляхом побудови базису у вигляді математичних класів векторів, поліномів та матриць, які інкапсулюють у собі операції над елементами відповідних множин та типові процедури обробки даних, що базуються на цих операціях. Після завершення цієї роботи студенти одержують можливість записувати у своїх програмах операції над відповідними типами у природній математичній нотації.

На другому етапі довільній групі чисельних методів ставиться у відповідність конкретна прикладна задача, яка вимагає їх використання. Так, при вивченні методів розв'язування диференціальних рівнянь ставиться задача комп'ютерного моделювання поведінки лінійної динамічної системи з розрахунком вільних та вимушених рухів під дією стандартизованих збурень типу імпульсного та ступінчатого впливів, гармонічних функцій та довільних функцій часу. Задача розв'язується як за допомогою програмної реалізації аналітичного операторного методу, так і за допомогою кінцево-різницевих схем Адамса-Башфорта-Моултона з можливістю порівняння результатів. При цьому інтенсивна робота з комплексними, матричними, векторними та поліноміальними об'єктами не створює жодних технічних утруднень за рахунок попередньо проведеної роботи. Це дозволяє досягти максимального наближення тексту програмної реалізації чисельного методу до його алгоритмічного опису.

Така побудова курсу дає можливість організувати вивчення чисельних методів на трьох рівнях: базовому, підвищеному та розширеному.

На базовому рівні студенти програмують прикладні задачі, що вимагають застосування чисельних методів, з використанням готової бібліотеки математичних об’єктів. Така бібліотека є допоміжним засобом, що полегшує програмну реалізацію. Студенти мають можливість розширювати її власними математичними, проблемними та алгоритмічними класами протягом усього курсу.

На підвищеному рівні перед тим, як приступати до програмування чисельних методів, студенти виконують об’єктно-орієнтований аналіз та проектування власної бібліотеки математичних об’єктів згідно запропонованої викладачем схеми. При цьому студент отримує можливість не лише розширення бібліотеки, а й повного її перепроектування з метою подання математичних об’єктів у найбільш суб’єктивно зручній формі.

На розширеному рівні студенти звертають увагу на інші числові об’єкти – гіперкомплексні числа [86], цілі та раціональні числа необмеженої довжини (точності) [28] тощо, не обмежуючись побудовою виділених класів векторів, поліномів та матриць. Реалізація цього рівня відбувається шляхом виконання курсових проектів, творчих та конкурсних робіт і виконуються, як правило, групою від двох до п’яти студентів.

Така диференціація дозволяє організувати спільне навчання студентів, що мають різні рівні пізнавальної активності із повним завантаженням членів кожної групи [10, 27]. Успішне засвоєння базового рівня відповідає середньому рівню розвитку пізнавальної активності, засвоєння підвищеного та розширеного – високому рівню розвитку пізнавальної активності. Студенти з низьким рівнем пізнавальної активності – це ті, що повністю (0–2 бали) або частково (3–5 балів) не засвоюють навчальний матеріал з курсу на базовому рівні.

На якому б рівні – базовому, підвищеному чи розширеному – не відбувалося вивчення чисельних методів, етап об’єктивізації математичного знання присутній завжди. На базовому рівні не вимагається побудова вла-

сної бібліотеки математичних об’єктів, проте її використання неможливе без розуміння механізму її побудови.

Перед тим, як приступати до побудови бібліотеки, доцільно у якості зразка розглянути декілька простих, але корисних класів, що можуть бути використані надалі у якості допоміжних. Зважаючи на те, що курс чисельних методів традиційно вивчається після вивчення процедурного та об’єктно-орієнтованого програмування, як ілюстрацію можна розглянути знайомий з курсу програмування слухачу і реалізований у бібліотеці C++ клас комплексних чисел, необхідний при вивченні наступного матеріалу.

Програмну реалізацію цього класу можна взяти безпосередньо із середовища розробки Borland C++ [87, 207], проте бажано, за можливістю, детально її відкоментувати. Цей матеріал слугуватиме своєрідним зразком при реалізації інших розглянутих у цьому параграфі математичних класів – векторів та поліномів. Процедурну реалізацію бібліотеки для роботи з комплексними числами наведено у додатку до [255].

План розгляду класу комплексних чисел

1. Комплексне число – це математичний об’єкт, відображенням якого в мові програмування є клас (структура):

```
class complex
{
```

2. Комплексне число – це складений математичний об’єкт, що в алгебраїчній формі має вигляд:

$$c=Re+Im\cdot i,$$

де *Re*, *Im* – дійсні числа (*Re* – дійсна, *Im·i* – уявна частини комплексного числа):

```
  /*Нам знадобляться робочі змінні для збереження Re і Im*/
  double re, im;
```

3. Комплексне число будемо вважати заданим, якщо задані обидві його частини – дійсна та уявна. Якщо уявна частина комплексного числа дорівнює нулеві, то маємо дійсне число. Зокрема, якщо і дійсну, і уявну

частину комплексного числа встановити в 0, матимемо число 0. Математичний об’єкт «комплексне число» задається за допомогою конструкторів класу «комплексне число» (complex):

```
  /*А далі визначимо загальнодоступні функції-члени класу і
почнемо з конструкторів */
 public:
  /*конструктор з ініціалізацією дійсної і уявної частин при
оголошенні комплексного числа в прикладній програмі */
  complex(double __re_val, double __im_val=0)
  { re=__re_val; im=__im_val;}
  //конструктор за замовчуванням
  complex() {re=im=0;}
```

4. При роботі з комплексними числами досить часто буває потрібно одержати значення дійсної та уявної частин окремо. Для цього у мові програмування використаємо функції доступу до відповідних даних класу:

```
  /*поточні значення дійсної і уявної частин повернуть функції
*/
  double real() {return re;}
  double imag() {return im;}
```

5. Два комплексних числа називають спряженими, якщо вони відрізняються тільки знаками уявних частин. Отже для того, щоб з існуючого математичного об’єкту класу «комплексне число» сконструювати новий об’єкт, який містить у собі комплексне число, спряжене до даного, необхідно викликати конструктор класу complex, параметрами якого є дійсна частина та заперечення уявної частини поточного числа:

```
  //спряжене даному комплексне число
  complex conj() {return complex(re,-im);}
```

6. У геометричній інтерпретації в декартовій системі *Re*, *Im* – координати точки на числовій площині з дійсною та уявною осями. Якщо використовувати полярну систему координат, то одержимо тригонометричну форму запису комплексного числа:

$$Re+Im\cdot i=\rho\cdot(\cos\varphi+i\cdot\sin\varphi)$$

де ρ=sqrt(Re^2+Im^2) – модуль комплексного числа, φ=arctg(Im/Re) – його аргумент, тобто кут між радіусом-вектором і дійсною віссю:

```
  //Модуль і аргумент комплексного числа
  double norm() {return(re*re+im*im);}
  double arg() {return (!re&&!im)?0:atan2(im,re);}
```

7. Нарешті, визначимо операції над комплексними числами.

```
/*Операції над комплексними об'єктами. Бінарні операції можуть
бути визначені в двох варіантах, що розрізняються місцем роз-
міщення результату - або зі зміною поточного значення об'єкта,
або без, із приміщенням результату в зовнішню (стосовно об'єк-
та) змінну комплексного типу. Ми визначимо прості бінарні опе-
рації як зовнішні дружні класу функції не члени класу. Тому
спочатку оголосимо їхні прототипи, а визначення наведемо поза
тілом класу. */
```

7.1. *Операції порівняння*. Два комплексних числа рівні, якщо рівні їх дійсні і уявні частини. Поняття «більше» чи «менше» для комплексних чисел не визначені.

```
  /*Операції перевірки рівності і нерівності комплексних чисел
*/
  friend int operator==(complex &,complex &);
  friend int operator!=(complex &,complex &);
```

7.2. *Алгебраїчні операції*.

– додавання і віднімання:

$$c_1\pm c_2=(Re_1\pm Re_2)+(Im_1\pm Im_2)\cdot i;$$

– множення:

$$c_1\cdot c_2=(Re_1\cdot Re_2-Im_1\cdot Im_2)+(Re_1\cdot Im_2+Re_2\cdot Im_1)\cdot i;$$

– ділення:

$$c_1/c_2=(Re_1\cdot Re_2+Im_1\cdot Im_2)/(Re_2^2+Im_2^2)+i\cdot[(Re_2\cdot Im_1-Re_1\cdot Im_2)/(Re_2^2+Im_2^2)];$$

– піднесення до степеня:

$$c^n=\rho^n\cdot(\cos(n\cdot\varphi)+i\cdot\sin(n\cdot\varphi)).$$

```
  /*Бінарні алгебраїчні операції над парами комплексних чисел,
комплексним і дійсним, дійсним і комплексним */
  friend complex operator+(complex &,complex &);
  friend complex operator+(double,complex &);
  friend complex operator+(complex &,double);
  friend complex operator-(complex &,complex &);
  friend complex operator-(double,complex &);
  friend complex operator-(complex &,double);
  friend complex operator*(complex &,complex &);
  friend complex operator*(complex &,double);
  friend complex operator*(double,complex &);
  friend complex operator/(complex &,complex &);
  friend complex operator/(complex &,double);
  friend complex operator/(double,complex &);
/*Унарні +, - і * комбіновані з присвоюванням арифметичні опе-
рації зробимо функціями-членами, що змінюють поточне значення
комплексного об'єкта. Ми об'єднаємо їхні оголошення з визна-
ченнями. */
  complex operator+() {return *this;}
  complex operator-() {return complex(-re,-im);}
  complex &operator+=(complex &)
  {re+=__z2.re; im+=__z2.im; return *this;}
  complex &operator+=(double __re_val2)
  {re+=__re_val2; return *this;}
  complex &operator-=(complex &__z2)
  {re-=__z2.re; im-=__z2.im; return *this;}
  complex &operator-=(double __re_val2)
  {re-=__re_val2; return *this;}
  complex &operator*=(complex &__z2)
  {
    re=re*z2.real()-im*z2.imag();
    im=re*z2.imag()+z2.real()*im;
    return *this;
  }
  complex &operator*=(double __re_val2)
```

```
  {re*=__re_val2; im*=__re_val2; return *this;}
  complex &operator/=(complex &__z2)
  {
    re=(re*z2.real()+im*z2.image())/
       (z2.real()*z2.real()+z2.imag()*z2.imag());
    im=(z2.real()*im-re*z2.imag())/
       (z2.real()*z2.real()+z2.imag()*z2.imag());
    return *this;
  }
  complex &operator/=(double __re_val2)
  {re/=__re_val2; im/=__re_val2; return *this;}

  /*На закінчення визначимо і функцію потокового виведення
комплексного об'єкта*/
  ostream &operator<<(ostream &, complex &)
  {return os<<"("<<x.real()<<","<<x.imag()<<")";}
};

/* Тепер наведемо реалізацію дружніх операторних функцій, що
не є членами класу та реалізують бінарні операції над комплек-
сними об'єктами. */

inline complex operator+(complex& __z1, complex& __z2)
{return complex(__z1.re+__z2.re, __z1.im+__z2.im);}
inline complex operator+(double __re_val1, complex& __z2)
{return complex(__re_val1+__z2.re, __z2.im);}
inline complex operator+(complex& __z1, double __re_val2)
{return complex(__z1.re+__re_val2, __z1.im);}
inline complex operator-(complex& __z1, complex& __z2)
{return complex(__z1.re-__z2.re, __z1.im-__z2.im);}
inline complex operator-(double __re_val1, complex& __z2)
{return complex(__re_val1-__z2.re,-__z2.im);}
inline complex  operator-(complex& __z1, double __re_val2)
{return complex(__z1.re-__re_val2, __z1.im);}
inline complex operator*(complex& __z1, complex& __z2)
```

```
{
  double r=z1.real()*z2.real()-z1.imag()*z2.imag();
  double i=z1.real()*z2.imag()+z2.real()*z1.imag()};
  return complex(r,i);
}
inline complex operator*(complex& __z1, double __re_val2)
{return complex(__z1.re*__re_val2, __z1.im*__re_val2);}
inline complex operator*(double __re_val1, complex& __z2)
{return complex(__z2.re*__re_val1, __z2.im*__re_val1);}
inline complex operator/(complex& __z1, complex& __z2)
{
  double r=(z1.real()*z2.real()+z1.imag()*z2.image())/
           (z2.real()*z2.real()+z2.imag()*z2.imag());
  double i=(z2.real()*z1.imag()-z1.real()*z2.imag())/
           (z2.real()*z2.real()+z2.imag()*z2.imag());
  return complex(r,i);
}
```

Розгляд даного класу не викликає особливих труднощів навіть у тих студентів, рівень пізнавальної активності яких є низьким. Відомості, використані при його побудові, не є новими: комплексні числа вивчаються в курсі алгебри, а об’єктно-орієнтоване програмування – в курсі інформатики. Введення класу, об’єкти якого моделюють комплексні числа, дозволяє, з одного боку, актуалізувати міжпредметні зв’язки математики та інформатики, а з другого – підвести студентів до необхідності використання ООП при моделюванні об’єктів числової природи.

Після розгляду класу комплексних чисел дається декілька простих вправ на його використання: введення та виведення комплексних чисел за допомогою потокових функцій, конструювання комплексних чисел різними способами, виконання різних операцій тощо. Для студентів з високим рівнем пізнавальної активності варто запропонувати завдання на модифікації запропонованого класу; зокрема, побудувати клас комплексних чисел, даними якого є не дійсна та уявна частина комплексного числа, а його

модуль та аргумент.

У якості довідкового матеріалу при цьому можна використати документацію до математичної бібліотеки SL++ (проект «Наукова бібліотека»), розробленої спеціалістами Європейського центру ядерних досліджень CERN та вільно поширюваною у Інтернет (http://wwwinfo.cern.ch/~ldeniau/html/sl++.html). Інші приклади побудови класу комплексних чисел наведено у [241, 262]. У [189] наведено опис математичного класу для роботи з двійково-десятковою арифметикою (bcd), який звичайно входить до поставки компілятора з мови C++. Структури класів bcd та complex мають багато спільних рис, тому можна запропонувати студентам розглянути його самостійно.

При вивченні комплексних чисел вводиться поняття векторного об'єкта на комплексній площині з числовими компонентами, яке далі розширюється до векторного об'єкта в n-вимірному просторі, розуміючи під ним послідовність з n чисел (у загальному випадку будь-якої природи) – складових вектора. Такі вектори називають арифметичними, а сукупність усіх таких векторів утворює n-вимірний векторний простір $\mathbf{R}^n$.

Перш ніж будувати математичний клас для моделювання об'єктів векторної природи, слід нагадати деякі відомості з розділу «Векторні простори» курсу алгебри [204].

Два вектори будемо вважати *рівними* тоді і тільки тоді, коли всі їхні компоненти рівні.

Множення вектора на число визначимо як операцію множення всіх складових вектора на це число.

Додавання векторів буде зводитися до додавання їх складових.

Нульовим будемо вважати вектор, у якого всі складові дорівнюють нулю.

Лінійно-залежними будемо вважати вектори $\boldsymbol{x}^{(1)}$, $\boldsymbol{x}^{(2)}$, …, $\boldsymbol{x}^{(n)}$, якщо існують такі не всі нульові константи C_1, C_2, …, C_n, що

$$C_1 \cdot \boldsymbol{x}^{(1)} + C_2 \cdot \boldsymbol{x}^{(2)} + \ldots + C_n \cdot \boldsymbol{x}^{(n)} = 0.$$

У протилежному випадку вектори вважаються *лінійно-незалежними*.

Останнє векторне рівняння рівносильне системі з n рівнянь з невідомими $C_1, C_2, \ldots, C_n$:

$$C_1 \cdot x_1^{(1)} + C_2 \cdot x_1^{(2)} + \ldots + C_l \cdot x_1^{(n)} = 0$$
$$C_1 \cdot x_2^{(1)} + C_2 \cdot x_2^{(2)} + \ldots + C_l \cdot x_2^{(n)} = 0$$
$$\ldots\ldots\ldots$$
$$C_1 \cdot x_n^{(1)} + C_2 \cdot x_n^{(2)} + \ldots + C_l \cdot x_n^{(n)} = 0$$

Якщо кількість векторів більше розмірності простору $n<l$, тобто число рівнянь менше числа невідомих, то система очевидно буде мати відмінні від нульового розв'язки C_j і вектори будуть відповідно лінійно-залежними. Іншими словами – кількість лінійно-незалежних векторів не перевищує розмірності простору. При $n=l$ (кількість рівнянь дорівнює кількість невідомих) система може мати ненульове рішення (а вектори можуть бути лінійно-залежними) тільки тоді, коли її визначник дорівнює нулю.

Скалярним добутком $(\boldsymbol{x}, \boldsymbol{y})$ двох векторів $\boldsymbol{x}(x_1, x_2, \ldots, x_n)$ і $\boldsymbol{y}(y_1, y_2, \ldots, y_n)$ називається число $\sum_{s=1}^{n} x_s y_s$.

Взаємно-ортогональними чи взаємно-перпендикулярними будемо називати два вектори, якщо їхній скалярний добуток дорівнює нулю.

Корінь квадратний зі скалярного добутку $(\boldsymbol{x}, \boldsymbol{x})$ вектора $\boldsymbol{x}(x_1, x_2, \ldots, x_n)$ на цей же вектор

$$\|\boldsymbol{x}\|^2 = (\boldsymbol{x}, \boldsymbol{x}) = \sum_{s=1}^{n} x_s^2 ; \qquad \|\boldsymbol{x}\| = \sqrt{\sum_{s=1}^{n} x_s^2} .$$

називають *довжиною* вектора $\boldsymbol{x}$.

Використання скалярного добутку дозволяє записати систему однорідних лінійних алгебраїчних рівнянь у вигляді:

$$(\boldsymbol{x}, \boldsymbol{a}_j) = 0 \ (i = 1, 2, \ldots, n; j = 1, 2, \ldots, n),$$

звідки видно, що її розв'язування зводиться до знаходження вектора $\boldsymbol{x}$, ортогонального до усіх векторів $\boldsymbol{a}_j$.

Розглянемо структуру класу для роботи з векторами:

```
template <class YourOwnFloatType> class vector
{
```

Клас для роботи з векторними об'єктами – параметризований, тобто визначається шаблон, за яким для конкретних типів, що підставляються замість YourOwnFloatType, компілятор автоматично генерує клас.

```
long m;
```

Вектор характеризуватимемо розмірністю, що може бути будь-яким натуральним числом. Скаляри можна розглядати як вектори розмірності 1, комплексні числа – як вектори розмірності 2 тощо. У загальному випадку розмірність вектора дорівнює кількості його складових, тобто розмірність вектора $(a_1, a_2, \ldots, a_n) \in n$.

```
YourOwnFloatType *vec;
```

Дані, що характеризують вектор, будемо зберігати за цим покажчиком. У даному випадку це – компоненти (координати) вектора.

```
virtual void In(istream &);
virtual void Out(ostream &);
```

Віртуальні функції читання з потоку і запису в потік; їх необхідно перевизначити в похідних класах для забезпечення можливості використання єдиних перевизначених операторів << і >>.

```
public:
```

У цьому розділі – загальнодоступні методи класу, що використовуються для дій над векторними об'єктами.

```
vector(char *);
```

Конструктор класу «вектор», параметром якого є ім'я текстового файлу. Передбачається, що першим числом у цьому файлі є розмірність вектора, після якого ідуть дані. Цей конструктор визначає розташування даних про

вектор – розмірність і компоненти.

```
vector();
```

Конструктор за замовчуванням, що створює нульовий вектор одиничної розмірності, необхідний при динамічному створенні масивів векторів. Геометрично такий вектор – це крапка на числовій прямій в точці «0»; такий підхід пов’язаний із труднощами розв’язування питання про розмірність вектора за замовчуванням і, звичайно, не є єдиним. Необхідність цього конструктора зумовлює компілятор обраної мови програмування.

```
vector(long);
```

Параметром цього конструктора є розмірність вектора; після створення компоненти вектора обнуляються. Якщо, приміром, прийнята розмірність – 5, то одержимо вектор (0, 0, 0, 0, 0).

```
vector(long, YourOwnFloatType *);
```

Цей конструктор намагається створити вектор розмірності, заданої першим параметром, і заповнити його даними з масиву. Корисний у тому випадку, коли компоненти вектора заздалегідь відомі. Нехай параметри цього конструктора – розмірність 3 і деякий масив з числами 1, 2, 3, 4, 5, 6, ... У такому випадку цей конструктор створить вектор (1, 2, 3).

```
vector(vector<YourOwnFloatType> &);
```

Конструктор копіювання (ініціатор копії), що створює новий вектор з наявного. Розмірність нового вектора встановлюється в розмірність попереднього, а дані переносяться без змін. В результаті його виконання одержується вектор, ідентичний копійованому, але такий, що знаходиться в іншій ділянці пам'яті.

```
~vector();
```

Деструктор. Звільняє динамічно розподілену пам’ять з-під компонентів вектора.

```
friend vector<YourOwnFloatType> operator+ (vector
<YourOwnFloatType> &, vector<YourOwnFloatType> &);
```

Дружня функція додавання двох векторів. Складає вектори тільки співпадаючих розмірів, вектор-результат конструюється і повертається.

```
friend vector<YourOwnFloatType> operator+= (vector
<YourOwnFloatType> &, vector<YourOwnFloatType> &);
```

Перевизначена операція скороченого додавання складає перший вектор із другим, модифікуючи при цьому перший вектор, записуючи в нього результат додавання, і повертає результат для того, щоб він міг брати участь в інших операціях над векторами.

```
friend vector<YourOwnFloatType> operator-(vector
<YourOwnFloatType> &, vector<YourOwnFloatType> &);
```

Дружня функція для віднімання векторів однакової розмірності; віднімає перший вектор від другого, конструюючи і повертаючи результат як новий вектор.

```
friend vector<YourOwnFloatType> operator-=(vector
<YourOwnFloatType> &, vector<YourOwnFloatType> &);
```

Скорочене віднімання, що працює так само, як і скорочене додавання: від першого вектора віднімається другий, результат знову записується в перший вектор, а його копія повертається.

```
friend YourOwnFloatType operator* (vector <YourOwnFloatType>
&, vector<YourOwnFloatType> &);
```

Скалярний добуток двох векторів однакової розмірності – число того ж типу, що і компоненти вихідних векторів.

```
friend vector<YourOwnFloatType> operator* (YourOwnFloatType,
vector<YourOwnFloatType> &);
```

Ще одна дружня функція, що перевантажує операцію множення по-іншому – як множення скаляра на вектор. Результатом є вектор тієї ж розмірності, що і вихідний. Добутком вектора на число буде вектор, кожний компонент якого помножений на це число.

```
friend vector<YourOwnFloatType> operator* (vector
<YourOwnFloatType> &, YourOwnFloatType);
```

Множення скаляра на вектор операція комутативна, що вимагає перевантажити операцію множення з такими самими аргументами, як і попередню, взятими в іншому порядку.

```
friend vector<YourOwnFloatType> operator*= (vector
<YourOwnFloatType> &, YourOwnFloatType);
```

Скорочене множення вектора на число. Перший аргумент цієї функції – вектор – після виконання даної операції модифікується, копія результату повертається.

```
friend ostream &operator<<(ostream &, vector<YourOwnFloatType>
&);
```

Перевантажена операція виведення вектора в потік. При цьому виводяться лише компоненти вектора, розділені пробілами; розмірність не виводиться. Модифікований потік виведення повертається.

```
friend istream &operator>>(istream &, vector<YourOwnFloatType>
&);
```

Введення вектора з потоку здійснюється шляхом прийому з потоку кількості чисел, що дорівнює його розмірності. Модифікований потік повертається для участі в подальших операціях уведення з нього.

```
vector<YourOwnFloatType> operator=(vector <YourOwnFloatType>
&);
```

Присвоювання – операція, що не може бути перевантажена з використанням механізму дружніх функцій. Першим, неявним параметром цієї функції є поточний об’єкт (*this), другим – об’єкт, що присвоюється поточному. Якщо розмірності присвоюваного і поточного векторів збігаються, то компоненти останнього копіюються. У протилежному випадку розмірність поточного вектора встановлюється в розмірність копійованого, а пам’ять під компоненти перерозподіляється, після чого виконується переписування даних. Значенням, що повертається, є сам поточний вектор.

```
vector<YourOwnFloatType> operator-();
```

Унарний мінус. Ця операція створює вектор тієї ж розмірності, що і поточ-

ний, з компонентами, що мають протилежний знак.

```
vector<YourOwnFloatType> operator+();
```

Цю функцію-член включено для повноти набору. Не виконуючи ніяких перетворень, вона повертає копію поточного вектора.

```
vector<YourOwnFloatType> operator~();
```

Метод, який для даного ненульового вектора конструює вектор, що має такий самий напрямок і одиничну довжину, тобто виконує нормування даного вектора за модулем.

```
YourOwnFloatType operator!();
```

Метод, який для вектора будь-якої розмірності визначає його модуль. Зауважимо, що при цьому робиться припущення, що довжина вектора виражається в одиницях того ж типу, що і компоненти вектора. Наприклад, для цілих векторів ця операція дасть лише наближений результат, а для багатовимірних векторів дана операція має сенс норми.

```
virtual long IsEqual(void *);
```

Ця віртуальна функція порівнює поточний вектор з об’єктом, що лежить за адресою, переданою через узагальнений покажчик. При цьому робиться неявне припущення про те, що покажчик містить адреса вектора. Привівши одержуваний покажчик до покажчика на вектор, ця функція намагається порівняти його з поточним.

```
friend long operator==(vector<YourOwnFloatType> &,
vector<YourOwnFloatType> &);
friend long operator!=(vector<YourOwnFloatType> &,
vector<YourOwnFloatType> &);
```

Використовуючи описану вище функцію, вводимо операторне порівняння двох векторів – перевірка на рівність і перевірка на нерівність.

```
YourOwnFloatType &operator[](long a);
```

Індексування елементів вектора – це операція, що повертає посилання на компонент вектора із заданим номером. Якщо цей номер виходить за межі розмірності вектора, після діагностики повертається спеціальний код по-

милки. Завдяки тому що дана функція посилальна, її можна використовувати в операторах присвоювання як праворуч, так і ліворуч (lvalue) – модифікація посилального об'єкту впливає на те, що під ним ховається, тобто на сам векторний компонент. Зазначимо, що вектор розмірності *a* можна індексувати від 0 до (*a*–1), а не від 1 до *a*.

```
long getm() { return m; }
```

Розмірність вектора можна отримати, використовуючи цей метод.

```
};
```

У Додатку А наведено інтерфейс та методи класу vector, побудованого за наведеною схемою. Ця реалізація може бути використана студентами як зразок при побудові власних векторних класів.

Проектування класів у мові C++, обраної нами в якості основної мови при програмуванні чисельних методів, є розширенням можливостей мови за рахунок введення нових типів даних. Застосування класу complex для роботи з комплексними числами розширює мову C++ до можливостей мови Fortran, в якій комплексна арифметика є вбудованою [246].

Застосування класу комплексних чисел дає студентам можливість у зручний спосіб записувати у програмі операції над комплексними числами, змішувати комплексну та дійсну арифметику тощо. Як приклад розглянемо процес розв'язування алгебраїчного рівняння другого степеня (квадратного рівняння), параметрами якого є коефіцієнти при другому, першому і нульовому степенях відповідно.

У загальному випадку коефіцієнти *a*, *b* та *c* при відповідних степенях *x* є комплексними, так само, як і результат. Згідно основної теореми алгебри – теореми Гаусса, алгебраїчне рівняння *n*-го степеня у комплексній області має рівно *n* коренів. Отже, результатом розв'язування такого рівняння є комплексний вектор розмірності *n*. Відповідно, результатом розв'язування квадратного рівняння буде комплексний вектор розмірності 2.

Побудований нами векторний клас є параметризованим. Це означає, що компонентами вектора можуть бути числа будь-якої природи, зокрема – комплексні. Для того, щоб визначити, який тип матимуть компоненти вектора, досить у кутових дужках після імені класу вказати цей тип. При програмній реалізації зручним є використання не повної назви vector<complex>, а його скороченого аналогу – наприклад, cvector. Для цього досить виконати операцію перейменування відповідного типу:

```
typedef vector<complex> cvector;
//Розв'язування квадратного рівняння з комплексними коефіцієн-
тами
cvector square(complex a,complex b,complex c)
{
 /*Знайдені комплексні корені повинні бути записані в двохко-
мпонентний комплексний вектор-результат */
  cvector res(2);
  if(a!=(complex)0)//перевіримо, чи не нульовий коефіцієнт при
x^2
  {
    //якщо так, шукаємо корені через квадратичний дискримінант
    complex D=b*b-4*a*c;
    res[0]=(-b+sqrt(D))/(2*a); //і заносимо їх у відповідні
    res[1]=(-b-sqrt(D))/(2*a); //компоненти вектора-результату
  }
  else res[0]=res[1]=-c/b; //інакше розв'язуємо рівняння 1
степеня
  return res;
}
```

У цьому прикладі комплексна арифметика використовується разом із дійсною. При його розгляді слід звернути увагу на те, що векторний клас використано лише як динамічний масив – ми не виконали жодних векторних операцій, проте застосували даний тип для зберігання даних, виходячи із його зручності.

При побудові універсальної функції для розв'язування алгебраїчного рівняння n-го степеня (відшукання всіх коренів полінома) можна використати комплексні вектори не лише для зберігання результату, але й для передавання параметрів – доцільним є передавання вектору коефіцієнтів полінома, корені якого відшукуються. При цьому степінь полінома визначається як розмірність полінома мінус одиниця.

Велика роль поліноміальної арифметики у курсі чисельних методів та можливість використання вектору для зберігання коефіцієнтів поліному, а векторних операцій додавання, присвоювання тощо як відповідних поліноміальних операцій ведуть до необхідності введення разом з типами «комплексне число» та «арифметичний вектор» нового типу даних – «поліном», який буде базуватися на типі «вектор». Отже, при побудові поліноміального класу студенти використовують побудований раніше векторний клас як базовий, застосовуючи ще одну концепцію ООП – наслідування.

При побудові цього класу слід розглянути наступні теоретичні відомості.

Поліном із невід'ємними степенями змінних у загальному вигляді записується так:

$$f(z)=a_n\cdot z^n+a_{n-1}\cdot z^{n-1}+\ldots+a_{n-k}\cdot z^{n-k}+\ldots+a_1\cdot z+a_0,$$

де a_0, a_1, …, a_k, …, a_n – задані числа (коефіцієнти полінома), z – змінна. Старший коефіцієнт a_n будемо вважати відмінним від нуля.

Значення z, при підстановці яких поліном обертається в нуль, називаються *коренями* (чи нулями) цього полінома, тобто корені полінома – це розв'язки рівняння

$$f(z)=a_n\cdot z^n+a_{n-1}\cdot z^{n-1}+\ldots+a_{n-k}\cdot z^{n-k}+\ldots+a_1\cdot z+a_0=0.$$

Це рівняння називають *алгебраїчним рівнянням n-го степеня*.

При діленні $f(z)$ на двочлен $(z-a)$ частка $Q(z)$ буде поліномом $(n-1)$-го степеня зі старшим коефіцієнтом a_n, остача R не буде містити z, тобто має

місце тотожність:

$$f(z)=(z-a)\cdot Q(z)+R.$$

Після підстановки в нього $z=a$ одержимо $R=f(a)$ – остача від ділення полінома на $(z-a)$ дорівнює $f(a)$ (теорема Безу). При діленні без остачі (з нульовою остачею) $f(a)=0$, тобто $z=a$ повинне бути коренем полінома. Знаючи цей корінь, можна виділити з полінома множник $(z-a)$:

$$f(z)=(z-a)\cdot f_1(z),$$

де

$$f_1(z)=b_{n-1}\cdot z^{n-1}+b_{n-2}\cdot z^{n-2}+\ldots+b_1\cdot z+b_0 \ (b_{n-1}=a_0)$$

і для виділення інших коренів треба розв'язати рівняння на один порядок нижче:

$$b_{n-1}\cdot z^{n-1}+b_{n-2}\cdot z^{n-2}+\ldots+b_1\cdot z+b_0=0.$$

Відповідно до основної теореми алгебри, будь-яке алгебраїчне рівняння має хоча б один дійсний чи комплексний корінь (наприклад, z_1) і ділиться без остачі на $(z-z_1)$, поліном-частка теж буде мати корінь (наприклад, z_2) і ділиться на $(z-z_2)$ і т.д. – таким чином, будь-який поліном степеня n розкладається на $n+1$ множник, один із яких дорівнює старшому коефіцієнту, а інші є двочлени виду $(z-a)$:

$$f(z)=a_n\cdot(z-z_1)\cdot(z-z_2)\cdot\ldots\cdot(z-z_n)$$

Такий розклад на множники єдиний з точністю до упорядкування.

Серед коренів полінома можуть бути *кратні* – необхідною і достатньою умовою того, що значення $z=a$ є коренем кратності k є обертання в нуль при цьому значенні полінома і всіх його похідних до $(k-1)$-ої включно і необертання в нуль k-ої похідної. Корінь кратності k деякого полінома є коренем кратності $(k-d)$ для d-ої похідної цього полінома, тобто якщо має місце розклад поліному:

$$f(z)=a_n\cdot(z-z_1)^{k_1}\cdot(z-z_2)^{k_2}\cdot\ldots\cdot(z-z_m)^{k_m},$$

де $z_1, z_2, \ldots, z_m$ – різні і $k_1+k_2+\ldots+k_m=n$, то розклад похідної буде:

$$f'(z)=(z-z_1)^{k_1-1}\cdot(z-z_2)^{k_2-1}\cdot\ldots\cdot(z-z_m)^{k_m-1}\cdot w(z),$$

де $w(z)$ – поліном, що вже не має загальних з $f(z)$ коренів.

Найбільшим спільним дільником (НСД) двох поліномів є добуток усіх спільних для них двочленних множників з меншими з двох варіантів показниками степеня. Якщо поліноми не мають загальних коренів, то вони взаємнопрості. Складання полінома – найбільшого загального дільника двох інших поліномів можна виконати відомим в арифметиці методом визначення НСД двох цілих чисел: поліном зі степенем не менше степеня другого ділимо на другий, потім другий ділимо на остачу від першого ділення, цю першу остачу ділимо на остачу від другого ділення і т.д. до одержання нульової остачі. Остання ненульова остача і є НСД: якщо вона не містить z, то поліноми взаємнопрості. Розділивши поліном на його НСД і НСД його похідної, одержимо поліном, що має всі прості корені, що збігаються з різними коренями вихідного полінома – так можна звільнитися від кратних коренів без розв'язування рівняння $f(z)=0$.

Основні операції з поліномами добре відомі з елементарної алгебри, тому при їх розгляді має сенс докладно зупинитися тільки на операції ділення поліномів з остачею і процедурах обчислення значень поліномів у зв'язку з тим, що ці операції часто використовуються майже у всіх розділах курсу чисельних методів.

Ділення поліномів здійснюють за правилами, прийнятими для ділення цілих чисел, при якому як результат одержують частку і остачу. Найпростіше реалізувати його за відомими правилами ділення «у стовпчик» за методом Евкліда.

Як перший елемент частки від ділення полінома $f_1(z)$ на поліном $f_2(z)$ беруть змінну z у степені, рівній різниці порядків поліномів діленого і дільника з коефіцієнтом, рівними частці від ділення коефіцієнта при старшому степені діленого на коефіцієнт дільника при старшому степені дільника; цей елемент множать на дільник і результат віднімають від діленого.

З отриманим різницевим поліномом зниженого порядку всі дії по-

вторюють, одержуючи другий і наступні елементи частки, поки його порядок не стане нижче порядку дільника – цей поліном являє собою остачу від ділення.

Згадавши теорему Безу про те, що остача від ділення полінома на двочлен ($z–a$) дорівнює значенню полінома при $z=a$, можна використовувати для обчислення значення полінома операцію ділення на двочлен. Схему ділення для цього частинного випадку можна спростити, використовуючи a замість **(**$z–a$**)** і підсумовування замість віднімання, а також використовуючи запис тільки коефіцієнтів без степенів z; відсутні степені позначаються нульовими коефіцієнтами. Така схема обчислень відома як *схема Горнера*.

Методи визначення коренів для поліномів степенів нижче п’ятої, відомі з курсу елементарної алгебри, розглянуті коротко при описі програмної реалізації у Додатку Б. Структуру класу для роботи з поліномами, побудовану у відповідності до рекомендацій з [27], наведено нижче.

```
template <class YourOwnFloatType> class polynom: public
vector<YourOwnFloatType>
{
```

Якщо ми запишемо коефіцієнти полінома в порядку спадання, включаючи нульові, ми одержимо упорядкований кортеж довжиною n+1*:* (a_n, a_{n-1}, …, a_2, a_1, a_0), тобто не що інше, як вектор розмірності n+1, що повністю характеризує заданий поліном. Тому цілком природним є те, що поліном буде базуватися на векторі. Як і векторний клас, він буде параметризованим. При цьому тип-параметр відноситиметься як до коефіцієнтів полінома, так і до значень, що підставляються в нього. Внутрішній формат для зберігання такого полінома буде $a_0+a_1x+a_2x^2+a_3x^3+a_4x^4+a_5x^5+...+a_{n-1}x^{n-1}$, що обумовлено вимогою зручності його індексування, а зовнішнє подання буде в канонічній формі – введення і виведення виконуватимуться, починаючи з коефіцієнта при найвищому степені.

```
void optimize();
```

У процесі роботи з поліномом може виникнути ситуація, коли його порядок необхідно зменшити у зв'язку з тим, що став нульовим коефіцієнт при старшому степені. Тому після кожної операції, що може привести до зміни степеня полінома, виконується перевірка, чи записаний поліном у канонічній формі. Якщо ні, то за допомогою цієї внутрішньої функції знижуємо його порядок дотих, поки поліном не буде перетворений у канонічну форму.

```
polynom<YourOwnFloatType> reverse();
```

У зв'язку з розходженням внутрішнього і зовнішнього представлення полінома іноді буває необхідно записати поліном у зворотному порядку.

```
void In(istream &);
void Out(ostream &);
```

За аналогією з векторами перевизначимо дві віртуальні функції введення і виведення. При цьому відпадає необхідність у перевантаженні операцій потокового введення-виведення: один раз визначені у векторному класі, вони використовують саме ці віртуальні функції, а визначення, з якого саме класу необхідно їх викликати, здійснюється вже на етапі виконання, у залежності від того, до якого типу перетвориться базовий покажчик. Зауважимо, що будь-який поліноміальний об'єкт можна перетворити до векторного.

```
public:
```

У цьому розділі ми розмістимо загальнодоступні дружні функції і методи поліноміального класу.

```
polynom(char *);
```

Параметром цього конструктора є ім'я файлу, у якому знаходяться дані у виді [Степінь Полінома-1] [Вільний Член] [Коефіцієнт При Першому Степені] … [Коефіцієнт При Старшому Степені].

```
polynom(long,YourOwnFloatType *);
```

Конструктор, що приймає два параметри – зменшений на одиницю стеінь полінома і покажчик на дані – коефіцієнти полінома, починаючи з вільного

члена.

```
polynom(polynom<YourOwnFloatType> &);
```

Конструктор копіювання робить зліпок із заданого полінома.

```
polynom();
```

Конструктор за замовчуванням створює поліном особливого виду – нуль-поліном, тобто поліном розмірності 1 (відповідно степеня 0), вільний член якого дорівнює нульовому елементу типу-параметра поліноміального класу.

```
polynom(long);
```

У деяких випадках буває корисно задати спочатку степінь полінома, але тимчасово залишити невизначеними його коефіцієнти. Цей конструктор створює поліном заданої розмірності й обнуляє коефіцієнти, порушуючи канонічну форму запису полінома. Це буває необхідно рідко і тільки в тих випадках, коли коефіцієнти полінома стають відомими після того, як заданий його степінь. Зрозуміло, і в цьому випадку параметром конструктора є степінь полінома–1.

```
polynom<YourOwnFloatType> operator-();
```

Поліном, протилежний до поліному $P(x)=a_nx^n+a_{n-1}x^{n-1}+...+a_2x^2+a_1x+a_0$, визначається як $-P(x)=-a_nx^n-a_{n-1}x^{n-1}-...-a_2x^2-a_1x-a_0$, тобто знак коефіцієнтів при степенях полінома змінюється на протилежний.

```
polynom<YourOwnFloatType> operator+();
```

Унарний плюс – операція, що повертає копію поточного полінома.

```
friend polynom<YourOwnFloatType> operator+(polynom
<YourOwnFloatType> &, polynom<YourOwnFloatType> &);
```

Дружня функція додавання поліномів приймає два параметри – поліноми-доданки, і повертає результуючий поліном-суму. Сумою двох поліномів, $f(x)=a_nx^n+a_{n-1}x^{n-1}+...+a_2x^2+a_1x+a_0$ і $g(x)=b_mx^m+b_{m-1}x^{m-1}+...+b_2x^2+b_1x+b_0$ степенів n і m відповідно ($n\geq m$) називають поліном $c_nx^n+c_{n-1}x^{n-1}+...+c_2x^2+c_1x+c_0$, де $c_i=a_i+b_i$ (i=0, 1, …, m), $c_i=a_i$ (i=m+1, …, n). Зрозуміло,

після одержання полінома-суми його необхідно записати в канонічній формі.

```
friend polynom<YourOwnFloatType> operator+=(polynom
<YourOwnFloatType> &, polynom<YourOwnFloatType> &);
```

Дружня функція, що реалізує операцію скороченого додавання двох поліномів, додає перший поліном до другого і записує результат у перший, повертаючи його копію для подальших перетворень.

```
friend polynom<YourOwnFloatType> operator-(polynom
<YourOwnFloatType> &, polynom<YourOwnFloatType> &);
```

Так як операція додавання поліномів оборотна, то ми можемо визначити віднімання поліномів, використовуючи операції додавання й одержання полінома, протилежного до даного. Таким чином, різницею двох поліном $f(x)$ і $g(x)$ є поліном $P(x)$ такий, що $P(x)=f(x)+(-g(x))$.

```
friend polynom<YourOwnFloatType> operator-=(polynom
<YourOwnFloatType> &, polynom<YourOwnFloatType> &);
```

За аналогією зі скороченим додаванням, ми можемо визначити скорочене віднімання як дружню функцію, перший параметр якої модифікується.

```
friend polynom<YourOwnFloatType> operator*(polynom
<YourOwnFloatType> &, polynom<YourOwnFloatType> &);
```

Добутком двох поліномів,

$f(x)=a_nx^n+a_{n-1}x^{n-1}+\ldots+a_2x^2+a_1x+a_0$ і $g(x)=b_mx^m+b_{m-1}x^{m-1}+\ldots+b_2x^2+b_1x+b_0$,

називають поліном $c_{n+m}x^{n+m}+\ldots+c_1x+c_0$, де

$$c_i=\begin{cases} a_ib_0+a_{i-1}b_1+\ldots+a_0b_i, & i=0,1,\ldots,m; \\ a_ib_0+a_{i-1}b_1+\ldots+a_{i-m}b_m, & i=m+1,\ldots,n; \\ a_nb_{i-n}+a_{n-1}b_{i-n+1}+\ldots+a_{i-m}b_m, & i=n+1,\ldots,\mathrm{n}+m. \end{cases}$$

Крім того, ми можемо знаходити добуток поліномів за правилом множення сум:

$$f(x)\cdot g(x)=\sum_{i=0}^{n}a_ix^i\cdot\sum_{k=0}^{m}b_kx^k\ .$$

```
friend polynom<YourOwnFloatType> operator*=(polynom
```

```
<YourOwnFloatType> &, polynom<YourOwnFloatType> &);
```

Скорочене множення полінома на поліном.

```
friend polynom<YourOwnFloatType> operator*( YourOwnFloatType,
polynom<YourOwnFloatType> &);
friend polynom<YourOwnFloatType> operator*(polynom
<YourOwnFloatType> &, YourOwnFloatType);
```

Множення числа на поліном, так само як і множення полінома на число – це операція, що впливає на коефіцієнти полінома, і відповідає множенню числа на вектор (вектора на скаляр), тобто:

$$\lambda f(x)=\lambda a_n x^n+\lambda a_{n-1}x^{n-1}+\ldots+\lambda a_2 x^2+\lambda a_1 x+\lambda a_0.$$

```
friend polynom<YourOwnFloatType> operator*=(polynom
<YourOwnFloatType> &, YourOwnFloatType);
```

Скорочене множення полінома на число.

```
friend long operator<(polynom<YourOwnFloatType> &,
polynom<YourOwnFloatType> &);
friend long operator>(polynom<YourOwnFloatType> &,
polynom<YourOwnFloatType> &);
friend long operator<=(polynom<YourOwnFloatType> &,
polynom<YourOwnFloatType> &);
friend long operator>=(polynom<YourOwnFloatType> &,
polynom<YourOwnFloatType> &);
```

Набір операцій для порівняння поліномів. Два поліноми вважаються рівними, якщо вони однакового степеня і коефіцієнти при відповідних степенях *x* рівні. У протилежному випадку поліноми не рівні. При цьому вважається, що перший поліном менше другого, якщо в канонічній формі його степінь нижче (розмірність менше). Якщо ж ці поліноми однакової розмірності (а, відповідно, і степеня), то послідовно порівнюємо їхні коефіцієнти, починаючи зі старшого степеня. Перше розходження в коефіцієнтах і визначає, який знак необхідно поставити між цими поліномами.

```
YourOwnFloatType &operator[](long);
```

Індексація полінома – операція, що приймає як параметр степінь одночле-

на, коефіцієнт при якому необхідно повернути.

```
YourOwnFloatType operator()(YourOwnFloatType);
```

Одержати функціональне значення полінома в заданій точці ми можемо, використовуючи цю функцію. Її задача – або просумувати добутки коефіцієнтів полінома на відповідний степінь аргументу даної функції, або обчислити це значення за теоремою Безу.

```
friend long div(polynom<YourOwnFloatType> &,
polynom<YourOwnFloatType> &, polynom<YourOwnFloatType> &,
polynom<YourOwnFloatType> &);
```

З курсу алгебри відома теорема, яка говорить про те, що для будь-яких двох поліномів $f(x)$ і $g(x)$ існує єдина пара поліномів $l(x)$ і $r(x)$, що задовольняють умові $f(x)=g(x)l(x)+r(x)$, де степінь $r(x)$ менше степеня $g(x)$ чи $r(x)$ – нуль-поліном. $f(x)$ – ділене, $g(x)$ – дільник, $l(x)$ – неповною часткою, а $r(x)$ – остача від ділення $f(x)$ на $g(x)$.

```
friend polynom<YourOwnFloatType> operator/(polynom
<YourOwnFloatType> &, polynom<YourOwnFloatType> &);
friend polynom<YourOwnFloatType> operator%(polynom
<YourOwnFloatType> &, polynom<YourOwnFloatType> &);
```

Використовуючи описану вище функцію, визначимо дві допоміжні операторні функції – для добування цілої частини і остачі від ділення.

```
long IsEqual(void *);
```

Віртуальна функція з таким самим ім’ям, визначена у векторному класі, призначена для порівняння поточного вектора з вектором, адреса якого передається через узагальнений покажчик. Така ж операція виконується і для поліномів з однією єдиною метою – використовувати однократно, у векторному класі, визначені операторні функції == і !=, що збільшує гнучкість і універсальність даного класу за рахунок використання механізму віртуальних функцій.

```
polynom<YourOwnFloatType> operator=(polynom<YourOwnFloatType>
&);
```

Присвоювання полінома-параметра поточному.

```
friend polynom<YourOwnFloatType> derive(polynom
<YourOwnFloatType>, long);
```

Поліном, як функція аналітична, зручний тим, що для нього завжди можна знайти аналітичну похідну. Як відомо, похідною степеневої функції є теж степенева функція, тому похідна від полінома буде теж поліном. Параметром даної функції є порядок похідної – кількість разів, що поліном диференціюється.

```
friend polynom<YourOwnFloatType> integral(polynom
<YourOwnFloatType>, long);
```

За аналогією з аналітичною похідною для полінома можна визначити аналітичний інтеграл. Параметром даної функції є кратність інтегрування.

```
friend polynom<YourOwnFloatType> pow(polynom
<YourOwnFloatType>, unsigned int);
polynom<YourOwnFloatType> operator^(unsigned int);
```

Використовуючи введену раніше операцію множення, ми можемо визначити ступінь полінома як перевантажену операцію чи як дружню функцію.

```
polynom<YourOwnFloatType> operator^=(unsigned int);
```

Скорочений степінь.

```
};
```

Визначення поліноміального класу так само, як і визначення класу для роботи з векторами, не вимагає засвоєння нового матеріалу – при побудові цього класу ми обмежуємося лише повторенням вивченого у суміжних курсах алгебри та інформатики матеріалу. Це не є випадковістю, адже всю роботу, що виконують студенти на етапі побудови математичних класів, вони могли виконати і в рамках курсу інформатики.

Актуалізація математичного знання, що виконується на початку побудови кожного класу, дає можливість провести його об’єктивізацію у найбільш раціональний спосіб. Поступовість та повторюваність дій, що виконуються при цьому, дозволяє навіть для студентів з низьким рівнем

пізнавальної активності створити ситуацію успіху, вселити в них впевненість у свої силах (створити установку «Я можу»). При цьому не слід вимагати самостійної побудови кожного класу – виконання повного циклу об'єктно-орієнтованого аналізу, проектування та програмування. На етапі об'єктивізації математичного знання такі студенти повинні усвідомити інструментальну роль математичних класів, що будуються, та навчитися використовувати об'єкти цих класів при програмуванні чисельних методів.

Після вивчення поліноміального класу для набуття навичок його застосування слід запропонувати серію вправ на сумісне використання дійсних та комплексних чисел, векторів та поліномів. Цікавою роботою, що також спирається на вивчений раніше матеріал, є програмування методів розв'язання алгебраїчних рівнянь 3-го та 4-го степенів за методами Кардано-Тартальї та Феррарі. Для студентів з високим рівнем пізнавальної активності можна запропонувати самостійно вивчити один з методів визначення поліноміальних нулів.

Чисельні методи розв'язування цієї задачі будуються за таким шаблоном: вибирається перше (за відсутності апріорних даних – довільне) значення кореня і обчислюється значення функції при цьому значенні; тепер ставиться задача скоригувати поточне значення x_k так, щоб відповідне значення полінома виявилося ближче до нуля:

$$x_{k+1}=x_k+d.$$

Доводиться визначати напрямок і величину кроку корекції d. Для нашого випадку, коли ліва частина нелінійного рівняння – поліном $P(z)$, тобто легко (і аналітично) диференційована функція, найбільш придатним буде, очевидно, класичний метод Ньютона чи його модифікації. Алгоритм Ньютона виходить із простих геометричних співвідношень – крок корекції можна визначити як катет прямокутного трикутника, іншим катетом якого є значення полінома в поточній точці, а тангенс протилежного йому кута є похідна полінома в тій самій поточній точці:

$$d=-P(z_k)/P^{(1)}(z_k),$$

$$z_{k+1}=z_k-P(z_k)/P^{(1)}(z_k),$$

де k – номер ітерації.

Визначальною особливістю цього методу є його гарантована збіжність на всій комплексній площині для поліномів степеня вище першого. До того ж швидкість збіжності не залежить від початкового наближення. Можна обчислити похідну лише в точці першого наближення і використовувати її значення на всіх наступних кроках, жертвуючи деяким зниженням швидкості збіжності:

$$z_{k+1}=z_k-P(z_k)/P^{(1)}(z_0).$$

Неприємності можуть очікувати нас на положистих ділянках функції, коли перша похідна близька до нульового значення – у цьому випадку занадто великий крок корекції «викидатиме» у далеку від кореня область. Це особливо неприємно, якщо ми використовуємо лише однократне диференціювання в точці першого наближення – випадкове влучення на положисту ділянку в цьому першому наближенні викликає ефект «нишпорення» пошукової процедури з великим кроком. Для усунення цього ефекту можна обмежити величину кроку деяким припустимим значенням, наприклад, роблячи його пропорційним не першій похідній, а тій, яка досить далека від нульового значення чи використовувати інші прийоми відходу від положистої ділянки кривої. У цьому випадку алгоритм може виглядати, наприклад, так:

1. Для початку ітерації задамося точністю ε, з якою необхідно знайти розв’язок, і початковим наближенням до деякого кореня $z_0=x_0+iy_0$.
2. Знаходимо похідну полінома в точці z_k (k – номер ітерації). Якщо вона дорівнює нулю, диференціюємо $P(z)$ далі доти, поки $P^{(j)}(z_k)$ не стане відмінною від нуля.
3. Наступне наближення до кореня знаходимо за формулою

$$z_{k+1} = z_k + t\left(-\frac{P(z_k)}{P^{(j)}(z_k)}\right)^{\frac{1}{j}},$$

де параметр t добирається так, щоб виконувалася умова:

$$|P(z_{k+1})|<|P(z_k)|.$$

4. Перевіряємо виконання умови $|P(z_{k+1})|<\varepsilon$; якщо умова не виконується, переходимо до пункту 1. Якщо умова виконується, то z_{k+1} вважаємо коренем рівняння, поліном $P(z)$ ділимо на двочлен $(z–z_k)$ і одержуємо поліном $(n–1)$-го степеня, для якого повторюємо всі обчислення, починаючи з п. 1.

Зауважимо, що перед початком пошуку можна позбутися від кратних коренів діленням вихідного полінома на НСД його і першої похідної, але потім доведеться визначати кратність коренів, доки їхня кількість не стане рівною порядку полінома.

Описаний алгоритм можна реалізувати за допомогою такої функції:

```
typedef polynom<complex> cpolynom;
typedef vector<complex> cvector;

/*метод Ньютона пошуку комплексних коренів полінома */
cvector newton(cpolynom p)
{
  double t=1,eps=1e-8,j;

  //результуючий вектор буде мати розмірність,
  //рівну порядку полінома
  cvector result=p.getm()-1;

  /*якщо вектор-результат одномірний, то маємо справу з полі-
номом першого степеня*/
  if(result==cvector())
  { result[0]=-p[0]/p[1]; return result; }
```

```
  //якщо двовимірний - із квадратним рівнянням
  if(result.getm()==2) return square(p[2],p[1],p[0]);

  complex z=0;/*початкове наближення до кореня - (0,0)*/
  do
  {
    complex dz=0;
  //знаходимо порядок і значення ненульової похідної в точці z
    for(j=1;dz==complex(0);dz=derive(p,j++)(z));
    z+=t*pow(-p(z)/dz,1/(-j));/*модифікуємо наближення */
    t*=(1-eps);
  }while(abs(p(z))>=eps);
  //повторюємо до досягнення заданої точності

  /*У цьому циклі знаходиться тільки один корінь z. Для виді-
лення інших поділимо вихідний поліном на x-z, знижуючи тим са-
мим степінь на одиницю, і знаходимо ще один корінь, і т.д. до
першого степеня */
  result[0]=z;
  cpolynom z1=2;
  z1[0]=-z,z1[1]=1;
  cvector more=newton(p/z1);
  for(long i=1;i<result.getm();i++)
    result[i]=more[i-1];

  //повертаємо вектор результату
  return result;
}
```

Визначення поліноміального класу завершує перший етап вивчення курсу чисельних методів – етап об’єктивізації наявного математичного знання.

§ 2.3. Активізація пізнавальної діяльності на етапі програмної реалізації чисельних методів

При побудові комплексного, векторного та поліноміального класів ми оперували лише тими відомостями з інформатики та вищої математики, що були вивчені у попередніх курсах. При цьому наявні математичні знання узагальнювались та об'єктивізувались у класах мови програмування. Така робота дозволяє сформувати навички побудови математичних об'єктів, застосовуючи лише вивчений раніше матеріал. Вказані класи, даючи зразок для подальших побудов, виконують не лише тренувальну функцію – вони широко застосовуються на другому етапі вивчення чисельних методів у об'єктній методології.

Як зазначалося у § 1.3, об'єктну тріаду обчислювальної математики складають математичний об'єкт, обчислювальний алгоритм та чисельна проблема. Відповідно до неї, об'єктивізація математичного знання та побудова математичних об'єктів становить значну частину об'єктної реалізації чисельних методів, проте не вичерпує її: для реалізації чисельних методів у об'єктній методології необхідно поряд з математичними ввести алгоритмічні та проблемні класи. Таким чином, вивчення кожного розділу чисельних методів вимагає побудови відповідного алгоритмічного класу (з використанням раніше визначених математичних) з подальшим тестуванням його методів у проблемному класі.

Після розгляду особливостей наближених обчислень вивчається другий розділ курсу – «Матриці та задачі лінійної алгебри». Матриці, матричні операції й обчислювальні процедури лінійної алгебри складають основу інженерного і наукового програмування, тому на початку цього розділу нагадуємо основні їх положення, вивчені у попередніх курсах, щоб вони були «під рукою» при розгляді чисельних методів лінійної алгебри [171, 172].

У цьому розділі розглядаються три основні класи задач:

- розв’язування систем лінійних алгебраїчних рівнянь (СЛАР), обчислення визначників, обернення матриць;
- обчислення власних значень і власних векторів матриць;
- метод найменших квадратів.

Усі ці задачі мають важливе прикладне значення при розв’язуванні різних проблем науки і техніки як самостійно, так і як допоміжні алгоритми в інших задачах обчислювальної математики, математичної фізики, опрацювання результатів експериментальних досліджень тощо [39].

Ідея формування базового набору процедур, що найбільш часто використовуються при реалізації методів лінійної алгебри, вперше була сформульована ще на початку 70-х рр. Пакет, що первісно був призначений для роботи з векторними об’єктами, отримав назву BLAS (Basic Linear Algebra Subprograms). До нього увійшли процедури, які реалізують наступні перетворення:

– скалярний добуток двох векторів;
– перетворення вигляду $y \leftarrow \alpha \cdot x + y$;
– копіювання елементів одного вектора в інший;
– перестановка місцями елементів двох векторів;
– множення вектора на скаляр;
– векторні норми (евклідова норма, сума абсолютних значень елементів та максимальне абсолютне значення);
– обертання Гівенса.

Легко переконатися, що побудований у попередньому розділі векторний клас реалізує всі перетворення пакету BLAS, за винятком останнього. Це дозволяє використати векторний клас у якості основи для побудови алгоритмічної бібліотеки лінійної алгебри та математичного класу «матриця».

Сучасні версії пакету BLAS значно розширені. У залежності від ви-

користовуваних алгоритмів в них виділяють процедури типу «вектор–вектор», «матриця–вектор», «матриця–матриця», об'єднаних загальними назвами BLAS1, BLAS2 та BLAS3. Сьогодні пакет BLAS вважається стандартним базовим засобом розробки чисельного програмного забезпечення, доступним у вигляді бібліотек алгоритмів та програм. Серед найбільш відомих бібліотек, в основу яких покладено BLAS, можна назвати PyMat – матричний пакет для мови програмування Python, JAMA – пакет елементарної лінійної алгебри для мови Java, MV++ – чисельні матрично-векторні класи мовою C++, SparseLib++ – бібліотеку для розріджених матричних обчислень, ISIS++ – об'єктно-орієнтоване середовище для розв'язування розріджених систем лінійних рівнянь. Питанням створення бібліотек лінійної алгебри присвячені роботи Р. Барретта [236], Т. Вельдхайзена [263–266], Г.В. Зеглінського [268], Дж.А. Макдональда [248], Р. Позо [254], В.Х. Преса [255], А.Д. Робінсона [256], Б. Страуструпа [259], К.С. Хорстманна [244] та інших дослідників.

Пакети та бібліотеки BLAS призначені для роботи з векторними та матричними об'єктами, поданими у деякому фіксованому форматі. Прикладом такої бібліотеки може бути розроблена нами бібліотека для роботи з матричними об'єктами, структуру якої наведено у Додатку В. Визначальною її особливістю є застосування механізму шаблонів, що дозволяє конструювати спеціалізовані матриці шляхом виклику відповідного шаблону із вказанням конкретних типів.

Як зазначають В.А. Семенов та О.А. Тарлапан, застосування техніки шаблонів при реалізації пакета BLAS має ціллю виключити використання віртуальних методів доступу до елементів векторних і матричних операндів і тим самим забезпечити максимально можливу ефективність генерованого коду [170]. Таким чином, побудований нами клас для роботи з матрицями задовольняє сформульованим вимогам до розробки математичного програмного забезпечення, разом з векторним класом реалізує операції па-

кету BLAS та є зручним інструментом програмування чисельних методів лінійної алгебри.

Матричний клас не є лише математичним класом – як можна побачити з Додатку В, його методи реалізують алгоритми лінійної алгебри, тому цей клас можна вважати також і алгоритмічним. Це дозволяє розглядати цей клас разом із вивченням відповідних чисельних методів.

Спочатку будується «кістяк» вектор-орієнтованого матричного класу, який складають конструктори, деструктор, операції додавання, віднімання, множення, обчислення визначника за методом Крамера, введення, виведення та порівняння матриць, знаходження мінорів та алгебраїчних доповнень, транспонування, обчислення натуральних степенів тощо. При переході до розгляду методів розв'язування систем лінійних алгебраїчних рівнянь та обчислення обернених матриць першим розглядається аналітичний *метод Крамера*, що базується на визначених раніше операціях обчислення визначника і алгебраїчних доповнень. Але метод Крамера не є ефективним – через велику кількість арифметичних операцій навіть для невеликих матриць (з порядком у кілька десятків) він займає забагато комп'ютерного часу, а для матриць порядку десятків і сотень тисяч стає зовсім неприйнятним. Тому в практичних обчисленнях звичайно використовують методи так званого псевдообернення – точні чи наближені.

Серед прямих методів найбільш простим і популярним при розв'язуванні систем порядку до 200–400 (у залежності від швидкодії використовуваного комп'ютера) є *метод Гаусса*, відомий також під назвою методу послідовного виключення змінних чи методу Гауссових виключень. Існує багато варіантів методу Гаусса, з них найбільш ефективний метод Жордана-Гаусса чи метод повного виключення – саме цей метод і реалізується у відповідних операціях матричного класу (обернення матриці, обчислення визначника та розв'язування СЛАР). Алгоритм Гауссових виключень можна використовувати і для факторизації матриць. Підкреслюю-

чи його універсальність, слід проте звернути увагу студентів на доцільність застосування його частинних методів (методу Холецького для симетричних позитивно визначених матриць та інших).

Основні операції, що використовуються у методі Гауссових виключень – множення вектор-рядка на скаляр та додавання вектор-рядків – є стандартними для пакету BLAS. Проте операція пошуку елемента в вектор-стовпці при виборі головного елемента не входить до складу цього пакету, і ця обставина дещо утруднює програмну реалізацію методу Гаусса у векторно-матричному середовищі.

Наступний метод, що розглядається у цьому розділі, базується на ортогональних перетвореннях і має назву *методу ортогоналізації*. Він реалізується за допомогою надзвичайно компактного алгоритму, не вимагає перевірок збіжності і скільки-небудь істотних перетворень вихідної системи, операція ділення на коефіцієнти матриці в ньому відсутня, а наявна операція ділення на норму вектор-рядка є набагато безпечнішою, тому що вектор нульової довжини не може бути присутнім у невиродженій матриці. Усе сказане обумовило його широке використання в прикладних задачах. У цьому методі використовуються лише векторні операції – знаходження модуля вектора та нормування за модулем, скалярний добуток двох векторів, множення вектора на число та додавання двох векторів. За рахунок використання операцій пакету BLAS1, реалізованого у векторному класі, програмна реалізація методу записується у надзвичайно компактній формі:

```
for(long l=1;l<f.m+1;l++)
{
  //ортогоналізуємо поточний вектор-рядок до усіх попередніх
  for(long i=0;i<l;i++)
    mtr2[l]-=mtr2[l]*mtr2[i]*mtr2[i];

  mtr2[l]=~mtr2[l]; //нормуємо поточний вектор-рядок
}
```

При розгляді сім'ї ітераційних методів – методу простих ітерацій та його модифікації, методу Зейделя – велика увага приділяється питанню збіжності ітераційного процесу. Під *збіжністю ітераційного процесу* розуміється наявність границі послідовності одержуваних розв'язків і рівність цієї границі при нескінченному числі ітерацій точному розв'язку системи. Для аналізу збіжності СЛАР приводять до виду

$$\boldsymbol{x}=\mathbf{A}\boldsymbol{x}+\boldsymbol{b}.$$

Ітераційний процес для такої лінійної системи збігається до єдиного розв'язку, якщо будь-яка канонічна норма матриці **A** менше 1, тобто для ітераційного процесу

$$\boldsymbol{x}_k=\boldsymbol{b}+\mathbf{A}\boldsymbol{x}_{k-1}\ (k=1, 2, ...)$$

достатня умова збіжності при довільному початковому наближенні $\boldsymbol{x}_0$ є:

$$||\mathbf{A}||<1.$$

У якості норми можна взяти максимальну з сум модулів елементів вектор-рядків (вектор-стовпців) або евклідову норму, яка реалізована у вигляді операції матричного класу. Таким чином, програмна реалізація методу ітерацій так само, як і методу ортогоналізації, може бути записана засобами BLAS також у надзвичайно компактній формі:

```
for(xn=2*xp;!(xn-xp)>eps;xp=xn)
  xn=A*xp+b;
```

Друга група методів, що розглядається у цьому розділі – це *методи обчислення власних значень та власних векторів матриць*. Проблема власних значень виникає в багатьох обчислювальних і дослідницьких задачах, наприклад, при дослідженні динаміки процесів у різних галузях – у техніці, біології, економіці тощо. Але розв'язування цієї проблеми пов'язане з істотними труднощами – досі не розроблені задовільні за точністю та ефективністю загальні методи, придатні для матриць загального виду, що приводить до нестійкості результатів обчислень до малих змін значень матричних елементів [154]. Існує багато спеціальних методів, призначених для матриць спеціальної структури – симетричних, стрічкових, квазідіагональ-

них та інших.

В усіх випадках, коли це виявляється можливим, намагаються із застосуванням *перетворень подібності*, які не змінюють власних значень матриці, привести матрицю або до трикутної форми (і уникнути процедур отримання і розв'язування характеристичного рівняння), або до форми, що дозволяє одержати коефіцієнти характеристичного полінома безпосередньо з перетвореної подібної матриці. Відомі методи таких приведень досить складні, їхнє обґрунтування і докладний виклад вимагає спеціального курсу з проблем власних значень, тому у нашому курсі ми обмежуємося коротким оглядом найбільш характерних підходів до вирішення цієї задачі.

Метод невизначених коефіцієнтів. Загальний вид характеристичного полінома відомий, якщо відомий порядок матриці – він являє собою скалярний добуток двох векторів; складовими одного є коефіцієнти при степенях λ, а складові другого – степені λ, які можна трактувати як «коефіцієнти при коефіцієнтах»:

$$P(\lambda)=c_n\lambda^n+c_{n-1}\lambda^{n-1}+\ldots+c_j\lambda+\ldots+c_1\lambda+c_0$$

Якщо задатися послідовністю значень λ_i і їхні степені підставити у вищенаведений запис, обчислити відповідну послідовність значень визначника D_i, то можемо скласти систему лінійних рівнянь відносно c:

$$P(\lambda_i)=D_i.$$

Розв'язання цієї системи дасть нам коефіцієнти характеристичного полінома і залишиться задача обчислення його коренів. Основний недолік методу – необхідність багаторазового обчислення значень визначника. Програмна реалізація цього методу використовує визначені в матричному класі операції обчислення детермінанту і розв'язування СЛАР та метод поліноміального класу для відшукання усіх коренів полінома, що дозволяє записати його у згорнутій формі, що відображає відповідний алгоритм.

Метод Данилевського ґрунтується на використанні перетворення по-

дібності $\mathbf{F}^{-1}\mathbf{AF}$ матриці **A**, де **F** – довільна матриця, до такої форми, з якої можна безпосередньо одержати коефіцієнти характеристичного полінома. У цьому методі вихідна матриця **A** приводиться до так званої канонічної форми Фробеніуса, верхній рядок якої містить значення коефіцієнтів характеристичного полінома.

Задача відшукання власних значень та власних векторів несиметричних матриць є більш важкою, ніж для симетричних матриць. Один з найбільш загальних методів її розв'язування – *QR-алгоритм*. Основна ідея цього методу полягає у розкладі вихідної матриці $\mathbf{A}_0=\mathbf{A}$ в добуток ортогональної матриці і верхньої трикутної. Послідовність перетворень дає чергові модифікації матриці **A**: $\mathbf{A}_k=\mathbf{Q}_k\mathbf{R}_k$, $\mathbf{A}_{k+1}=\mathbf{R}_k\mathbf{Q}_k$, $k=0, 1, 2, \ldots$, де кожна з матриць $\mathbf{Q}_k$ є ортогональною (якщо матриця **A** дійсна) чи унітарною (якщо **A** комплексна). $\mathbf{R}_k$ – верхні трикутні матриці. Перетворення можна виконати за відомим алгоритмом Хаусхолдера, для підвищення ефективності якого перед його застосуванням рекомендують привести матрицю до так званої форми Хессенберга; це приведення здійснюють із застосуванням перетворень Хаусхолдера. Після цього розкладання матриці виконується значно простіше за допомогою перетворень Хаусхолдера чи визначених у BLAS1 перетворень Гівенса (плоских обертань).

Серед класичних методів, крім рідко використовуваного на практиці методу безпосереднього розгортання вікового визначника, для обчислення найбільших за абсолютним значенням власних значень великих розріджених матриць іноді виявляється корисним *ітераційний степеневий метод*. Перевага методу – відсутність перетворень вихідної матриці, а головний недолік – повільна збіжність, особливо при близьких значеннях першого і другого за величиною власних значень. При кратному найбільшому власному значенні метод взагалі не збігається. Інша проблема виникає при необхідності обчислення наступних після найбільшого власних значень – кожне наступне буде обчислюватися з усе більшою похибкою.

Метод Крилова та метод *Левер'є-Фаддєєва* відносяться до прямих методів і відрізняються лише способами побудови характеристичного поліному, тому один з цих методів можна віддати на самостійне опрацювання.

Метод найменших квадратів (МНК) відноситься до методів аналітичного наближення функцій, заданих у табличній формі [3]. Його розміщення у матричному класі викликано тим, що він базується на конструюванні матриць і на матричних операціях – таке розміщення приводить до зменшення накладних витрат на виклик методу в порівнянні з розміщенням його в окремому класі.

У задачах моделювання при визначенні статичних характеристик об'єктів (залежності скалярної вихідної змінної y від векторної вхідної змінної $\boldsymbol{x}$) може стояти задача визначення вектора коефіцієнтів лінійної алгебраїчної моделі $\boldsymbol{a}$; у цьому випадку розв'язуванню підлягає система $\boldsymbol{y}=\mathbf{X}\boldsymbol{a}$, де $\mathbf{X}$ – матриця отриманих експериментально значень (чи значень деяких функцій від експериментальних даних) складових вектора входу, $\boldsymbol{y}$ – відповідні рядкам $\mathbf{X}$ значення складових вектора, $\boldsymbol{a}$ – вектор коефіцієнтів моделі, що підлягає визначенню, який треба обчислити так, щоб забезпечити найкраще наближення обчислених за моделлю значень $y_i(\boldsymbol{x}_i)$.

Термін «найкраще наближення» у МНК трактується як критерій Гаусса – мінімум суми Q квадратів відхилень обчислених (модельних) значень $\boldsymbol{y}_m$ від вимірюваних $\boldsymbol{y}$. Очевидно, що Q є функцією вектора $\boldsymbol{a}$:

$$Q(\boldsymbol{a})=\sum_{t=1}^{N}\left(y_t-\sum_i a_i x_{ti}\right)^2=(\boldsymbol{y}-\mathbf{X}\boldsymbol{a})^{\mathrm{T}}(\boldsymbol{y}-\mathbf{X}\boldsymbol{a})=\boldsymbol{y}^{\mathrm{T}}\boldsymbol{y}-2\boldsymbol{y}^{\mathrm{T}}\mathbf{X}\boldsymbol{a}+\boldsymbol{a}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{X}\boldsymbol{a}\to min.$$

Визначення значень аргументу, що забезпечують мінімум функції $Q(\boldsymbol{a})$, виконуємо засобами аналізу: $\partial Q(\boldsymbol{a})/\partial\boldsymbol{a}=-2\mathbf{X}^{\mathrm{T}}\boldsymbol{y}+2\mathbf{X}^{\mathrm{T}}\mathbf{X}\boldsymbol{a}=0$, чи

$$\mathbf{X}^{\mathrm{T}}\boldsymbol{y}=\mathbf{X}^{\mathrm{T}}\mathbf{X}\boldsymbol{a}.$$

Це звичайна система лінійних рівнянь із симетричною матрицею коефіцієнтів $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ (матриця Грама), стовпцем вільних членів $\mathbf{X}^{\mathrm{T}}\boldsymbol{y}$ і шуканим векто-

ром $\boldsymbol{a}$.

Розв'язування цієї системи лінійних рівнянь дає значення вектора $\boldsymbol{a}$, що забезпечує мінімум суми квадратів відхилень (у силу позитивності другої похідної), якщо ранг матриці спостережень $\mathbf{X}$ дорівнює розмірності вектора $\boldsymbol{a}$:

$$\boldsymbol{a}=(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\boldsymbol{y}.$$

Порядок N матриці $\mathbf{X}$ звичайно значно більше її рангу $N>(\mathrm{rank}(\mathbf{X})=m)$ (m – розмірність вектора $\boldsymbol{a}$), щоб була можливість виконання усереднення розв'язку по значному числу вимірів і відповідно ефективного «згладжування» випадкових викидів.

Розраховані коефіцієнти звичайно піддають перевірці на значимість за критерієм Стьюдента, незначущі коефіцієнти (відповідні їм члени моделі практично не впливають на значення вихідної змінної) бажано видалити з моделі. Отримані в результаті обчислення коефіцієнти математичної моделі вимагають перевірки на адекватність реальним даним. Ця перевірка виконується на окремій контрольній вибірці даних, що не використовувалась у розрахунках коефіцієнтів, по відношенню залишкової дисперсії y до загальної дисперсії; отримане відношення порівнюється з критеріальним, обумовленим розподілом ймовірностей Фішера.

Остання формула дозволяє записати розв'язання задачі визначення невідомого вектора $\boldsymbol{a}$ у один рядок:

```
matrix a=!(~x*x)*~x*y;
```

Таким чином, програмна реалізація методу найменших квадратів за рахунок використання операцій пакету BLAS та матричного класу набуває прозорості, безпосередньо відображає алгоритм та не викликає утруднень, стимулюючи до засвоєння подальшого матеріалу.

Традиційно задачі *лінійного програмування* в курсі чисельних методів розглядаються або у окремому розділі, або разом із задачами оптимізації. На наш погляд, доцільно розглядати їх разом з усіма іншими задачами

лінійної алгебри, адже лінійне програмування є однією з тих численних областей застосувань лінійної алгебри в прикладних задачах, що не піддаються розв'язанню методами класичного аналізу.

Основна задача лінійного програмування (ОЗЛП) виникає при кількості рівнянь, меншому за кількість незалежних змінних. У цьому випадку не залишається нічого іншого, як вибрати N змінних (за кількістю рівнянь) у якості базисних, а інші залишити вільними; розв'язків системи щодо такого базису нескінченна множина, якщо на них не накласти обмежень і не пред'явити додаткових вимог. В ОЗЛП вимогами, що забезпечують можливість одержання єдиного розв'язку, є вимога невід'ємності коренів $x_i \geq 0$ системи лінійних рівнянь

$$a_{11}x_1+a_{12}x_2+...+a_{1n}x_n=b_1$$
$$a_{21}x_1+a_{22}x_2+...+a_{2n}x_n=b_2$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$a_{m1}x_1+a_{m2}x_2+...+a_{mn}x_n=b_m$$

і необхідність забезпечення мінімуму деякої лінійної цільової функції

$$\min L=c_1x_1+c_2x_2+...+c_nx_n.$$

Система рівнянь може бути записана у вигляді матриці коефіцієнтів **A** та вектор-стовпця $\boldsymbol{b}$, а цільова функція – як скалярний добуток вектора коефіцієнтів $\boldsymbol{c}$ та вектора невідомих $\boldsymbol{x}$. Таким чином, ОЗЛП можна записати у векторно-матричній формі, а процес її розв'язування у вигляді операцій відповідних класів.

Подання функції у вигляді дискретної послідовності значень аргументу (так званих вузлів) і відповідних значень функції у вузлових точках – звичайне і природне її подання при опрацюванні даних на цифровій обчислювальній машині, тому третій розділ нашого курсу присвячено аналітичному наближенню функцій, заданих у табличній формі. Наприклад, підключені до машини цифрові перетворювачі вимірювальних пристроїв поставляють послідовність значень параметра (температури, тиску, витрати,

відстані, швидкості тощо) у дискретні моменти часу. Ми можемо використовувати також і дискретні подання аналітичних функцій досить складної структури. Якщо при цьому виникає потреба у виконанні таких операцій, як інтегрування, диференціювання функції чи просто необхідність оцінити її значення в невузлових точках, то бажання здійснити аналітичну заміну дискретної послідовності теж виглядає досить природним.

Для спрощення будемо вважати спочатку, що значення функції задані при рівновіддалених один від одного значеннях аргументу, тобто, якщо не обговорене альтернативне подання, будемо розглядати задачу наближення при рівновіддалених вузлах; це дає можливість подати аргумент функції у вигляді, наприклад, послідовності цілих чисел з обраним постійним кроком між ними.

Постановка задачі аналітичного наближення (*апроксимації*) функцій і методи її розв'язування визначаються наступними основними факторами, які необхідно проаналізувати на самому початку:

1) У нашому розпорядженні числовий ряд значень функції, отриманий на деякому інтервалі значень аргументу – назвемо його інтервалом спостереження. Якщо аналітична заміна цього ряду є необхідною для наступної роботи усередині інтервалу спостереження, то відповідна задача називається *задачею інтерполяції*. Може виявитися, що нас цікавлять оцінки значень функції правіше інтервалу спостереження, тобто в тій області значень аргументу, де значення функції не задані, тоді говорять про *задачу екстраполяції чи прогнозування*. Нарешті, може стояти задача отримання достовірної оцінки значення функції безпосередньо на правій межі інтервалу чи спостереження в будь-якому його вузлі – ця задача виникає при наявності істотних помилок у визначенні значень функції (так званих шумів у спостереженнях) і називається *задачею фільтрації чи згладжування*. Таким чином, задача аналітичного наближення табличних функцій – це комплексна задача інтерполяції, екстраполяції і згладжування. Її «спрямо-

ваність» значною мірою визначає методи її розв'язування: коли аргументом функції є час, можна говорити про спрямованість у минуле, майбутнє чи оцінках сьогодення.

2) Другий істотний аспект проблеми полягає у визначенні, що називати «гарним» чи «найкращим» наближенням, тобто у виборі *критерію оптимальності наближення*. Класичні методи наближення функцій використовують як критерій оптимальності наближення вимогу точного збігу значень функції, що наближає, з табличними значеннями у вузлах (наближення за критерієм Лагранжа). Перевага такого підходу – у простоті теорії наближення й обчислювальних процедур; недолік – в ігноруванні неминучої в реальних умовах наявності шумів у спостереженнях. Загальна задача інтерполяції, згладжування й екстраполяції була вперше математично сформульована А.М. Колмогоровим, на початку другої світової війни нею займався Норберт Вінер у застосуванні до керування зенітним вогнем, пов'язаним з необхідністю прогнозувати координати мішені на час польоту снаряда. Задача спостереження за мішенню завжди пов'язана з її непередбаченими маневрами і помилками спостереження, тому згладжування і попередження траєкторії мішені – актуальна задача, що не вписується в класичну постановку з точним відстеженням значень у вузлах. Надалі необхідність у згладжуванні і попередженні при різних умовах і різних вимогах до якості й області припустимих прогнозів виникла в різних задачах економіки, метеорології, автоматичному регулюванні та ін. При необхідності врахування помилок спостережень найчастіше використовують класичний критерій мінімуму суми квадратів різниць між розрахованими за функцією, що наближає, та спостереженими значеннями у вузлах – *критерій Гаусса* і його сучасні модифікації. Інший критерій пов'язують з ім'ям Чебишова; він полягає у вимозі мінімізувати максимальну різницю між спостереженими і розрахованими значеннями функції у вузлах – так званий *мінімаксний критерій*. Існують і інші підходи до оцінки якості набли-

ження.

3) Третій ключовий момент полягає у виборі класу аналітичних функцій, що наближають. Основна вимога, що ставиться до цих функцій – це незалежність результатів апроксимації від початку відліку, тобто від зсуву по послідовності значень аргументу. Іншими словами, необхідно, щоб скінченна множина функцій обраного для апроксимації класу переходила сама в себе при заміні x на $x+k$. Таку властивість мають:

- лінійні комбінації степеневих функцій $1, x, x^2, \ldots, x^n$;
- тригонометричні функції $\cos(a_i x)$, $\sin(a_i x)$;
- експонентні функції виду $\exp(-a_i z)$.

Використання будь-якої іншої скінченної множини апроксимуючих функцій (крім трьох перерахованих) передбачає наявність природного початку відліку, тому що вибір початку вплине на результат, але в більшості задач немає природного початку і ми змушені вибирати з перерахованих класів функцій.

Відповідно до перерахованих класів розрізняють *поліноміальну* апроксимацію, Фур'є чи *тригонометричну* апроксимацію й *експонентну* апроксимацію. Ще одна важлива властивість, яку доцільно було б надати множині апроксимуючих функцій – це інваріантість до зміни масштабу, тобто щоб множина не змінювалася при заміні x на kx. З розглянутих множин такою властивістю володіє лише множина $1, x, x^2, \ldots, x^n$, тому за відсутності в задачі природного масштабу ми змушені або вибирати поліноми, або впливати на результат вольовим вибором масштабу. Правда, більшість практичних задач, зокрема, пов'язаних з функціями часу, мають природний масштаб і перевага поліноміальної апроксимації над іншими видами полягає у більшій «просунутості» поліноміальної алгебри і простотою обчислювальних процедур з поліномами.

Звичайно у якості апроксимуючої функції вибирають лінійну комбінацію функцій з розглянутих класів вигляду

$$t(x)=\sum_{i=0}^{n} c_i\varphi_i(x)$$

з дійсними коефіцієнтами c_i. Якщо такий узагальнений поліном сформовано і критерій оптимальності наближення обрано, то можна скласти і розв'язати систему алгебраїчних рівнянь, лінійних щодо коефіцієнтів c_i:

$$\sum_{i=0}^{n} c_i\varphi_i(x_k)=f(x_k),$$

де k=0, 1, 2, … – порядковий номер вузла, $f(x_k)$ – задане табличне значення функції в k-ому вузлі.

Очевидно, що кількість таких рівнянь повинна бути не менше n+1, а спосіб розв'язування залежить від критерію наближення.

Якщо потрібно точне проведення апроксимуючої функції через вузлові точки, тобто розглядається задача інтерполяції в постановці Лагранжа, то $k=n+1$, а на систему функцій $\varphi_i(x)$ накладається наступна вимога: для того, щоб для заданої функції $f(x)$, визначеної на відрізку $[a, b]$, і набору n+1 вузлів $x_0, x_1, \ldots, x_n$ ($x\subset[a, b]$) існував і був єдиним узагальнений інтерполяційний поліном $t(x)=\sum_{i=0}^{n} c_i\varphi_i(x)$, необхідно і достатньо, щоб будь-який узагальнений поліном (з хоча б одним ненульовим коефіцієнтом) по обраній системі функцій $\varphi_i(x)$ мав на відрізку $[a, b]$ не більше n коренів.

У нашому курсі розглянуті лише основні методи алгебраїчної апроксимації в розрізі задач інтерполяції функцій і зглажування за методом найменших квадратів: класичний інтерполяційний поліном, інтерполяційний поліном Лагранжа, інтерполяційний поліном Ньютона, інтерполяція сплайнами та апроксимація функцій за методом найменших квадратів [47]. Проблемний клас апроксимуючих функцій повинен, очевидно, містити як члени-дані два вектори однакової розмірності: 1) для збереження послідовності вузлів і 2) для збереження значень заданої функції в цих вузлах. Набір функцій-членів, крім обов'язкового набору конструкторів, повинен включати методи реалізації розглянутих алгоритмів і, можливо, підпрог-

рами для інтерфейсу користувача. Нижче наведено зміст інтерфейсного файлу з визначенням класу апроксимацій, що містить зазначені компоненти із супровідними коментарями.

```
#include "vector.h" //інтерфейсний файл векторного класу
#include "polynom.h" //інтерфейсний файл поліноміального класу
#include "matrix.h" //інтерфейсний файл матричного класу

/*Клас поліноміальних апроксимацій */
template <class Type> class Approximate
{
  /*Для розв'язання задачі апроксимації нам необхідні масиви
вузлів і значень функції у вузлах, тобто об'єкти класу vector
*/
  vector<Type> x, y;/* Вузли і значення у вузлах*/

 public:/* Загальнодоступні методи */

/*Основний конструктор як параметр приймає два вектори - век-
тор вузлів і вектор значень у вузлах */
  Approximate(vector<Type> _x, vector <Type> _y): x(_x), y(_y)
  {
    if(x.getm()!=y.getm()) /*Перевірка розмірностей на коррект-
ність*/
      throw xmsg("Кількість вузлів не збігається з кількістю
значень у них");
    /*Напевно, у цьому випадку задача апроксимації втрачає
зміст:*/
    if(x.getm()<2) throw xmsg("Занадто мало точок");
  }

  /*Конструктор копіювання */
  Approximate(Approximate &_): x(_.x), y(_.y) {} /*Результатом
рішення повинен бути, природно, об'єкт класу polynom */
```

```
  //Прототипи методів класу апроксимацій
  polynom<Type>  classic(),//Інтерполяція канонічним поліномом
                 newton(); //Інтерполяція поліномом Ньютона

  polynom<Type>* spline(); /*Інтерполяція кубічними сплайнами.
Значення, що повертається – динамічний масив поліномів. По за-
кінченню роботи пам'ять з під нього необхідно звільнити */

  polynom<Type>  MNK(int); /*Апроксимація функції однієї змін-
ної за методом найменших квадратів при степеневому базисі. Ар-
гумент - порядок полінома */
};
```

Результатом роботи методів апроксимуючого класу є об’єкт поліноміального класу – апроксимуючий поліном, який будується за допомогою методів матричного та векторного класів:

```
//Інтерполяція канонічним поліномом
template <class Type> polynom<Type>
Approximate<Type>::classic()
{
//Матриці для збереження лівої і правої частин системи рівнянь
  matrix<Type> mtr(x.getm(),x.getm()), f(x.getm(),1);

  for(int i=0;i<x.getm();i++) //Формування матриці СЛАР
  {
    /*Права частина системи - значення функції у вузлах */
    f[i][0]=y[i];
    /*Ліва частина системи - вектор степенів вузлів */
    for(int j=0;j<x.getm();j++)
      mtr[i][j]=pow(x[i],j);
  }

 //Викликаємо метод Гаусса з матричного класу для рішення СЛАР
  matrix<Type> sol=SLAE_Gauss(mtr,f);
```

```
//Результат рішення побудованої системи - коефіцієнти полінома
  polynom<Type> res=x.getm();
  for(int i=0;i<x.getm();i++)//формуємо поліном з коефіцієнтів
    res[i]=sol[i][0];

  /*Повертаємо результат – апроксимуючий поліном*/
  return res;
}
```

Четвертий розділ курсу присвячено методам чисельного інтегрування та диференціювання. Класичні інтерполяційні поліноми, вивчені в попередньому розділі, відіграють фундаментальну роль у підходах, що застосовуються при чисельному інтегруванні функцій. Як зазначають В.А. Семенов та С.В. Морозов, «… роль поліномів не обмежується лише теоретичними побудовами і відповідними обґрунтуваннями. Вони також використовуються у якості практичних методик для конструювання квадратурних формул, організації алгоритмів чисельного інтегрування, отримання необхідних оцінок обчислювальних похибок» [165, с. 120–121]. Дійсно, будь-який наближуючий поліном, для якого визначені процедури інтерполяції чи апроксимації, породжує цілу сім'ю квадратурних методів.

Задача чисельного інтегрування може бути сформульована таким чином: обчислити визначений інтеграл $\int_a^b f(x)dx$ за значеннями функції f_k та її похідних $f_k^{(m)}$ в точках $x_k \in [a, b]$, $k=1, 2, \ldots, n$ із заданою точністю в межах відведених ресурсів (як правило, використовуються лише значення самої функції).

Існує два різні підходи до організації методів чисельного інтегрування. Перший базується на застосуванні квадратурних формул вигляду

$$\int_a^b f(x)dx = \sum_{k=1}^{n} w_k f(x_k),$$

де w_k – вагові коефіцієнти, а x_k – вузли квадратури. Для виведення таких

формул застосовується процедура інтерполяції підінтегральної функції одним з класичним поліномів (звичайно поліномом Лагранжа), первісна якого виражається аналітично.

Другий підхід полягає в безпосередній побудові полінома $\varphi(x)$, що інтерполює підінтегральну функцію: $\varphi(x_k)=f(x_k)$, та обчисленні інтеграла за формулою

$$\int_a^b f(x)dx = \int_a^b \varphi(x)dx = \Phi(b)-\Phi(a),$$

де $\Phi(x)$ – первісна $\varphi(x)$. Цей підхід часто використовується в геометричних застосуваннях, в яких підінтегральна функція подається чи наближується сплайнами різних видів.

Поліноміальний клас, описаний у § 2.3 дослідження, містить методи обчислення аналітичної похідної та інтегралу заданого порядку (кратності) і обчислення значення полінома в заданій точці. Застосування цих методів дозволяє звести задачу інтегрування до задачі інтерполювання, розглянутої у третьому розділі курсу, та до задачі знаходження функціонального значення поліному, розглянутої у першому розділі. Подібне використання внутрішньопредметних зв’язків, постійне повертання до раніше вивченого матеріалу та активне використання у наступних розділах курсу чисельних методів, вивчених у попередніх, відповідає принципу циклічності (за А.П. Єршовим [64]).

Четвертий розділ курсу так само, як і перший, не вимагає засвоєння нових знань, тут достатньо лише актуалізації попередніх, на відміну від другого і третього та п’ятого і шостого розділів. Така побудова курсу дозволяє на початку вивчення чисельних методів створити однакові умови для студентів з різними рівнями пізнавальної активності, а в середині – дещо «пригальмувати», звернувши більшу увагу на застосування вивчених методів. Цей своєрідний «перепочинок» дозволяє підготувати студентів до засвоєння одного з найважливіших розділів курсу – чисельних методів

розв’язування диференціальних рівнянь.

Використовувані на практиці методи чисельного інтегрування можна класифікувати за способами апроксимації підінтегральної функції:

Методи Ньютона-Котеса базуються на поліноміальній апроксимації і відрізняються один від одного степенем апроксимуючого полінома. Ці алгоритми прості і легко програмуються.

Сплайнові методи базуються на сплайновій апроксимації підінтегральної функції, розрізняються за типом обраних сплайнів і застосовуються в задачах обробки даних з використанням сплайнів.

Методи найвищої алгебраїчної точності (Гаусса-Кристофеля й ін.) застосовні в основному до аналітичних функцій, використовують нерівновіддалені вузли, розташовані по алгоритму, що забезпечує мінімальну похибку інтегрування для найбільш складних функцій при заданій кількості вузлів.

Методи Монте-Карло використовують випадкове розташування вузлів і дають результат імовірнісного змісту.

Незалежно від методу, у процесі чисельного інтегрування необхідно обчислити наближене значення інтеграла й оцінити похибку, що залежить від числа ділянок поділу інтервалу інтегрування – зменшується з ростом числа ділянок за рахунок більш точної апроксимації й одночасно росте за рахунок похибки підсумовування часткових інтегралів. Ця складова починає з деякого значення N переважати, що перешкоджає надмірному подрібненню інтервалу інтегрування.

В сімействі методів Ньютона-Котеса звичайно виділяють такі методи:

Методи прямокутників – це найпростіші з класу Ньютона-Котеса методи, що ґрунтуються на апроксимації підінтегральної функції поліномом нульового степеня – константою на заданому відрізку інтервалу. Для такої апроксимації досить однієї точки – будь-якого значення підінтегра-

льної функції в будь-якому вузлі; це значення вважається постійним на всьому проміжку між сусідніми вузлами.

Метод трапецій утворюється при заміні підінтегральної функції поліномом першого степеня $P_1(x)$, для чого доводиться використовувати обидві межі часткового інтервалу. На кожному елементарному відрізку аргументу x ділянка кривої інтегрування являє собою відрізок прямої – дві ординати і відрізок вісі абсцис разом з цієї прямою обмежують фігуру трапецевидної форми, що і дає назву цьому методу кусочно-лінійної апроксимації підінтегральної функції. Наближене значення інтеграла визначається площею трапеції.

Метод Сімпсона базується на заміні підінтегральної функції поліномом другого степеня, інтеграл від якого називають квадратурною формулою Сімпсона.

Як зазначалося вище, програмна реалізація методів чисельного інтегрування базується безпосередньо на поліноміальній арифметиці і не викликає жодних утруднень. Інтерфейсна частина проблемного класу, що реалізує методи інтегрування, може мати такий вигляд:

```
#include "vector.h"  //Інтерфейсний файл векторного класу
#include "polynom.h" //Інтерфейсний файл поліноміального класу

/*Параметризований клас для обчислення інтегральних сум */
template <class Type> class Integrate
{
  vector<Type> x, y; /*Вузли і значення у вузлах*/

 public:
  /*Конструктор як параметр приймає два вектори, що містять
відповідно вузли і значення в них */
  Integrate(vector<Type> _x, vector <Type> _y): x(_x), y(_y)
  {
    /*Перевіряємо вхідні дані на коректність */
    if(x.getm()!=y.getm())
```

```
      throw xmsg("Кількість вузлів не збігається з кількістю
значень у них");
    if(x.getm()<2) throw xmsg("Занадто мало точок");
  }

  Integrate(Integrate &_): x(_.x), y(_.y) {} /*Конструктор ко-
піювання */

  //Прототипи методів чисельного інтегрування
  Type rectangle_method(),    //Метод прямокутників
       trapecion_method(),    //Метод трапецій
       simpson_method(),      //Метод Сімпсона
       monte_karlo_method();  //Метод Монте-Карло
};
```

При необхідності обчислити похідну табличної функції в тій чи іншій точці (у вузлі чи міжвузловому проміжку) може здатися, що для цього досить виконати аналітичну заміну легко диференційованою функцією (наприклад, поліномом), продиференціювати отриману апроксимуючу функцію та обчислити значення функції-похідної при заданому значенні аргументу. Але при цьому виникає проблема «хвилястості» аналітичної заміни через занадто високий степінь інтерполяційного полінома, побудованого по великій кількості вузлів чи по вузлах із занадто великим кроком. Це, як правило, приводить до того, що навіть перша похідна в заданій точці для вихідної й апроксимуючої функцій будуть відрізнятися дуже сильно, аж до розбіжності їхніх знаків.

Диференціювання табличних функцій є некоректною задачею, тому що похибка обчислення похідних може у багато разів перевищувати похибку інтерполяції самої функції. Тому при чисельному диференціюванні табличних функцій слід застосовувати прийоми зменшення потенційних похибок диференціювання.

До таких прийомів відносяться методи фільтрації апроксимуючої

функції – видалення з неї високих частот незначної амплітуди, близьких до припустимої похибки апроксимації. До такого ефекту усереднення приводить використання методу найменших квадратів зі степенем апроксимуючого полінома, значно нижчим за кількість використовуваних вузлів. При використанні як базисних функцій ортогональних поліномів Чебишова можна поступово додавати члени з більш високими степенями з обчисленням для кожного степеня величини залишкової дисперсії – коли вона зменшиться до задовільного значення чи темп її зменшення уповільниться, нарощування степеня полінома варто зупинити. Попутно можна обчислювати похідну в заданій точці і якщо її значення піддається при нарощуванні степеня полінома значним змінам, до отриманого результату слід відноситися з недовірою.

Інший спосіб фільтрації високих частот полягає в тому, що отриману тим чи іншим способом апроксимуючу функцію пропускають через інерційну ланку, використовуючи апроксимуючу функцію як праву частину неоднорідного лінійного диференціального рівняння й одержують в результаті розв'язування рівняння функції, використовуваної для наступного диференціювання. Розв'язування цієї задачі розглядається в п'ятому розділі курсу – «Вступ до чисельних методів розв'язування диференціальних рівнянь».

Звичайним диференціальним рівнянням для функції $f(t)$ називається рівняння виду

$$F(t, f(t), f^{(1)}(t), \ldots, f^{(n)}(t))=0,$$

де F – задана функція для нескінченного чи скінченого інтервалу t. *Порядок рівняння* визначається порядком n старшої похідної $f^{(n)}(t)$ у цьому рівнянні. Рівняння називають *лінійним*, якщо функція F лінійно залежить від усіх своїх аргументів:

$$F(t)=f^{(n)}(t)+a_{n-1}f^{(n-1)}(t)+\ldots+a_1 f(t)+a_0,$$

де a_0, a_1, …, a_{n-1} – або задані постійні коефіцієнти, або задані функції t. Це

рівняння можна розглядати й у векторному варіанті, коли $\boldsymbol{f}$ і $\boldsymbol{F}$ є вектор-функціями і ми маємо справу не з одним рівнянням, а з системою рівнянь.

В [261] подано перелік та коротку характеристику основних математичних об'єктно-орієнтованих бібліотек, що реалізують різні методи інтегрування диференціальних рівнянь, такі як ODE++ – бібліотека класів для звичайних диференціальних рівнянь з інтеграторами різних типів, Godess – бібліотека для розв'язування звичайних диференціальних рівнянь та рівнянь у частинних похідних, PETSc та Diffpack – об'єктно-орієнтовані бібліотеки для рівнянь у частинних похідних.

Більшість методів розв'язування диференціальних рівнянь призначені для розв'язування системи рівнянь першого порядку

$$\boldsymbol{f}^{(1)}(t)=\boldsymbol{F}(t,\boldsymbol{f}(t)),$$

де $\boldsymbol{f}$ і $\boldsymbol{F}$ – n-вимірні вектори з координатами $F_1, F_2, \ldots, F_n, f_1, f_2, \ldots, f_n$, оскільки рівняння n-го порядку зводиться до системи n рівнянь першого порядку, а систему m рівнянь n-го порядку можна звести до системи nm рівнянь першого порядку підстановкою *методом заміни змінних*:

$$f_i(t)=f^{(i-1)}(t),\ i=1,\ \ldots,\ n.$$

У випадку лінійності системи рівнянь відносно $\boldsymbol{f}(t)$ вона набуває вигляду

$$\boldsymbol{f}^{(1)}(t)+\mathbf{A}\boldsymbol{f}(t)=\boldsymbol{b}(t),$$

де $\mathbf{A}$ – задана матриця розмірності nxn, $\boldsymbol{b}$ – задана вектор-функція від t.

Якщо елементи матриці $\mathbf{A}$ не залежать від t і $\boldsymbol{b}=0$, то ми приходимо до *лінійної однорідної системи зі сталими коефіцієнтами*:

$$\boldsymbol{f}^{(1)}(t)+\mathbf{A}\boldsymbol{f}(t)=0,$$

розв'язок якої можна одержати явно у вигляді розкладу

$$\boldsymbol{f}(t)=\boldsymbol{c}\sum_{n=0}^{\infty}\frac{\mathbf{A}^n t^n}{n!},$$

де $\boldsymbol{c}$ – довільний n-вимірний сталий вектор. Розклад у дужках є експонента з показником $\mathbf{A}t$ і розв'язок можна записати в компактній формі

$$\boldsymbol{f}(t)=e^{\mathbf{A}t}\boldsymbol{c}.$$

Оскільки загальний розв'язок системи залежить від n довільних сталих c_i, то для визначення конкретного єдиного розв'язку необхідно задати n додаткових умов, що звичайно являють собою початкові умови виду $\boldsymbol{f}(0)=\boldsymbol{f}_0$ чи граничні умови виду $\boldsymbol{f}(a)=\boldsymbol{f}_a$ і, наприклад, розв'язком системи однорідних рівнянь першого порядку з постійними коефіцієнтами для заданих початкових умов буде

$$\boldsymbol{f}(t)=e^{\mathbf{A}t}\boldsymbol{f}_0.$$

Таким чином, задачу побудови розв'язку системи диференціальних рівнянь першого порядку можна сформулювати у термінах BLAS, тобто із застосуванням векторної, матричної та поліноміальної арифметики.

При розгляді цього розділу доцільно обговорити, звідки виникають диференціальні рівняння і кому необхідні їхні розв'язки. Це питання не виникає при підготовці фахівців інженерного профілю, адже вивчення прикладної галузі неминуче супроводжується математичним описом динаміки досліджуваних у ній процесів і диференціальні рівняння сприймаються як природний інструмент дослідження. Інший стан справ при підготовці фахівців з математики, що мають педагогічну спрямованість – вивчення аналітичних і чисельних методів розв'язування систем алгебраїчних, трансцендентних і диференціальних рівнянь часто сприймається просто як «вправи для розуму» й у студента найчастіше залишаються дуже нечіткі уявлення про прикладне значення курсу чисельних методів.

Якщо шкільні задачі, як правило, включають змістову частину типу наповнюваних рідинами посудин чи автомобілів, що рухаються в різних напрямках, то студент-математик найчастіше просто одержує завдання розв'язати конкретну систему рівнянь, не уявляючи, навіщо і кому це може знадобитися (крім необхідності одержати оцінку). Саме тому на етапі формуючого експерименту виникла ще одна задача – підвищити практичну значущість курсу чисельних методів шляхом введення у нього в якості

прикладної частини задач з теорії керування. В розв'язуванні цієї задачі був використаний досвід, накопичений автором в процесі викладання курсів, суміжних з обговорюваним (курсу автоматики в Криворізькому державному педагогічному університеті та теорії автоматичного керування в Криворізькому технічному університеті). Результати цієї роботи були пізніше узагальнені в навчальному посібнику для студентів педвузів [141].

Така побудова розділу потребує розгляду деяких додаткових відомостей з теорії керування:

Під *процесом* розуміють зміну деякої величини в часі і просторі – це може бути зміна хімічного складу речовини в хімічному реакторі, зміна прибутку підприємства, зміна координат рухомої ракети, зміна чисельності визначеного виду тварин у деякому регіоні тощо. Процес починається і протікає завдяки зовнішнім впливам на реалізуючий його об'єкт. Ці впливи можуть бути цілеспрямованими (*керуючими*), прикладеними для забезпечення бажаного характеру протікання процесу – зміна подачі реагентів у хімічний реактор, зміна обсягу використовуваних оборотних коштів підприємства, зміна тяги ракетного двигуна чи положення рулів літака, зміна факторів, що впливають на приріст чисельності тварин тощо.

Інший тип впливів на процес називають *збурюваннями* (небажаними впливами) – коливання сили вітру, що відхиляє літальний апарат від розрахункового курсу, коливання попиту на продукцію підприємства та інші.

Управління процесами полягає у визначенні і наступній реалізації таких керуючих впливів, що забезпечили б бажаний чи, за можливості, близький до нього плин процесу, що піддається дії небажаних, часто непередбачених і неконтрольованих збурень. Але для визначення необхідних керуючих впливів необхідно знати, як процес реагує на ці впливи – іншими словами, для обчислення керувань необхідно знати їхній взаємозв'язок з *керованими (вихідними) величинами* і цей взаємозв'язок повинен бути виражений у вигляді математичних співвідношень, що називають *матема-*

тичною моделлю процесу. Таким чином, *математичне моделювання процесів* є невід'ємною складовою частиною процесу керування.

Із загальною задачею керування пов'язані три основні її складові:

1) *Задача ідентифікації* математичної моделі керованого процесу виникає, якщо математичні співвідношення, що пов'язують входи і виходи процесу невідомі. Ці співвідношення можуть бути визначені теоретично з використанням основних законів відповідної прикладної області чи експериментально.

2) *Задача аналізу.* Якщо математичну модель процесу ідентифіковано, вона дозволяє здійснити імітаційні дослідження процесу – задаючись різними функціями керування, можна, розв'язуючи диференціальні рівняння, що описують процес, одержати функції, що описують реакцію процесу на ці керування. Ця задача є допоміжною для розв'язання основної задачі – задачі керування.

Задача аналізу може розглядатися в різних постановках. Якщо задані значення вихідної величини процесу і всіх її похідних до (n–1)-ої включно у деякий момент часу, прийнятий початковим, при відсутності керуючого впливу – це *задача визначення вільного руху процесу*. Вона розглядається на напівнескінченному інтервалі часу і має сенс, якщо хоча б одна з n–1 похідних не дорівнює нулю – у противному разі рух відсутній і вихід процесу постійний. Ця ж задача може розглядатися і при наявності керуючого впливу – у цьому випадку розв'язок диференціального рівняння являє собою комбінацію з власного руху і змушеного під дією керування. Обидві підгрупи задач з початковими умовами звуться *задачами Коші для диференціальних рівнянь*.

Якщо процес аналізується на обмеженому інтервалі часу і частина умов задані на лівій межі інтервалу, а частина на правій (загальна кількість додаткових умов повинна дорівнювати порядку рівняння), ми маємо справу з так званою *граничною задачею* знаходження такого розв'язку, що за-

довольняє заданим умовам на обох межах.

3) *Задача керування.* У задачі керування ставиться задача знайти керування, що переводить процес із довільного (відомого) стану, що характеризується значеннями вихідної функції та її похідних, у заданий кінцевий стан, теж заданий значеннями вихідної функції та її похідних. Очевидно, що при надлишковій кількості додаткових умов для одержання єдиного розв'язку необхідно визначити, яке з безлічі можливих керувань треба вважати кращим, ніж інші. У цій постановці ми приходимо до класу *задач оптимального керування*. Наприклад, при керуванні рухом ракети можна в одних випадках вважати найкращою таку керуючу функцію, що забезпечить виведення ракети в задану точку з найменшою кількістю витраченого палива (керування, оптимальне за витратами ресурсів); в інших випадках найкращим може вважатися керування, що виводить ракету в задану точку за найкоротший час (керування, оптимальне за швидкодією).

Керуючий пристрій і керований об'єкт разом являють собою систему, описувану загальною системою диференціальних рівнянь. Параметри об'єкта (яким відповідають коефіцієнти рівняння руху), як правило, не піддаються цілеспрямованій зміні в процесі керування – вони визначаються його конструкцією і можуть самі дрейфувати в процесі старіння об'єкта під час експлуатації. Але параметри іншої складової системи – керуючого пристрою – можуть змінюватися і ця обставина призвела до розробки способу керування, заснованого на зміні структури системи в процесі її руху – це так звані *системи із змінною структурою*.

Розв'язування комплексної задачі ідентифікації, аналізу та керування динамічними системами дозволяє збагатити цей розділ методами знаходження розв'язків лінійних диференціальних рівнянь зі змінною правою частиною, зокрема – методами операційного числення. В результаті перетворень Лапласа чи Карсона функція $f(t)$ дійсного змінного t перетвориться у функцію $F(p)$ комплексного аргументу p, а лінійне диференціальне рів-

няння зі сталими коефіцієнтами – у відповідне алгебраїчне рівняння, ліва частина якого є комплексним поліномом чи раціональним поліноміальним дробом. Наявність визначених у першому розділі курсу математичних класів дозволяє у зручний спосіб створити автоматичний інтегратор, що базується на методах операційного числення [253].

Клас диференціальних рівнянь повинен мати такі дані:

1) int rngl, rngr – порядок рівняння (лівої та правої частин);
2) dvector nv – вектор початкових умов;
3) double tend, tstep – кінцевий час розв'язування та крок дискретизації;
4) int PointSolCnt – розмірність вектора розв'язку;
5) cpolynom PL, PR, PN – поліноми лівої, правої частини та початкових умов;
6) cpolynom PSOL, QSOL – знаменник і чисельник зображення розв'язку;
7) CA *R – покажчик на структуру із даними про корені:
   – complex ROOT – значення кореня;
   – complex A – коефіцієнт для кореня в оригіналі;
   – int J – кратність кореня;
   – int O – кому кратний.
8) long RootCount – загальна кількість коренів (за наявності правої частини рівняння вона не збігається з його порядком ліворуч);
9) long rcount – кількість різних коренів (за винятком кратних).

Методи класу диференціальних рівнянь:

1) void GetRoot() – функція для обчислення коренів характеристичного полінома;
2) cpolynom Qsol(int der) – функція формування чисельника зображення похідної функції рішення;
3) void GetCoeffOrigin() – функція, що обчислює коефіцієнти оригіналу для всіх коренів;
4) double GetValue(double t) – функція, що повертає значення виходу при

заданому значенні аргументу;

5) void Sol() – функція обчислення розв'язку диференціального рівняння;
6) void FreqChar() – функція для обчислення частотних характеристик диференціального рівняння.

Застосування цього класу дозволяє виконати комп'ютерне моделювання лінійних динамічних систем у значно ширшому обсязі, ніж це пропонується у більшості робіт (зокрема, [22]). Створене нами програмне забезпечення для комплексного аналізу динаміки лінійних систем, що являє собою об'єктно-орієнтоване графічне середовище, дозволяє аналізувати системи із стандартизованими та довільними збуреннями в правій частині. При цьому забезпечується виведення графіків руху (вільного, під впливом імпульсного, східчастого, синусоїдального, косинусоїдального, експоненціального та довільного збурень), частотних характеристик (дійсної, уявної, амплітудної, фазової та амплітудно-фазової), фазових портретів тощо.

Розгляд операторного методу у курсі чисельних методів не є традиційний, адже цей метод є аналітичний, і лише окремі компоненти його програмної реалізації вимагають застосування чисельних методів відшукання поліноміальних нулів. Проте вивчення цього методу дозволяє, з одного боку, розглянути ряд важливих застосувань диференціальних рівнянь, а з іншого, надає можливість порівняння результатів роботи аналітичного та чисельного інтеграторів.

Чисельні методи розв'язування звичайних диференціальних рівнянь, на відміну від аналітичного операторного, не вимагають лінійності розв'язуваного рівняння. Однокрокові (метод Ейлера, методи Рунге-Кутта 2-го та 4-го порядків) та багатокрокові методи (сімейство методів Адамса-Башфорта-Моултона) розглядаються у тісному зв'язку з питаннями стійкості розв'язку диференціального рівняння. Інтерфейсна частина класу, що реалізує ці методи, може мати такий вигляд:

```
typedef vector<double> dvector;
typedef matrix<double> dmatrix;
```

```
/*Клас диференціальних рівнянь */
class DEqu
{
  double t0, //початковий час
  t1, //кінцевий час
  step; //крок за часом
  dvector x0; //вектор початкових умов

/*Будемо розглядати клас рівнянь, що вдається розв'язати від-
носно старшої похідної. У цьому випадку при поданні рівняння
n-го порядку у вигляді системи рівнянь першого порядку і
розв'язанні цієї системи нам потрібна лише одна функція - для
правої частини останнього рівняння в системі, що обчислює його
праву частина за вже обчисленими до цього значеннями молодших
похідних і самої функції. Для першого рівняння права частина -
у нульовий момент часу - це початкове значення першої похід-
ної, для другого рівняння - другої і т.д., а права частина для
останнього рівняння може включати залежність від часу, значен-
ня функції і всіх молодшої похідних. Для уніфікації введення
цієї функції в інтерактивному режимі треба програмувати інтер-
претатор. Оскільки ця робота виходить за межі обговорюваного
курсу, то використовувати цей клас зможе тільки програмуючий
користувач - йому доведеться запрограмувати функцію правої ча-
стини і вказати її ім'я при виклику відповідного методу
розв'язання системи рівнянь. Прототип цієї функції оголошений
через покажчик на функцію, щоб його можна було використовувати
як аргумент іншої функції */

  dvector (*dxbydt)(double t, dvector x);

public:
  /*конструктор приймає як параметри все перераховане вище */
  DEqu(double T0, double Tmax, double Step, dvector X0,
dvector (*Dxbydt)(double T, dvector X)): t0(T0), t1(Tmax),
```

```
step(Step), x0(X0), dxbydt(Dxbydt) {  }

  //конструктор копіювання
  DEqu(DEqu &x): t0(x.t0), t1(x.t1), step(x.step), x0(x.x0),
dxbydt(x.dxbydt) { }

/*Прототипи функцій, що реалізують методи Ейлера, Рунге-Кутта
другого і четвертого порядку точності й Адамса. Функції повер-
тають матриці з кількістю стовпців, що дорівнює порядкові сис-
теми, тобто в нульовому стовпці - значення функції, а в насту-
пних – значення похідних порядку, що відповідає номеру стовпця
*/
  dmatrix EilerSol(), RK2Sol(), RK4Sol(), Adams4Sol();

  //набір функцій для доступу і модифікації параметрів системи
  double &TStart() { return t0; }//початковий час
  double &TEnd() { return t1; }//кінцевий час
  dvector &XStart() { return x0; }/*вектор початкових умов */
  double &StepBy() {return step;}//крок за часом
};
```

Як приклад застосування цього класу доцільно розглянути задачу інтегрування рівнянь руху планет Сонячної системи різними методами з порівнянням результатів обчислень та відповідними висновками про стійкість кожного з методів.

Завершується розділ розглядом часткових методів розв’язування граничних задач [247].

Шостий розділ курсу присвячено чисельним методам розв’язування систем нелінійних рівнянь та пошуку екстремумів функцій. Ми об’єднали ці два класи задач в одному розділі тому, що вони є спорідненими – методи, що використовуються для обчислення коренів, можна використовувати для визначення екстремумів і навпаки.

Будь-яка система нелінійних рівнянь з p невідомими може бути за-

писана у вигляді:

$$f_i(x_1, \ldots, x_p)=0,\ i=1, \ldots, p$$

причому хоча б одна функція f_i нелінійна. З цією системою можна пов'язати деяку невід'ємну функцію $\Phi(x_1, \ldots, x_p)$ таку, що її нульовий мінімум є розв'язком системи нелінійних рівнянь, наприклад:

$$\Phi(x_1, \ldots, x_p)=\sum_{i=1}^{p} f_i^2(x_1,...,x_p)$$

і розв'язувати задачу пошуку коренів методами пошуку екстремумів.

З іншого боку, при заданій для пошуку мінімуму функції багатьох змінних можна, диференціюючи її по кожній із змінних і прирівнюючи частинні похідні нулю, скласти систему рівнянь, розв'язання якої дасть координати шуканого мінімуму:

$$\frac{\partial \Phi}{\partial x_i}=0,\ i=1, \ldots, p$$

і виконати визначення координат мінімуму розв'язуванням системи рівнянь.

Обидва класи задач розпадаються на підкласи:

1) за розмірністю вектора аргументів – одновимірний чи багатовимірний пошук (коренів чи екстремуму);
2) за наявністю чи відсутністю істотних похибок у визначенні значення функцій – пошук в умовах перешкод чи при їхній відсутності;
3) за виконанням попередньої локалізації шуканого об'єкта – визначення екстремуму унімодальної чи багатоекстремальної функції в заданому діапазоні (з одним чи багатьма коренями в інтервалі невизначеності у випадку обчислення коренів).

Відокремлення коренів – це ще мистецтво, тому що загальна кількість підлягаючих локалізації коренів, швидше за все, невідома. На жаль, крім табулювання функції і фіксації меж зміни знаків чи побудови графіків функцій сучасна теорія нічого не пропонує. Істотне спрощення задачі для

поліноміальних рівнянь підказує ідею поліноміальної апроксимації нелінійної функції з наступним обчисленням коренів, але неминучі при апроксимації похибки не дозволять одержати вірогідний результат.

У тих випадках, коли корені дійсні і здійснено їхнє попереднє відокремлення в скінченому інтервалі значень (інтервалі невизначеності), використовують *методи послідовного скорочення інтервалу невизначеності* (метод дихотомії, метод хорд, метод Монте-Карло) та *рекурентні методи уточнення поточної оцінки значення кореня* (метод січних та метод простих ітерацій).

Якщо значення функції при заданих значеннях аргументу визначаються з похибками, якими не можна знехтувати і які істотно спотворюють результати пошуку кореня в заданому інтервалі, використовують так звані *методи стохастичної апроксимації* (процедура Роббінса-Монро).

Серед методів пошуку екстремумів унімодальної функції за відсутності перешкод розглядається *метод Фібоначчі* та його частинні випадки (зокрема, методу золотого перетину), *метод координатного спуску* та *градієнтний метод*. Та найбільшу увагу в цьому розділі приділено *послідовному симплексному пошуку субоптимальної області в багатовимірному просторі*.

Оригінальність методу полягає в тому, що рух до оптимуму в багатовимірному просторі незалежних змінних здійснюється послідовним відображенням вершин симплекса. У *k-вимірному* евклідовому просторі *симплексом* називають фігуру, утворену k+1 точками (вершинами), що не належать одночасно жодному підпростору меншої розмірності. В одновимірному просторі симплекс є відрізок прямої, у двовимірному – трикутник, у тривимірному – тетраедр і т.д. Симплекс називається *регулярним*, якщо відстані між його вершинами рівні. У методі використовуються регулярні симплекси. З будь-якого симплекса, відкинувши одну його вершину, можна одержати новий симплекс, якщо до вершин, що залишилися, додати

всього одну точку. Ця чудова властивість і була використана авторами методу при побудові алгоритму руху симплекса убік шуканого екстремуму. Для оцінки напрямку руху у всіх вершинах симплекса необхідно визначити значення цільової функції Y_j. При пошуку максимуму найбільш доцільним буде рух від вершини v_s з найменшим значенням Y_s до протилежної грані симплекса. Крок пошуку виконується переходом з деякого симплекса $S_{n\text{-}1}$ у новий симплекс S_n шляхом виключення вершини v_s і побудови її дзеркального відображення v_{ns} відносно грані, що є спільною у обох симплексів. Багаторазове відображення гірших вершин приводить до покрокового руху центра симплекса до екстремуму за траєкторією деякої ламаної лінії. Якщо не враховувати експерименти у вершинах вихідного симплекса, то на кожен крок пошуку потрібно лише одне визначення цільової функції.

Основні переваги методу:

1) Число необхідних експериментів для визначення напрямку руху мале в порівнянні з іншими методами.
2) Легко враховуються обмеження на область зміни варійованих при пошуку факторів.
3) Ефективність методу росте зі збільшенням розмірності простору пошуку.
4) Малий обсяг обчислень на кожнім кроці.
5) Відсутність високих вимог до точності оцінки значення цільової функції – достатньо можливості прорангувати значення якісно за принципом «менше».
6) Метод придатний для переслідування дрейфуючого екстремуму (максимуму чи мінімуму), що дозволяє його застосовувати в адаптивних алгоритмах.
7) Можливість змінювати розмірність простору «на ходу» зміною кількості вершин симплекса.

Недоліки методу:

1) Відсутність даних про вплив кожного фактора на цільову функцію.
2) Труднощі інтерпретації характеру поверхні відгуку за даними реалізованих у методі експериментів.

В традиційних курсах чисельних методів симплексний пошук не вивчається, проте його простота та універсальність дозволили включити його в наш курс. У статті [144] наведено приклад програмної реалізації цього методу, об'єднаного з раніше розробленим комплексом для аналізу динамічних систем та показано, що у порівнянні з методами експоненціальної апроксимації та методом моментів при розв'язуванні задачі ідентифікації цей метод є більш стійким до перешкод.

Розгляд симплексного пошуку завершує останній розділ нашого курсу.

Обсяг розробленого курсу не дозволяє детально ознайомитися з усіма класами чисельних методів, проте дає уявлення про основні з них. При цьому матеріал у окремих розділах розташований таким чином, щоб створити цілісну взаємопов'язану систему *математична проблема – обчислювальний алгоритм – чисельний метод*. Реалізація компонентів цієї системи засобами об'єктної методології дозволяє максимально наблизити етапи алгоритмічної та програмної декомпозиції.

Активізація пізнавальної діяльності студентів при вивченні цього курсу здійснюється, з одного боку, шляхом підвищення наочності матеріалу за рахунок використання спеціалізованих математичних класів матриць, векторів, поліномів, комплексних чисел тощо, а з іншого – підвищенням практичної значущості виучуваного матеріалу, свідомості засвоєння знань, підтриманням високого рівня проблемності, широким використанням міжпредметних зв'язків.

§ 2.4. Організація, проведення і результати педагогічного експерименту

З метою перевірки справедливості висунутої гіпотези та визначення ефективності запропонованої методики навчання чисельних методів у об'єктно-орієнтованій технології програмування нами проводився педагогічний експеримент, який проходив у три етапи:

– констатуючий (вересень 1996 р. – червень 1997 р.);
– пошуковий (вересень 1997 р. – червень 1998 р.);
– формуючий (вересень 1998 р. – червень 2000 р.).

Мета експерименту:

– виявити рівень розвитку пізнавальної активності студентів;
– проаналізувати та виявити методології та мови програмування, які найбільш ефективні для розвитку пізнавальної активності в процесі навчання чисельних методів;
– уточнити шляхи та методичні прийоми розвитку пізнавальної активності в процесі навчання чисельних методів у об'єктній методології.

Для досягнення мети експерименту були розв'язані такі завдання:

1. Визначення показників та критеріїв для вимірювання рівнів розвитку пізнавальної активності студентів.
2. Виділення рівнів розвитку пізнавальної активності студентів в процесі навчання чисельних методів.
3. Уточнення шляхів та методичних прийомів розвитку пізнавальної активності в процесі навчання чисельних методів у об'єктно-орієнтованій технології програмування.
4. Перевірка ефективності розробленої методики із застосуванням методів математичної статистики.

В експерименті брали участь студенти фізико-математичного факультету Криворізького державного педагогічного університету (спеціальності «Математика та інформатика» (МІ), «Фізика та інформатика» (ФІ)) та Криворізького технічного університету (спеціальність «Професійне навчання»), всього – 390 студентів. Результативність розробленої методики навчання чисельних методів у об'єктно-орієнтованій технології програмування перевірялася на великій кількості груп. Студенти експериментальних груп навчалися за розробленим нами пробним навчальним посібником «Методи обчислень в класах мови С++», в якому була реалізована концепція дослідження.

На першому етапі (1996–1997 рр.) здійснювався аналіз наявної літератури з проблеми дослідження, вивчався досвід роботи викладачів, розроблялося об'єктно-орієнтоване програмне забезпечення для підтримки курсів лінійної алгебри та чисельних методів. Був проведений констатуючий експеримент, за наслідками якого були визначені критерії та виділені рівні розвитку пізнавальної активності студентів. В результаті експерименту було встановлено, що рівень пізнавальної активності студентів при вивченні курсу чисельних методів є недостатнім.

Під час експерименту були апробовані авторські середовища TUTOR та «Інформатика-96», які містили навчальний матеріал (лекційний курс, лабораторний практикум та батарею тестів) з чисельних методів. Результати цієї роботи опубліковані у [174, 175].

В ході констатуючого експерименту були виділені три рівні розвитку пізнавальної активності, якісну характеристику яких розглянуто в § 1.1. Виділені рівні в процесі констатуючого експерименту уточнювалися і коригувалися протягом дослідження. Для кожного студента обчислювалося середнє значення рівня розвитку пізнавальної активності за 11-бальною шкалою, в основу якої були покладені критерії активності пізнавальної діяльності студентів. При цьому рівень усвідомленості засвоєних знань і мі-

цність знань та навичок оцінювалися як інтегративний критерій за результатами учбової діяльності студентів в процесі вивчення чисельних методів, сформованість вмінь опрацьовувати навчальну інформацію – під час індивідуальної роботи зі студентами на лабораторних та позааудиторних заняттях, а сформованість учбових вмінь – протягом вивчення курсу. Максимальна кількість балів по перших двох критеріях разом була 5 балів, мінімальна – 0 балів. По третьому та четвертому критеріям мінімальна кількість балів була 0, максимальна – 3.

Внесок кожного критерію у загальну оцінку рівня розвитку пізнавальної активності показано у табл. 2.1.

Таблиця 2.1

Внесок критеріїв активності пізнавальної діяльності студентів у загальну оцінку розвитку пізнавальної активності

Критерій	*Бали*	*Вага*
Рівень усвідомленості засвоєння знань	2,5	0,23
Міцність знань та навичок	2,5	0,23
Сформованість вмінь опрацьовувати навчальну інформацію	3	0,27
Сформованість учбових вмінь	3	0,27
Разом	11	1

Для переведення отриманих оцінок розвитку пізнавальної активності студентів у виділені рангові градації нами була застосована така шкала відповідності кількості балів та рівнів розвитку пізнавальної активності: від 0 до 4 балів – низький рівень, від 5 до 7 балів – середній, від 8 до 10 балів – високий.

На другому етапі (1997–1998 рр.) було виділено основні аспекти проблеми дослідження, сформована концепція, гіпотеза і завдання дослідження, проаналізовано еволюцію та сучасний стан вітчизняних і зарубіж-

них курсів чисельних методів, розкрито можливості об’єктної методології до розв’язування задач обчислювальної математики.

Виділення вихідних теоретичних положень дослідження, наявність необхідних експериментальних даних дали змогу організувати і провести пошуковий експеримент. В ході експерименту автором було розроблено та прочитано факультативний курс «Використання алгебраїчних структур при моделюванні атомних процесів в фізиці твердого тіла» для студентів старших курсів і аспірантів фізико-математичного факультету КДПУ. Результати цієї роботи доповідалися на конференціях [156, 157, 213].

Окремі частини розробленого факультативного курсу були використані у методичних рекомендаціях [94] для студентів спеціальності «Математика та інформатика», що містять необхідні відомості з лінійної алгебри, приклади програмної реалізації матричного класу та побудовані на його основі програми регресійного аналізу експериментальних даних.

На третьому етапі (1998–2000 рр.) проводився формуючий експеримент з проблеми дослідження. Концепція дослідження була реалізована в пробних підручниках для фізико-математичних факультетів педагогічних вузів «Методи обчислень в класах мови С++» та «Автоматика», а основні результати опубліковані в [176–180].

Мета формуючого експерименту полягала в перевірці ефективності запропонованої методики розвитку пізнавальної активності в процесі навчання чисельних методів на основі об’єктного підходу. В процесі експерименту виділені особливості об’єктивізації математичного знання і відповідні методичні прийоми. Результати експерименту знайшли відображення в діючій програмі викладання курсу чисельних методів у Криворізькому педуніверситеті.

На кожному етапі експерименту аналізувалися одержані результати, вносилися відповідні корективи, уточнювалися вихідні теоретичні положення дослідження і методика [196]. Для забезпечення рівних умов прове-

дення експерименту навчання в експериментальних та контрольних групах здійснював по можливості один й той самий викладач. Щоб виключити випадковість в оцінюванні рівня розвитку пізнавальної активності студентів, в спірних випадках проводилась співбесіда, в ході якої студентам пропонувалося відповісти на додаткові запитання і виконати завдання, які дають змогу з'ясувати рівень розвиненості у них кожного з компонентів пізнавальної активності.

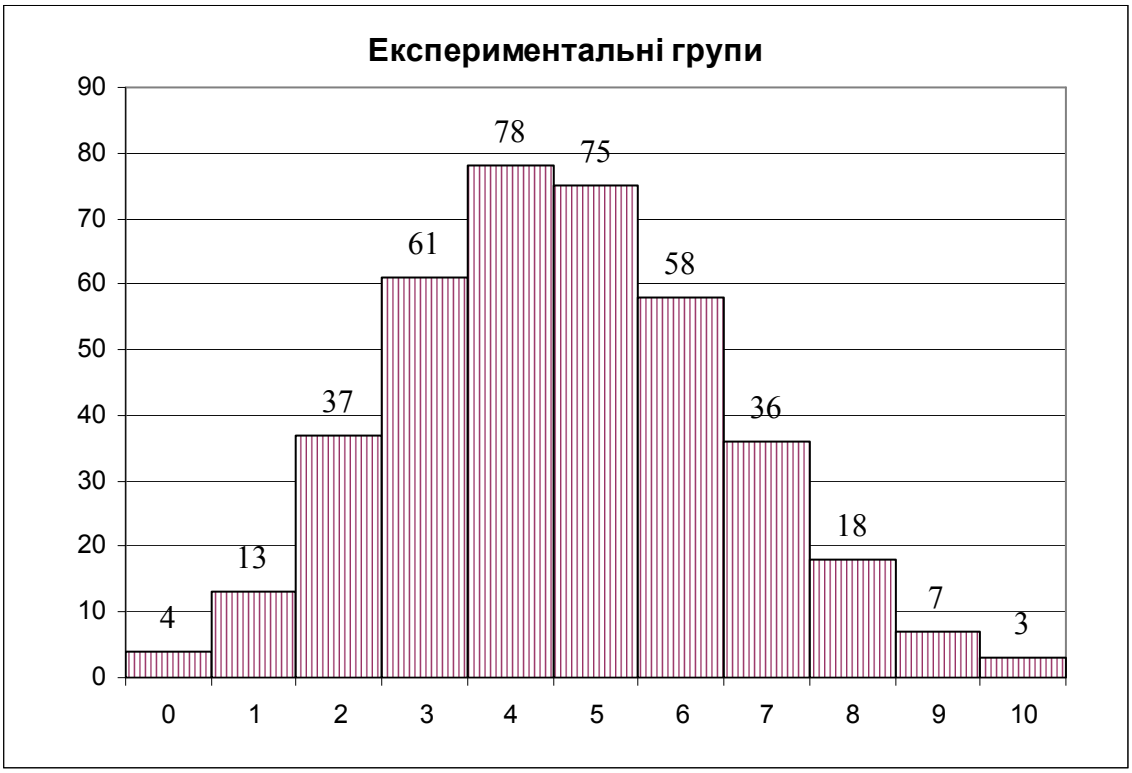
Результати констатуючого та формуючого експериментів у експериментальних (ЕГ) та контрольних (КГ) групах наведено у табл. 2.2. Гістограми розподілу студентів за кількістю набраних балів констатуючого та формуючого експериментів подано на рис. 2.8, 2.9 відповідно. На рис. 2.10 зображено порівняльний розподіл студентів у контрольних та експериментальних групах.

Таблиця 2.2

Розподіл студентів за рівнями пізнавальної активності

Бали	Кількість студентів, які набрали дану кількість балів					
	Констатуючий		Формуючий		Зміни	
	КГ	ЕГ	КГ	ЕГ	КГ	ЕГ
0	5	4	3	2	–2	–2
1	14	13	11	4	–3	–9
2	35	37	27	12	–8	–25
3	63	61	51	33	–12	–28
4	75	78	70	50	–5	–28
5	77	75	78	69	1	–6
6	60	58	65	78	5	20
7	37	36	46	66	9	30
8	16	18	23	45	7	27
9	6	7	12	22	6	15
10	2	3	4	9	2	6
Рівні	Кількість студентів, що мають даний рівень					
низький	192	193	162	101	–30	–92
середній	174	169	189	213	15	44
високий	24	28	39	76	15	48
Рівні	Відсоток студентів, що мають даний рівень					
низький	49,23	49,49	41,54	25,90	–7,69	–23,59
середній	44,62	43,33	48,46	54,62	3,85	11,28
високий	6,15	7,18	10,00	19,49	3,85	12,31

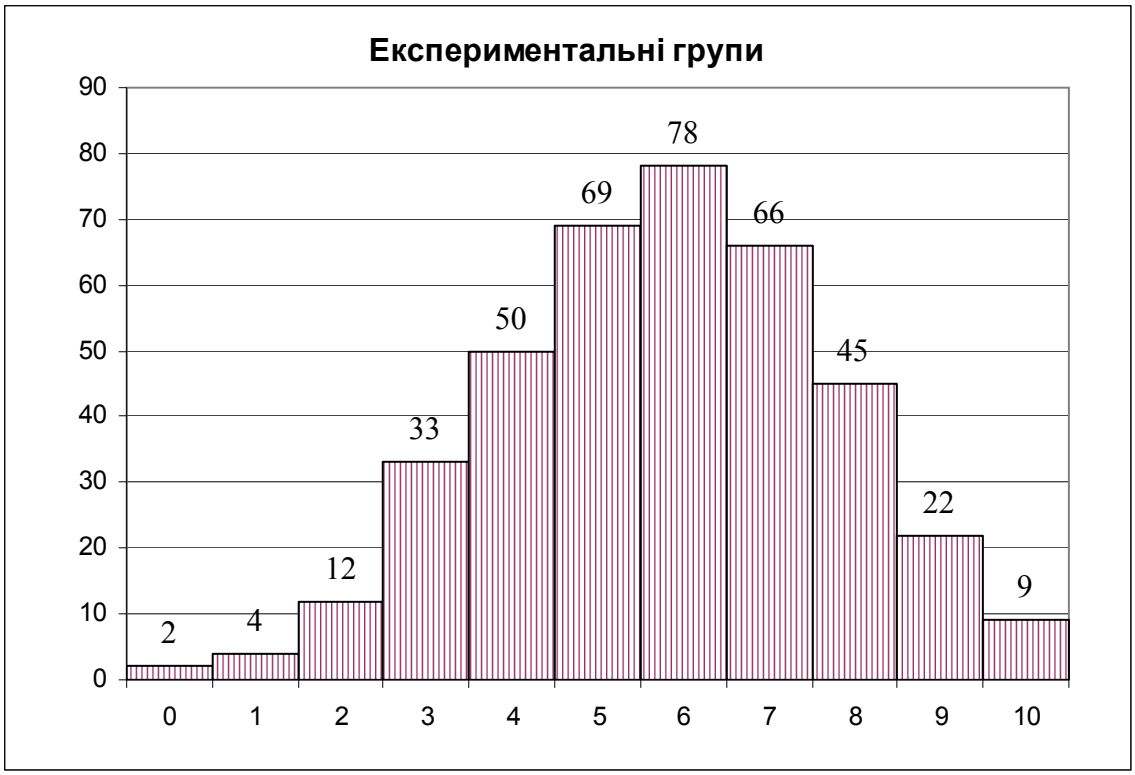
а)



б)

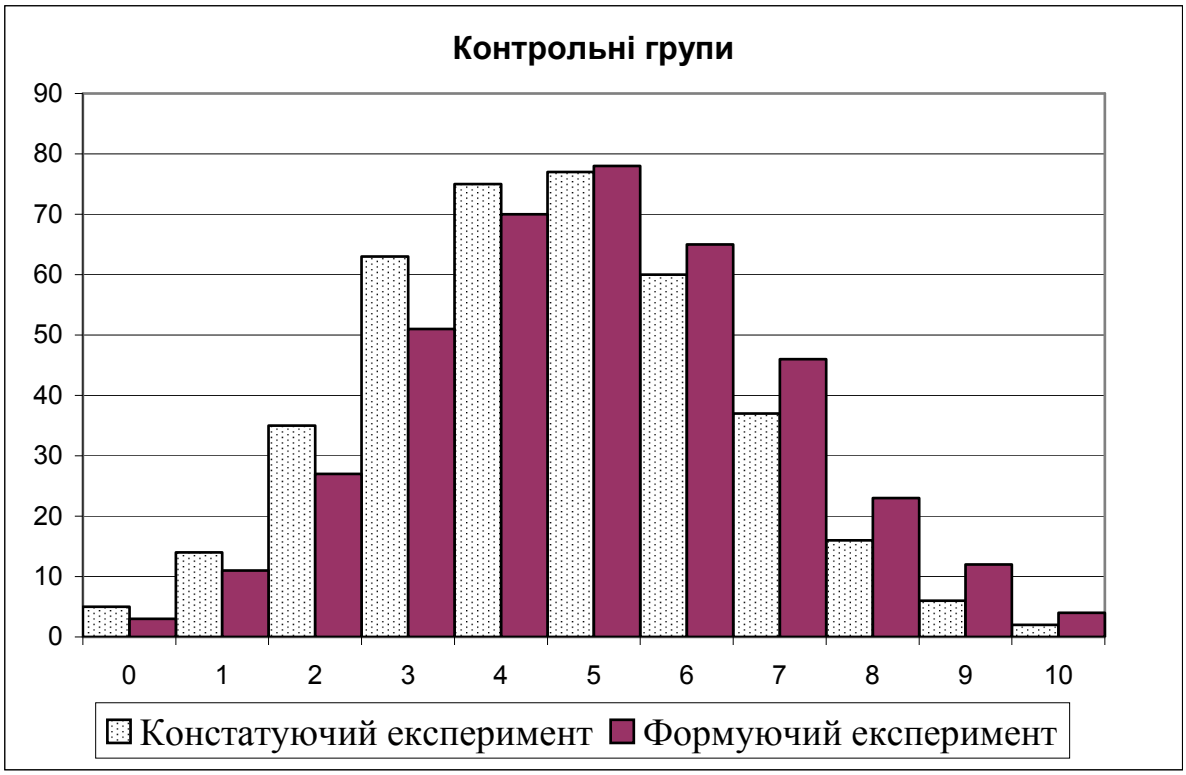
Рис. 2.8. Розподіл студентів за кількістю набраних балів на етапі констатуючого експерименту у контрольних (а) та експериментальних (б) групах

а)

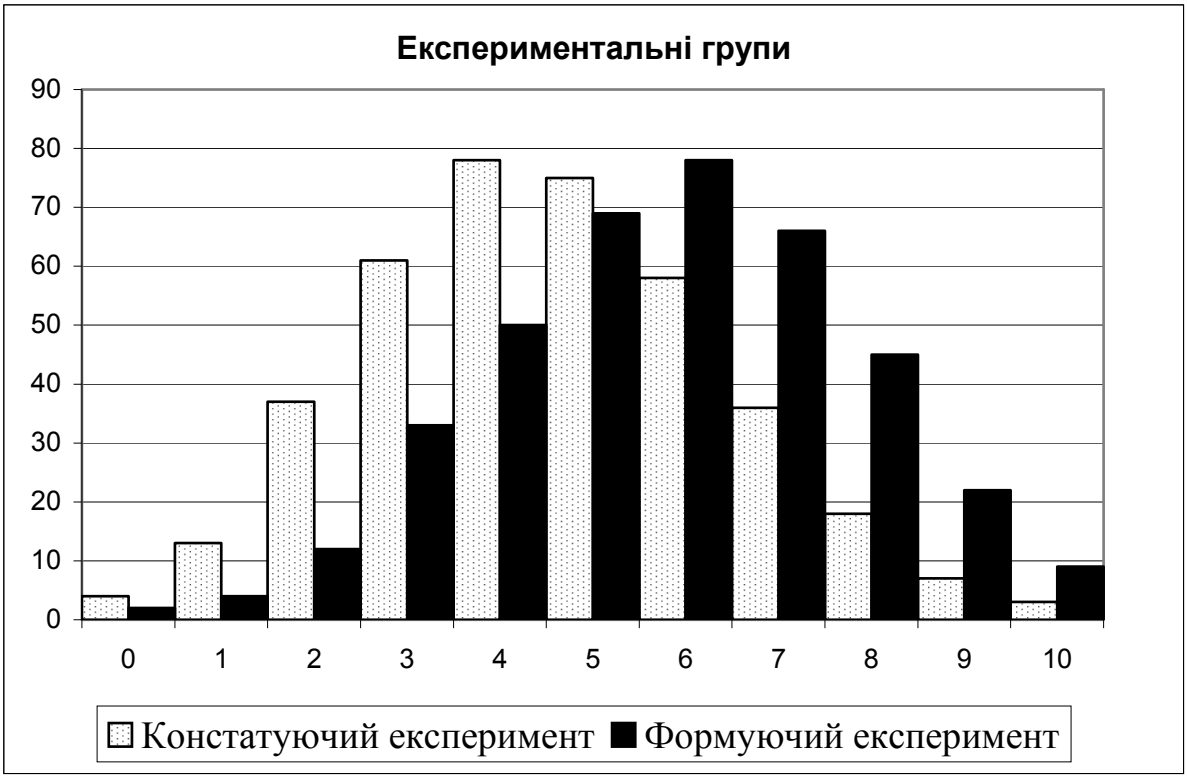


б)

Рис. 2.9. Розподіл студентів за кількістю набраних балів на етапі формуючого експерименту у контрольних (а) та експериментальних (б) групах



а)



б)

Рис. 2.10. Порівняльний розподіл студентів на етапах констатуючого і формуючого експерименту у контрольних (а) та експериментальних (б) групах

Опрацювання результатів експерименту та оцінка ефективності запропонованої методики здійснювалася методами математичної статистики [40, 43]. У нашому дослідженні вибірки випадкові і незалежні. Крім того, з рис. 2.8–2.10 видно, що обидві вибірки підпорядковані нормальному закону розподілу. Це дало нам можливість обробити узагальнені в табл. 2.2 експериментальні дані за критерієм Пірсона (χ^2).

Шкалою вимірювань є шкала з C=11 категоріями, накладено 3 незалежні умови. Отже, кількість степенів свободи $v=C-3=8$.

Нульова гіпотеза H_0: ймовірність попадання студентів контрольної та експериментальної вибірки ($n_1=n_2=390$) в кожну з i (i=0, 1, …, 10) категорій однакова, тобто H_0: $p_{1i}=p_{2i}$ (i=0, 1, …, 10).

Альтернативна гіпотеза H_1: $p_{1i}\neq p_{2i}$ хоча б для однієї із C категорій.

Значення χ^2 обчислюється за формулою:

$$T=\frac{1}{n_1 n_2}\sum_{i=0}^{C-1}\frac{\left(n_1 Q_{2i}-n_2 Q_{1i}\right)^2}{Q_{1i}+Q_{2i}}.$$

p_{1i} – ймовірність оцінювання рівня розвитку активності учасників експериментальних груп на i балів (i=0, 1, …, 10);

p_{2i} – ймовірність оцінювання рівня розвитку активності учасників контрольних груп на i балів (i=0, 1, …, 10);

Q_{1i} – кількість учасників експериментальних груп, які набрали i балів;

Q_{2i} – кількість учасників контрольних груп, які набрали i балів.

Враховуючи, що $n_1=n_2=n$, маємо

$$T=\sum_{i=0}^{C-1}S_{12i},\ S_{12i}=\frac{\left(Q_{2i}-Q_{1i}\right)^2}{Q_{1i}+Q_{2i}}.$$

Нами не було виявлено статистично значущих відмінностей в експериментальній та контрольній вибірках на етапі констатуючого експерименту. Були виявлені статистично значущі відмінності в експериментальній та контрольній вибірках, а також відмінності в контрольній вибірці перед формуючим експериментом і після його проведення.

Результати обчислення статистики вказаних вибірок наведені в табл. 2.3.

Таблиця 2.3

Обчислення χ^2

	Експериментальні та контрольні групи до формуючого експерименту			Експериментальні та контрольні групи після формуючого експерименту			Контрольні групи до та після формуючого експерименту		
	Q_{1i}	Q_{2i}	S_{12i}	Q_{1i}	Q_{2i}	S_{12i}	Q_{1i}	Q_{2i}	S_{12i}
0	4	5	0,111	2	3	0,200	5	3	0,500
1	13	14	0,037	4	11	3,267	14	11	0,360
2	37	35	0,056	12	27	5,769	35	27	1,032
3	61	63	0,032	33	51	3,857	63	51	1,263
4	78	75	0,059	50	70	3,333	75	70	0,172
5	75	77	0,026	69	78	0,551	77	78	0,006
6	58	60	0,034	78	65	1,182	60	65	0,200
7	36	37	0,014	66	46	3,571	37	46	0,976
8	18	16	0,118	45	23	7,118	16	23	1,256
9	7	6	0,077	22	12	2,941	6	12	2,000
10	3	2	0,200	9	4	1,923	2	4	0,667
			0,764			33,712			8,432

З таблиці значень χ^2 для рівня значущості α=0,05 і кількості степенів свободи $v=C-3=8$ визначаємо критичне значення статистики $T_{крит}$=15,51.

Для експериментальної та контрольної вибірок до проведення формуючого експерименту $T<T_{крит}$ (0,764<15,51), що є підставою для прийняття нульової гіпотези H_0. Це означає, що до формуючого експерименту вибірки не мають статистично значущих відмінностей на п’ятивідсотковому

рівні значущості.

Контрольна вибірка до формуючого експерименту та після його проведення не має статистично значущих відмінностей: $T<T_{крит}$ (8,432<15,51). Це означає, що зміни, які відбулися в контрольній виборці за час експерименту, не є істотними.

Обчислення критерію χ^2 для експериментальної та контрольної вибірки після проведення формуючого експерименту показало, що $T>T_{крит}$ (33,712>15,51). Це є основою для відхилення нульової гіпотези. Прийняття альтернативної гіпотези дає підстави стверджувати, що ці вибірки мають статистично значущі відмінності, тобто експериментальна методика більш ефективна, ніж традиційна.

Виконаємо перевірку отриманих під час формуючого експерименту вибірок за критерієм Колмогорова [200]. Цей критерій є непараметричним і застосовується за таких умов:

– вибірки випадкові і незалежні;

– гіпотетичні розподіли повністю визначені і неперервні.

Оскільки обидві ці умови для отриманих нами вибірок виконуються, ми можемо застосувати цей критерій для оцінювання відхилення розподілу в експериментальних групах від розподілу в контрольних групах. Позначимо:

$F(x)$ – невідома функція розподілу ймовірностей рівня пізнавальної активності студентів в контрольних групах;

$G(x)$ – невідома функція розподілу ймовірностей рівня пізнавальної активності студентів в експериментальних групах.

Нульова гіпотеза $H_0 : F(x)=G(x)$

Альтернативна гіпотеза $H_1 : F(x)\neq G(x)$

Коли гіпотеза $H_0 : F(x)=G(x)$ справджується, відхилення

$$D = \sup_x \left| G(x) - F(x) \right|$$

мале, а коли гіпотеза H_0 не справджується, це відхилення велике. При ве-

ликих n (n≥100) межа $\varepsilon_{\alpha;n}$, що відокремлює великі значення D від малих, вибирається як

$$\varepsilon_{\alpha;n} = \frac{\lambda_\alpha}{\sqrt{n}},$$

де λ_α – верхня α-границя розподілу Колмогорова, тобто корінь рівняння $K[\lambda_\alpha;+\infty]=\alpha$, або, що те саме, рівняння $1–K(\lambda_\alpha)=\alpha$.

Для n≥100 і рівня значущості α=0,05 використаємо асимптотичну границю

$$\varepsilon_{0,05;n} = \frac{1{,}36}{\sqrt{n}},$$

для якої справжній коефіцієнт надійності навіть трохи більший від 0,95.

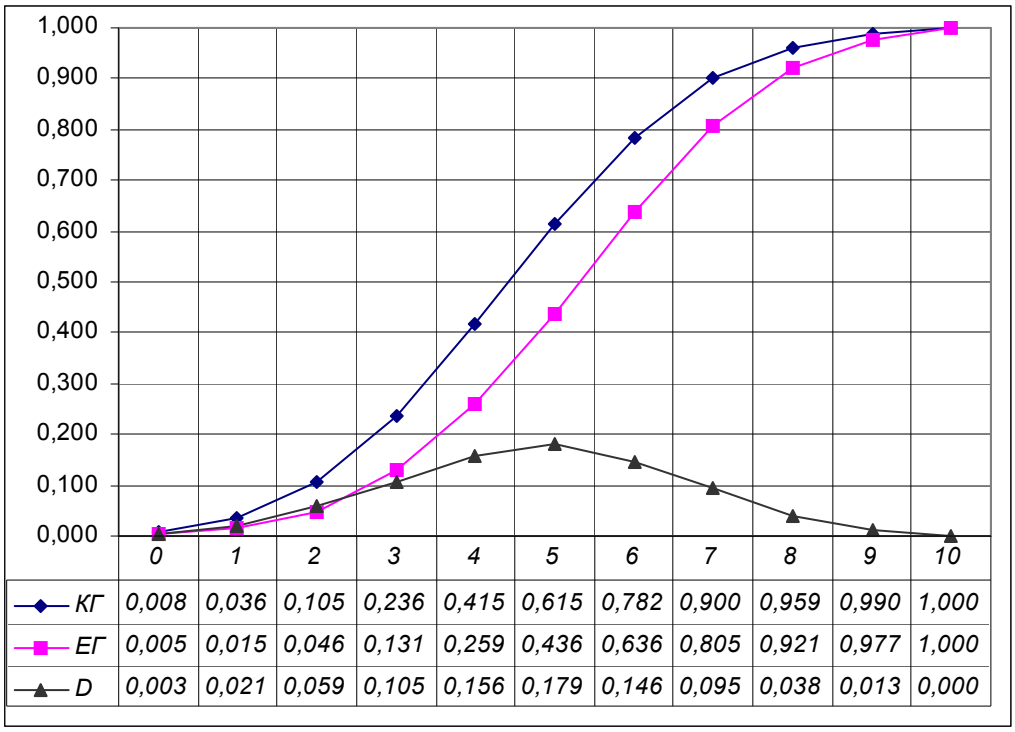
Результати обробки експериментальних даних наведені в табл. 2.4.

Таблиця 2.4

Обчислення критерію Колмогорова

Бали	Абсолютна частота		Накопичена частота		Відносна накопичена частота		D
	КГ	ЕГ	КГ	ЕГ	КГ	ЕГ	
0	3	2	3	2	0,008	0,005	0,003
1	11	4	14	6	0,036	0,015	0,021
2	27	12	41	18	0,105	0,046	0,059
3	51	33	92	51	0,236	0,131	0,105
4	70	50	162	101	0,415	0,259	0,156
5	78	69	240	170	0,615	0,436	0,179
6	65	78	305	248	0,782	0,636	0,146
7	46	66	351	314	0,900	0,805	0,095
8	23	45	374	359	0,959	0,921	0,038
9	12	22	386	381	0,990	0,977	0,013
10	4	9	390	390	1,000	1,000	0,000

На рис. 2.11 подано графічну інтерпретацію розподілів $F(x)$ та $G(x)$.



	0	1	2	3	4	5	6	7	8	9	10
КГ	0,008	0,036	0,105	0,236	0,415	0,615	0,782	0,900	0,959	0,990	1,000
ЕГ	0,005	0,015	0,046	0,131	0,259	0,436	0,636	0,805	0,921	0,977	1,000
D	0,003	0,021	0,059	0,105	0,156	0,179	0,146	0,095	0,038	0,013	0,000

Рис. 2.11. Графіки функцій розподілу студентів на етапі формуючого експерименту у контрольних (КГ) та експериментальних (ЕГ) групах за кількістю набраних балів та модуля їх різниці (D).

З таблиці 2.4 видно, що D=0,179. Граничне значення

$$\varepsilon_{\alpha;n}=\varepsilon_{0,05;\,390}=\frac{1,36}{\sqrt{390}}\approx 0,069.$$

Звідси $D>\varepsilon_{\alpha;n}$ (0,179>0,069), тобто у відповідності з критерієм Колмогорова нульова гіпотеза H_0 : $F(x)=G(x)$ відхиляється і приймається альтернативна гіпотеза H_1 : $F(x)\neq G(x)$. Це означає, що існує відмінність розподілу рівня пізнавальної активності студентів, які навчалися за традиційною методикою і експериментальною. Таким чином, студенти, що навчалися в експериментальних групах, мали більш високий рівень розвитку пізнавальної активності.

Враховуючи, що в експериментальних групах був введений змінний фактор – методичні прийоми і засоби навчання чисельних методів в об’єктній методології, спрямовані на розвиток пізнавальної активності,

можна припустити, що саме це і дало можливість досягти кращих результатів. Отже, можна говорити про експериментальне підтвердження висунутої гіпотези.

У табл. 2.5 наведено середні значення балів по кожній групі студентів, що відповідають виділеним рівням пізнавальної активності.

Таблиця 2.5

Розподіл середніх балів по рівнях пізнавальної активності

Рівні	Констатуючий		Формуючий		Зміни	
	КГ	ЕГ	КГ	ЕГ	КГ	ЕГ
низький	2,98	3,02	3,07	3,24	0,09	0,22
середній	5,77	5,77	5,83	5,99	0,06	0,22
високий	8,42	8,46	8,51	8,53	0,10	0,06

За результатами повторного визначення рівня розвитку пізнавальної активності кожного студента в експериментальній вибірці розподіл студентів за рівнями виявився таким: 76 студентів (20%) мали високий рівень, 213 (54%) – середній і 101 (26%) – низький, тобто 48 студентів (12,3%) перейшли з групи з середнім рівнем в групу з високим, 92 студенти (23,6%) – з групи з низьким рівнем перейшли в групу з середнім рівнем пізнавальної активності, а в останніх студентів відбулися зміни всередині рівня. Найбільш суттєві зміні відбулися у групах з низьким та середнім рівнем пізнавальної активності – середній бал у експериментальній вибірці на цих рівнях підвищився на 0,22 бали.

Підводячи підсумок, приходимо до висновку, що педагогічний експеримент підтвердив гіпотезу нашого дослідження. Аналіз його результатів свідчить про підвищення рівня пізнавальної активності студентів в процесі навчання чисельних методів у об’єктно-орієнтованій технології програмування, а, отже, і про ефективність розробленої методики.

Висновки до другого розділу

У другому розділі викладено основні положення методики розвитку пізнавальної активності студентів в процесі навчання чисельних методів у об'єктно-орієнтованій технології програмування, яка створена на основі закономірностей процесу навчання з урахуванням системи дидактичних та психологічних принципів навчання.

Цілі курсу полягають у ознайомленні з основними принципами побудови та дослідження математичних моделей; систематичному викладі найважливіших методів та прийомів обчислювальної математики, орієнтованих на машинні обчислення; навчанні навичок побудови та використання об'єктно-орієнтованих математичних бібліотек як специфічного засобу програмування обчислювальних задач; формуванні культури дослідницької роботи з використанням обчислювального експерименту.

Зміст курсу містить сукупність двох взаємопов'язаних складових: теоретичної та практичної. Теоретична складова спрямована на формування в студентів наукового теоретичного мислення, здатності до коректної постановки задач, передбачення наслідків прийнятих рішень і дій, свідоме і обґрунтоване використання засобів НІТ в навчанні та трудовій діяльності. Практична складова пов'язана з набуттям студентами умінь: конструювання і добору найбільш ефективних алгоритмів та готових бібліотечних підпрограм для розв'язування задач; виконання об'єктно-орієнтованого аналізу чисельних методів та проектування об'єктно-орієнтованих бібліотек чисельних проблем, алгоритмів та об'єктів; проведення обчислювального експерименту – вибору математичної моделі, створення на її основі дискретної моделі, складання алгоритму розв'язання обчислювальної проблеми з використанням чисельних методів, створення, налагодження та тестування програми, розрахунку за програмою, обробки й аналізу знайдених

чисельних результатів.

Основними організаційними формами навчання чисельних методів у об’єктно-орієнтованій технології програмування є лекції, які одночасно є і методом навчання, практичні та лабораторні заняття, самостійна науково-дослідна робота. Наявність авторського навчального посібника з курсу, що містить у собі більшість необхідного лекційного матеріалу, викликала до життя нову форму – лекції-семінари, на яких у формі диспуту відбувається розгляд теоретичного матеріалу, самостійно вивченого студентами за навчальним посібником.

Крім лекцій, використовуються такі методи навчання, як пояснення і евристична бесіда. Серед методів учіння найбільш ефективними для розвитку пізнавальної активності виявились моделювання та обчислювальний експеримент, вивчення підручників, навчальних посібників, першоджерел та інших матеріалів. Особливу увагу приділено ресурсам мережі Інтернет, що містять інформацію про чисельне об’єктно-орієнтоване програмне забезпечення, яке може бути ефективно використано для самостійної науково-дослідної роботи студентів.

Основним технічним засобом навчання, що застосовується у курсі, є комп’ютер, що виконує три основні функції: інформаційну, контролюючу та навчаючу. До програмних засобів, що виконують інформаційну функцію, відноситься авторський електронний підручник «Інформатика-96», до засобів автоматизованого навчання та тестування – система TUTOR, розроблена автором спільно з О.П. Поліщуком. Остання має підсистеми обліку знань, навчання, анонімного та рейтингового тестування та є ефективним засобом активізації пізнавальної діяльності студентів як при поданні програмового матеріалу (система передбачає індивідуальний темп навчання), так і під час контролю його засвоєння завдяки диференційованому тестуванню, що дозволяє встановити рівень вимог студента щодо оцінки.

Вивчення змісту курсу чисельних методів у об’єктно-орієнтованій технології програмування відбувається у два етапи. На першому етапі відбувається об’єктивізація математичного знання шляхом побудови базису у вигляді математичних класів векторів, поліномів та матриць, які інкапсулюють у собі операції над елементами відповідних множин та типові процедури опрацювання даних, що базуються на цих операціях. Після завершення цієї роботи студенти одержують можливість записувати у своїх програмах операції над відповідними типами у природній математичній нотації. На другому етапі довільній групі чисельних методів ставиться у відповідність конкретна прикладна задача, яка вимагає їх використання.

Актуалізація математичного знання, що виконується на початку побудови кожного класу, дає можливість провести його об’єктивізацію у найбільш раціональний спосіб. Поступовість та повторюваність дій, що виконуються при цьому, дозволяє навіть для студентів з низьким рівнем пізнавальної активності створити ситуацію успіху. При цьому не вимагається виконання повного циклу об’єктно-орієнтованого аналізу, проектування та програмування – на етапі об’єктивізації математичного знання такі студенти повинні усвідомити інструментальну роль математичних класів, що будуються, та навчитися використовувати об’єкти цих класів при програмуванні чисельних методів.

Така побудова курсу дає можливість організувати вивчення чисельних методів на трьох рівнях: базовому, підвищеному та розширеному.

На базовому рівні студенти програмують прикладні задачі, що вимагають застосування чисельних методів, з використанням готової бібліотеки математичних об’єктів. Така бібліотека є допоміжним засобом, що полегшує програмну реалізацію. Студенти мають можливість розширювати її власними математичними, проблемними та алгоритмічними класами протягом усього курсу.

На підвищеному рівні перед тим, як приступати до програмування чисельних методів, студенти виконують об’єктно-орієнтований аналіз та проектування власної бібліотеки математичних об’єктів згідно запропонованої викладачем схеми. При цьому вони отримують можливість не лише розширення бібліотеки, а й повного її перепроектування з метою подання математичних об’єктів у найбільш зручній формі.

На розширеному рівні студенти звертають увагу на інші числові об’єкти. Реалізація цього рівня відбувається шляхом виконання курсових проектів, творчих та конкурсних робіт і виконуються, як правило, групою від двох до п’яти студентів.

Для залучення студентів в активну навчально-пізнавальну діяльність досить ефективними виявилися такі методичні прийоми:

– розкриття необхідності і корисності засвоєння нових знань шляхом надання кожному виучуваному чисельному методу прикладного характеру, завдяки чому краще розвивається мотиваційний компонент пізнавальної активності;
– диференціація завдань на етапі об’єктивізації наявного математичного знання з метою вирівнювання знань студентів;
– використання системи додаткових самостійних завдань для студентів з високим рівнем пізнавальної активності з метою рівномірного засвоєння матеріалу студентами з різним її рівнем.

В ході педагогічного експерименту доведена (з використанням методу перевірки статистичних гіпотез за критеріями Пірсона та Колмогорова) ефективність запропонованої методики розвитку пізнавальної активності студентів в процесі навчання чисельних методів у об’єктно-орієнтованій технології програмування. В студентів експериментальних груп відмічено підвищення якості знань, посилення інтересу до вивчення чисельних методів та їх застосування у прикладних дослідженнях.

ВИСНОВКИ

В ході дослідження одержані такі основні результати:

1) уточнено структуру поняття пізнавальної активності;
2) встановлено рівні розвитку пізнавальної активності студентів в процесі навчання чисельних методів;
3) досліджено можливості застосування об'єктно-орієнтованого програмування до розв'язування задач обчислювальної математики та визначено принципи застосування об'єктного підходу до розробки математичного програмного забезпечення;
4) виконано об'єктно-орієнтований аналіз і проектування об'єктної бібліотеки математичних об'єктів, обчислювальних алгоритмів та чисельних проблем;
5) створено програмно-методичне забезпечення курсу чисельних методів у об'єктно-орієнтованій технології програмування в педвузі;
6) проведено педагогічний експеримент, який підтвердив ефективність запропонованої методики розвитку пізнавальної активності студентів;
7) розробка висунутих теоретичних положень доведена до практичної реалізації у вигляді навчальних посібників для студентів з курсу чисельних методів та автоматики.

Результати дослідження дозволяють зробити такі висновки:

1. Цілеспрямоване формування пізнавальної активності в процесі навчання на основі об'єктно-орієнтованої технології програмування підвищує якість знань студентів, надає результатам навчання практично значущого характеру, розвиває прийоми та властивості мислення, створює необхідні умови для розвитку в студентів творчого мислення, виховує в них відповідальність, силу волі, ініціативність.
2. Застосування об'єктно-орієнтованих бібліотек математичних, алгоритмічних та проблемних класів суттєво прискорює процес програмної реалізації методу, скорочуючи обсяг програми та витрати часу на її написання, роблячи її більш «прозорою» за рахунок підвищення рівня абстракції до операцій над новими типами даних.
3. Конструктивна побудова курсу чисельних методів у об'єктно-

орієнтованій технології програмування дозволяє розширити його розділами, програмна реалізація яких у процедурній методології викликає утруднення, ліквідувати повторюваність чисельних методів у різних розділах курсу, інтегрувати чисельні методи у єдину ієрархію математичних об'єктів за принципами ООП, залучити до курсу ефективні аналітичні та напівчисельні методи, по-новому поглянути на традиційні методи і розширити межі їх застосування.

4. Запровадження об'єктного підходу в навчальний процес впливає на методичну систему навчання чисельних методів на всіх її рівнях:
   – на рівні цілей навчання – з'являється мета вивчення чисельних методів як логічного продовження курсів математики та програмування і необхідної основи курсу моделювання; навчання навичок побудови і використання об'єктно-орієнтованих математичних бібліотек як специфічного засобу програмування обчислювальних задач;
   – на рівні змісту навчання – виникає потреба якісної перебудови усього курсу чисельних методів;
   – на рівні методів навчання – дозволяє ширше застосовувати продуктивні, розвиваючі методи навчання дослідницького характеру;
   – на рівні засобів навчання – виникає можливість побудови об'єктно-орієнтованих ППЗ, таких як інструментально-виконавча система навчання та тестування TUTOR;
   – на рівні організаційних форм – впровадження таких прогресивних форм навчання, як групова та індивідуально-диференційована та поява такої форми, як лекції-семінари.

Отримані результати дозволяють вказати деякі напрями подальших досліджень:

1. Дослідити можливості об'єктно-орієнтованого підходу як технології вивчення складних систем при викладанні курсів методів математичного моделювання та системного аналізу.
2. Створити нові підручники з курсів об'єктно-орієнтованого програмування, методів математичного моделювання, наукової візуалізації та комп'ютерної алгебри на основі об'єктного підходу та використання HITH.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Аврамчук Л.А. Формування активної пізнавальної діяльності студентів // Педагогіка і психологія. – 1997. – № 3. – С. 122-126.
2. Актуальные проблемы современного программирования. Всесоюзная конференция. Секция «Объектно-ориентированное программирование» / АН Эстонии, Ин-т кибернетики. – Таллинн, 1990. – 100 с.
3. Алгоритмы и программы восстановления зависимостей. – М.: Наука, 1984. – 816 с.
4. Алексюк А.М., Кашин С.О. Удосконалення навчального процесу в середній школі. – К.: Вища школа, 1986. – 56 с.
5. Алексюк А.М. Педагогіка вищої освіти України. – К.: Либідь, 1998. – 557 с.
6. Алексюк А.М. Педагогіка вищої школи: Курс лекцій. – К.: УСДО, 1993. – 220 с.
7. Анго А. Математика для электро- и радиоинженеров. – М.: Наука, 1964.
8. Андронов И.К., Окунев А.К., Сырнев Н.И. Вычислительный практикум. Пособие для студентов I и V курсов физ.-мат. фак. пед. ин-тов. – М.: М-во просвещения РСФСР. Моск. обл. пед. ин-т им. Н.К. Крупской, 1963. – 92 с.
9. Аристова Л.П. Активность учения школьника. – М.: Просвещение, 1968. – 138 с.
10. Архангельский С.И. Учебный процесс в высшей школе и его закономерные основы и методы. – М.: Высш. шк., 1980. – 368 с.
11. Бакушинский А.Б., Власов В.К. Элементы высшей математики и численные методы. Учеб. пособие для учащихся 9–10 кл. математич. школ. / Под ред. И.С. Березина. – М.: Просвещение, 1968. – 334 с.
12. Бартків А.Б., Гринчишин Я.Т., Ломакович А.М., Рамський Ю.С. Turbo Pascal: Чисельні методи в фізиці і математиці. – К.: Вища

школа, 1992. – 247 с.

13. Бахвалов Н.С., Жидков Н.П., Кобельков Г.М. Численные методы: Учебное пособие для студентов вузов. – М.: Наука, 1987. – 598 с.
14. Безикович Я.С. Математика. Руководство для поверителей. – М.-Л.: Центр. Упр-ние Промпроганды и Печати ВСНХ СССР, 1926. – 188 с.
15. Белецки Я. Фортран 77: Пер. с польск. О.И. Гуськовой / Под ред. В.Р. Носова. – М.: Высшая школа, 1991. – 207 с.
16. Березин И.С., Жидков Н.П. Методы вычислений. Учеб. пособие для вузов. – Т. 1, 2. – М.: Физматгиз, 1959.
17. Боглаев Ю.П. Вычислительная математика и программирование: Учебное пособие для студентов втузов. – М.: Высшая школа, 1990. – 544 с.
18. Богоявленская Д.Б. Интеллектуальная активность как проблема творчества. – Изд-во Ростовского ун-та, 1983. – 183 с.
19. Бочкин А.И. Методика преподавания информатики. – Минск: Вышейшая школа, 1998. – 431 с.
20. Брадис В.М. Вычислительная работа в курсе математики средней школы. – М.: Изд-во Акад. пед. наук РСФСР, 1962. – 252 с.
21. Брунер Дж. Психология познания: За пределами непосредственной информации / Пер. с англ. К.И. Бабицкого. – М.: Прогресс, 1977. – 412 с.
22. Бугаєнко Г.О., Триус Ю.В., Яринич Ю.О. Лінійні динамічні системи і їх комп’ютерне моделювання // Комп’ютерно-орієнтовані системи навчання. Збірник наукових праць – К.: «Комп’ютер у школі та сім’ї», 1998. – С. 62–70.
23. Бурда М.І. Методичні основи диференційованого формування геометричних умінь учнів основної школи: Дис. … д-ра пед. наук: 13.00.02 / АПН України, Інститут педагогіки. – К., 1994. – 347 с.
24. Бут Э.Д. Численные методы. – М.: Физматгиз, 1959. – 239 с.

25. Буч Г. Объектно-ориентированное проектирование с примерами применения. – М.: И.В.К., К.: Диалектика, 1992. – 519 с.
26. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C++. – 2-е изд. / Пер. с англ. – М.: Бином, СПб.: Невский диалект, 1999. – 560 с.
27. Быков В.И., Кытманов А.М., Лазман М.З. Методы исключения в компьютерной алгебре многочленов. – Новосибирск: Наука, 1991. – 233 с.
28. Быч Е.В., Семериков С.А. Теоретико-числовое преобразование в вычислениях с произвольной точностью // Комп'ютерне моделювання та інформаційні технології в природничих науках. – Кривий Ріг: Видавничий відділ КДПУ, 2000. – С. 443-446.
29. Вабищевич П.Н. Численное моделирование. – М.: Изд-во Моск. ун-та, 1993. – 152 с.
30. Вайнер Р., Пинсон Л. C++ изнутри. – К.: ДиаСофт, 1993. – 304 с.
31. Верлань А.Ф., Апатова Н.В. Інформатика. – Київ: Квазар-Мікро, 1998. – 200 с.
32. Верлань А.Ф., Олецький О.В. Методика створення інтелектуалізованого інтегрованого середовища для комп'ютерного моделювання складних систем: Сб. научн. трудов по материалам ежегодных конф. ин-та 1996 г. и 1997 г. – К.: НАН Украины, Институт проблем моделирования в энергетике, 1997. – 86 с.
33. Верлань А.Ф., Широчин В.П. Информатика и ЭВМ. – К.: Техніка, 1987. – 344 с.
34. Владимиров В.Б. Введение в объектно-ориентированное программирование // Компьютеры+программы. – 1993. – №2. – С. 70-73.
35. Волков Е.А. Численные методы. Учеб. пособие для инж.-техн. спец. вузов. – М.: Наука, 1982. – 254 с.
36. Воробьев А.Ф., Данелян Т.Я., Данилина Н.И. Вычислительная мате-

матика. – М.: Статистика, 1966. – 164 с.

37. Выготский Л.С. Развитие высших психологических функций. – М.: Изд-во АПН СССР, 1960.
38. Гаврилюк І.П., Макаров В.Л. Методи обчислень: Підруч. для студ. вузів, які навч. за спец. «Прикладна математика». – Ч. 1, 2. – К.: Вища школа, 1995.
39. Гантмахер Ф.Р. Теория матриц. – М.: Государственное издательство технико-теоретической литературы, 1953. – 492 с.
40. Гласс Д., Стэнли Д. Статистические методы в педагогике и психологии / Пер. с англ. – М.: Прогресс, 1971. – 495 с.
41. Гокунь О.О., Жалдак М.І., Машбиць Ю.І. та ін. Основи нових інформаційних технологій навчання. Посібник для вчителів. – К.: Віпол, 1997. – 262 с.
42. Головань М.С. Розвиток пізнавальної активності учнів в процесі навчання алгебри і початку аналізу на основі НІТ: Дис. … канд. пед. наук: 13.00.02 / Укр. держ. педагогічний ун-т ім. М.П. Драгоманова. – К., 1997. – 177 с.
43. Грабарь М.И., Краснянская К.А. Применение математической статистики в педагогических исследованиях: Непараметрические методы. – М.: Просвещение, 1977. – 136 с.
44. Грегори Р.Т., Кришнамурти Е.В. Безошибочные вычисления: Методы и приложения. – М.: Мир, 1988. – 207 с.
45. Гринчишин Я.Т. TURBO PASCAL: Чисельні методи в фізиці та математиці: Навч. посібник. – Тернопіль, 1994. – 121 с.
46. Грищенко Н.В. Використання чисельного об’єктно-орієнтованого програмного забезпечення для розрахунків твердотільних структур // Друга Всеукраїнська конференція молодих науковців «Інформаційні технології в науці та освіті». 18-20 квітня 2000 р. – Черкаси: ЧДУ. – 1998. – С. 9-10.

47. Грищенко Н.В., Семериков С.А., Хараджян А.А., Чернов Е.В. Сравнительный анализ методов аппроксимации. – Кривой Рог: КГПИ, 1998. – 25 с.
48. Грищенко Н.В., Хараджян О.А., Семеріков С.О. Дивергентне мислення та його роль у навчальній діяльності // Сучасна трудова та професійна підготовка шкільної молоді. Збірник наукових праць. – Кривий Ріг. – 1998. – С. 102-105.
49. Грищенко Н.В., Чернов Є.В., Семеріков С.О. Фізичні моделі в курсі «Основи комп'ютерного моделювання» // Перша міжнародна науково-практична конференція «Методичні та організаційні аспекти використання мережі ІНТЕРНЕТ в закладах науки та освіти» (ІНТЕРНЕТ – ОСВІТА – НАУКА – 98). Матеріали конференції. – Том 2. – Вінниця: «УНІВЕРСУМ-Вінниця». – 1998. – С. 341-348.
50. Грузман М.З. Обучение учащихся средней школы программированию на основе структурного подхода: Автореферат дис. … канд. пед. наук: 13.00.02 / НИИ педагогики. – К., 1987. – 22 с.
51. Грузман М.З. Эвристика в информатике. – Винница: Арбат, 1998. – 308 с.
52. Гусак А.А. Элементы методов вычислений. – Минск: Изд-во БГУ, 1974. – 167 с.
53. Гутер Р.С. и др. Программирование и вычислительная математика. – Т. 1, 2. – М.: Наука, 1971.
54. Дал У.-И., Мюрхауг Б., Нюгорд К. Симула–67: универсальный язык программирования. – М.: Мир, 1969. – 100 с.
55. Данилович В.П. Чисельні методи в задачах і вправах: Навчальний посібник для студ. спец. «Прикладна математика». – Львів: ІСДО «Львівська політехніка», 1995. – 248 с.
56. Демидович Б.П., Марон И.А. Основы вычислительной математики: Учеб. пособие для втузов. – М.: Физматгиз, 1960. – 659 с.

57. Демидович Б.П., Марон И.А., Шувалова Э.З. Численные методы анализа. – М.: Физматгиз, 1963. – 400 с.
58. Джидарьян И.А. Категория активности и ее место в системе психологического знания // Категории материалистической диалектики и психологии / Под. ред. И.А. Анциферовой. – М.: Наука, 1988. – С. 56-89.
59. Дьюхарст С., Старк К. Программирование на С++. – К.: ДиаСофт, 1993. – 272 с.
60. Дэвенпорт Дж., Сирэ И., Турнье Э. Системы и алгоритмы алгебраических вычислений. – М.: Мир, 1991. – 350 с.
61. Дэкин А. Прикладная математика для школ, курсов и самообразования / Пер. с англ. под ред. Д.А. Крыжановского. – М.: Госиздат, 1926. – 452 с.
62. Еникеев М.И. Теория и практика активизации учебного процесса. – Казань: Татариздат, 1963. – 122 с.
63. Ерофеев И.Г., Ляшук С.Р. Техническая математика. – Вып. II – М.-Л.: Огиз – Гос. изд. с.-х. и колхоз.-кооп. лит-ры, 1931. – 223 с.
64. Ершов А.П. Компьютеризация школы и математическое образование // Информатика и образование. – 1992. – № 5-6. – С. 3-20.
65. Ершов А.П. Об объектно-ориентированном взаимодействии с ЭВМ // Микропроцессорные средства и системы. – 1985. – № 3. – С. 2.
66. Єпішин О.В., Федоренко Д.С., Поліщук О.П. Створення спеціалізованих математичних класів та їх застосування при викладанні чисельних методів // Фізика. Математика. Нові технології навчання. Збірник матеріалів Всеукраїнської студентської науково-практичної конференції. – Кіровоград: РВГІЦ КДПУ ім. В. Винниченка. – 1999. – С 36-37.
67. Есипов Б.П. Самостоятельная работа учащихся на уроке. – М.: Учпедгиз, 1961. – 239 с.

68. Жалдак М.І. Комп'ютер на уроках математики. – К.: Техніка, 1997. – 303 с.
69. Жалдак М.І. Методика вивчення основ інформатики та обчислювальної техніки в педагогічному вузі: Учбовий посібник / КДПІ ім. О.М. Горького. – К., 1986. – 74 с.
70. Жалдак М.И. Система подготовки учителя к использованию информационной технологии в учебном процессе. Дисс. … докт. пед. наук. – М., 1989. – 48 с.
71. Жалдак М.І., Ковбасенко Б.С., Рамський Ю.С. Обчислювальная математика. Спец. курс факультативних занять у 9-х і 10-х кл. – К.: Рад. школа, 1973. – 184 с.
72. Жалдак М.І., Кузьміна Н.М., Берлінська С.Ю. Теорія ймовірностей і математична статистика з елементами інформаційної технології. – К.: Вища школа, 1995. – 352 с.
73. Жалдак М.І., Рамський Ю.С. Інформатика: Навч. посібник / За ред. М.І. Шкіля. – К.: Вища шк., 1991. – 319 с.
74. Жалдак М.І., Рамський Ю.С. Чисельні методи математики: Посібник для самоосвіти вчителів. – К.: Рад. школа, 1984. – 206 с.
75. Жильцов О.Б. Розвиток розумової діяльності учнів 7 класів середньої школи при вивченні математики з використанням НІТ: Дис. … канд. пед. наук: 13.00.02. – К., 1994. – 227 с.
76. Иванов В.В. Методы вычислений на ЭВМ: Справочное пособие. – К.: Наукова думка, 1986. – 568 с.
77. Иванова Л.А. Активизация познавательной деятельности учащихся при изучении физик: Пособие для учителей. – М.: Просвещение, 1983. – 160 с.
78. Иванова Т.П., Пухова Т.В. Вычислительная математика и программирование: Учебное пособие для студентов физико-математических факультетов педагогических ин-тов. – М.: Просвещение, 1978. –

320 с.

79. Извеков Б.И. Сборник задач по прикладной математике: Для студентов, аспирантов, и препод. втузов. – Ч. 1. – Л.-М.: Гос. техн.-теоретич. изд., 1932. – 396 с.
80. Ильин В.С. Проблема воспитания потребности в знаниях у школьников: Дис. … д-ра пед. наук. – М., 1971.
81. Использование Turbo Assembler при разработке программ. – К.: Диалектика, 1993. – 288 с.
82. Калайда О.Ф., Янішевський А.Т. Елементи програмування та обчислювальної математики. – Ч. 1, 2. – К., 1972.
83. Калеева И.Н. Основы теории вычислений. – М.: Статистика, 1973. – 136 с.
84. Калиткин Н.Н. Численные методы: Учеб. пособие для вузов / Под. ред. А.А. Самарского. – М.: Наука, 1978. – 512 с.
85. Калмыкова З.И. Психологические принципы развивающего обучения. – М.: Знание, 1979. – 48 с.
86. Кантор И.Л., Солодовников А.С. Гиперкомплексные числа. – М. Наука, 1973. – 144 с.
87. Касаткин А.И., Вальвачев А.Н. От Turbo C к Borland C++. – Минск: Вышейшая школа, 1992. – 228 с.
88. Касаткин В.Н., Верлань А.Ф. Основы информатики и вычислительной техники: Проб. учеб. пособие для 10-11 кл. сред. шк. – К.: Рад. шк., 1989. – 223 с.
89. Козин А.С., Лященко Н.Я. Вычислительная математика: Пособие для факультатив. занятий в 10 классе. – К.: Рад. школа, 1983. – 191 с.
90. Коллатц Л. Функциональный анализ и вычислительная математика. – М.: Мир, 1969. – 447 с.
91. Компьютерная алгебра: Символьные и алгебраические вычисления / Бухбергер Б., Калле Ж., Калтофен Э. и др. – М.: Мир, 1986. – 392 с.

92. Коновалец Л.С. Познавательная самостоятельность учащихся в условиях компьютерного обучения // Педагогика. – 1999. – № 2. – С. 46-50.
93. Копченова Н.В., Марон И.А. Вычислительная математика в примерах и задачах: Учеб. пособие для втузов. – М.: Наука, 1972. – 367 с.
94. Корольский В.В., Соловйов В.М., Семеріков С.О. Комп'ютерна підтримка курсу лінійної алгебри. – Кривий Ріг: КДПІ, 1998. – 26 с.
95. Кочанов Н.С. О вычислениях с помощью арифмометра. – М.: Знание, 1967. – 80 с.
96. Крутецкий В.А. Психология обучения и воспитания школьников: Пособие для учителей и классных руководителей. – М.: Просвещение, 1976. – 303 с.
97. Крылов А.Н. Лекции о приближенных вычислениях: Учеб. пособие для вузов. – Изд. 6-е. – М.: Гостехиздат, 1954. – 400 с.
98. Крылов В.И., Бобков В.П., Монастырский П.И. Вычислительные методы высшей математики: Учеб. пособие для фак. прикл. математики ун-тов. – Т. 1, 2. – Минск: Вышейшая школа, 1975.
99. Кузнецов А.Б. Программа курса «Основы объектно-ориентированного программирования» // Информатика и образование. – 1998. – № 7. – С. 17-24.
100. Ланина И.Я. Формирование познавательных интересов учащихся на уроках физики: Книга для учителя. – М.: Просвещение, 1985. – 128 с.
101. Лебедев М.П. Поняття пізнавальної активності учнів і шляхи її вимірювання // Радянська школа. – 1970. – № 9. – С. 6-11.
102. Леонтьев А.Н. Деятельность. Сознание. Личность. – М.: Высш. шк., 1976. – 302 с.
103. Леонтьев А.Н. Проблемы развития психики. – М.: Изд-во Моск. ун-та, 1981. – 584 с.
104. Лернер И.Я. Дидактические основы методов обучения. – М.: Педаго-

гика, 1981. – 185 с.

105. Лесневский А.С. Практикум по объектно-ориентированному проектированию и программированию // Информатика и образование. – 1998. – № 5. – С. 114-121.
106. Лисовский В.Т., Дмитриев А.В. Личность студента. – Л.: Изд-во ЛГУ, 1974. – 183 с.
107. Ломонова М.Ф. Формирование творческого отношения студентов к учению: Дис. … канд. пед. наук: 13.00.01. / Одесский пед. ин-т им. К.Д. Ушинского. – Одесса, 1992. – 168 с.
108. Лукас П. С++ под рукой. – К.: ДиаСофт, 1993. – 176 с.
109. Лященко М.Я., Головань М.С. Чисельні методи: Підручник для студ. пед. навч. закладів. – К.: Либідь, 1996. – 287 с.
110. Маликов В.Т., Кветный Р.Н. Вычислительные методы и применение ЭВМ: Учебное пособие. – К.: Выща школа, 1989. – 213 с.
111. Марчук Г.И. Методы вычислительной математики. – Новосибирск: Наука, 1973. – 352 с.
112. Маслов С.Г. О некоторых различиях объектно-ориентированного и сборочного программирования // Всесоюзная конференция «Актуальные проблемы современного программирования». Секция «Объектно-ориентированное программирование» / АН Эстонии, Ин-т кибернетики. – Таллинн. – 1990. – С. 25.
113. Математический энциклопедический словарь. – М.: Сов. энцикл., 1988. – 846 с.
114. Махмутов М.И. Развитие познавательной активности и самостоятельности учащихся в школах Татарии. – Казань: Татариздат, 1963. – 80 с.
115. Машбиц Е.И. Компьютеризация обучения: проблемы и перспективы. – М.: Знание, 1986. – 80 с.
116. Машбиц Е.И. Психологические основы управления учебной дея-

тельностью. – К.: Выща школа, 1987. – 224 с.

117. Машбиц Е.И. Психолого-педагогические проблемы компьютеризации обучения. – М.: Педагогика, 1988. – 192 с.
118. Медовой М.И. Абу-л-Вафа и средневековая бесцифровая вычислительная техника в странах Ислама. Автореферат дисс. канд. … физ.-мат. наук. / МГУ им. М.В. Ломоносова. – М., 1960. – 9 с.
119. Методи обчислень: Практикум на ЕОМ. Навч. посібник для студ. вузів, які навчаються із спец. «Прикладна математика» / Бурківська В.Л., Войцехівський С.О., Гаврилюк І.П. та ін. – К.: Вища школа, 1995. – 303 с.
120. Моисеев Н.Н. Человек. Среда. Общество. Проблемы формализованного описания. – М.: Наука, 1982. – 284 с.
121. Молонов Г.Ц. Формирование познавательной активности школьников в процессе обучения и воспитания: Дис. … д-ра пед. наук. – Улан-Уде, 1986. – 426 с.
122. Монахов В.М. Введение в школу приложений математики, связанных с использованием ЭВМ: Автореф. … докт. пед. наук. – М., 1973. – 63 с.
123. Морозов С.В. Объектно-ориентированная инструментальная среда для создания приложений численного моделирования. Дисс. … кандидата физ.-мат. наук. – М.: ИСП РАН, 1998.
124. Морозов С.В., Семенов В.А. Объектно-ориентированное программирование задач численного анализа // Вопросы кибернетики. Приложения системного программирования / Под ред. В.П. Иванникова. – М.: Науч. совет по комплексной проблеме «Кибернетика» РАН, 1995. – С. 189-211.
125. Морозов С.В., Семенов В.А., Тарлапан О.А., Ширяева Е.Ю. Объектно-ориентированная инструментальная среда для разработки вычислительных приложений // Материалы XXIII международной конфе-

ренции "Новые информационные технологии в науке, образовании и бизнесе", Украина, Крым, Ялта – Гурзуф, 15-24 мая 1996 г. – С. 63-66.

126. Мудров А.Е. Численные методы для ПЭВМ на языках Бейсик, Фортран и Паскаль. – Томск: Раско, 1991. – 280 с.
127. Нивельгерт Ю., Фаррар Дж., Рейнголд Э. Машинный подход к решению математических задач. – М.: Мир, 1977. – 351 с.
128. Низамов Р.А. Дидактические основы активизации учебной деятельности студентов. – Казань: Изд-во Казанского ун-та, 1975.
129. Оглоблин Н.В. Определение степени точности при логарифмических вычислениях. – К.: Типография Императорского Университета Св. Владимира, 1911. – 12 с.
130. Ожегов С.И. Словарь русского языка. – М.: Советская энциклопедия, 1973. – 846 с.
131. Основы компьютерной грамотности / Е.И. Машбиц, Л.П. Бабенко, Л.В. Верник и др. – К.: Выща шк. Головное изд-во, 1988. – 215 с.
132. Основы педагогики и психологии высшей школы / Под ред. Петровского А.В. – М.: Изд-во МГУ, 1986. – 303 с.
133. Педагогічна психологія. Навч. посібник / Л.М. Проколієнко та ін.; За ред. Л.М. Проколієнко, Д.Ф. Ніколенка. – К.: Вища школа, 1991. – 183 с.
134. Перри Д. Практическая математика. – М.: Тип. Т-ва И.Д. Сытина, 1909. – 300 с.
135. Петровский В.А. К психологии активности личности // Вопросы психологии. – 1975. – № 3. – С. 26-39.
136. Петухова Н.А., Жернак А.Н., Петухов О.А. Машинные методы вычислений: Учебное пособие. – Ярославль: СЗПИ, 1979. – 67 с.
137. Пиаже Ж. Избранные психологические труды. – М.: Международная педаг. академия, 1994. – 680 с.

138. Повышение вычислительной культуры учащихся средней школы. Сборник статей. Пособие для учителей. Ред.-сост. П.В. Стратилатов. – М.: Просвещение, 1965. – 168 с.
139. Пол А. Объектно-ориентированное программирование на C++. – 2-е изд. / Пер. с англ. – М.: Бином, СПб.: Невский диалект, 1999. – 560 с.
140. Полищук А.П. Персональный компьютер и его программирование (C, C++, Паскаль): Учебно-справочное пособие. – Кривой Рог, 1997. – 475 с.
141. Полищук А.П., Семериков С.А. Автоматика: Учебное пособие. – Кривой Рог: Издательский отдел КГПИ, 1999. – 277 с.
142. Полищук А.П., Семериков С.А. Концепция курса «Численные методы в объектной методологии» // Комп'ютерне моделювання та інформаційні технології в освітній діяльності. – Кривий Ріг: Видавничий відділ КДПУ. – 1999. – С. 131-138.
143. Полищук А.П., Семериков С.А. Методы вычислений в классах языка C++: Учебное пособие. – Кривой Рог: Издательский отдел КГПИ, 1999. – 350 с.
144. Полищук А.П., Семериков С.А. Последовательный симплекс-поиск в задачах параметрической идентификации // Комп'ютерне моделювання та інформаційні технології в природничих науках. – Кривий Ріг: Видавничий відділ КДПУ, 2000. – С. 125-136.
145. Полищук А.П., Семериков С.А., Грищенко Н.В. О выборе языка программирования для начального обучения // Комп'ютерне моделювання та інформаційні технології в природничих науках. – Кривий Ріг: Видавничий відділ КДПУ, 2000. – С. 212-228.
146. Половникова Н.А. Исследование процесса формирования познавательной активности школьников в обучении. – Казань, 1976. – 198 с.
147. Поттосин И.В. Об объектно-ориентированном подходе к конструированию программ // Всесоюзная конференция «Актуальные про-

блемы современного программирования». Секция «Объектно-ориентированное программирование» / АН Эстонии, Ин-т кибернетики. – Таллинн. – 1990. – С. 65-67.

148. Придатко С. Практические вычисления: Рук-во для физико-математических факультетов … – М.: Государственное Издательство, 1924. – 158 с.
149. Пулькин С.П. Вычислительная математика: Пособие для учащихся 9–10-х кл. по факультативному курсу. – М.: Просвещение, 1974. – 239 с.
150. Пулькин С.П. Вычислительная математика: Пособие для учителей по факультативному курсу. – М.: Просвещение, 1972. – 271 с.
151. Пулькин С.П. Теория и практика вычислений: Учеб. пособие для студентов заоч. отд-ний физ.-мат. фак. пед. ин-тов. – М.: Просвещение, 1967. – 183 с.
152. Пышкало А.М. Методическая система обучения геометрии в начальной школе. Авторский доклад по монографии «Методика обучения элементам геометрии в начальных классах», представленной на соискание ученой степени докт. пед. наук. – М.: НИИ СиМО АПН СССР, 1975.
153. Ракитин В.И., Первушин В.Е. Практическое руководство по методам вычислений с приложением программ для персональных компьютеров. – М.: Высшая школа, 1998. – 383 с.
154. Рамський Ю.С. Формування інформаційної культури вчителя математики при вивченні методів обчислень у педагогічному вузі // Комп'ютерно-орієнтовані системи навчання. Збірник наукових праць. Випуск 2. – К.: НПУ ім. М.П. Драгоманова, 2000. – С. 25-47.
155. Рассудовская М.М. Проблемы вычислительной математики на факультативных занятиях в 9-ом и 10-ом классах средней школы: Автореф. дис. … канд. пед. наук: 13.00.02 / Моск. обл. пед. ин-т им.

Н.К. Крупской. – М., 1973. – 20 с.

156. Родькин Д.И., Хараджян А.А., Михайлов С.Л., Чернов Е.В., Семериков С.А. О методах определения параметров и энергодиагностике систем электропривода // Збірник наукових праць Східноукраїнського державного університету. Серія «Машинобудування». – Луганськ: Видавництво Східноукраїнського державного університету. – 1998. – С. 170-182.
157. Родькин Д.И., Хараджян А.А., Семериков С.А. Метод определения параметров двигателя постоянного тока // Комп'ютерне моделювання та інформаційні технології в освітній діяльності. – Кривий Ріг: Видавничий відділ КДПУ. – 1999. – С. 27-33.
158. Рубинштейн С.Л. О мышлении и путях его исследования. – М.: Изд-во АН СССР, 1958. – 147 с.
159. Рубинштейн С.Л. Основы общей психологии. – М.: Госучпедгиздат Мин-ва просвещения, 1946. – 704 с.
160. Руденко М.П. Критерії активності пізнавальної діяльності учнів // Фізика та астрономія в школі. – 1999. – № 3. – С. 6-10.
161. Салтыков А.И., Семашко Г.Л. Программирование для всех. – М.: Наука. Главная редакция физико-математической литературы, 1986. – 176 с.
162. Самарский А.А. Введение в численные методы. – М.: Наука, 1987. – 286 с.
163. Самарский А.А., Гулин А.В. Численные методы: Учеб. пособие для студ. вузов, обучающихся по спец. «Прикладная математика». – М.: Наука, 1989. – 430 с.
164. Самохин А.Б., Самохина А.С. Численные методы и программирование на Фортране для персонального компьютера. – М.: Радио и связь, 1996. – 224 с.
165. Семенов В.А., Морозов С.В. Объектно-ориентированное программи-

рование квадратурных методов // Вопросы кибернетики. Приложения системного программирования / Под ред. В.П. Иванникова. – М.: Науч. совет по комплексной проблеме «Кибернетика» РАН. – 1996. – Вып. 2. – С. 120-146.

166. Семенов В.А., Морозов С.В., Тарлапан О.А., Ширяева Е.Ю. Объектно-ориентированная инструментальная среда для разработки систем численного моделирования // Вопросы кибернетики. Приложения системного программирования / Под ред. В.П. Иванникова. – М.: Науч. совет по комплексной проблеме «Кибернетика» РАН. – 1997. – Вып. 3. – С. 205-226.
167. Семенов В.А. Об объектно-ориентированном подходе к разработке численного математического обеспечения // Вопросы кибернетики. Приложения системного программирования / Под ред. В.П. Иванникова. – М.: Науч. совет по комплексной проблеме «Кибернетика» РАН, 1995. – С. 140-163.
168. Семенов В.А. Объектная систематизация и парадигмы вычислительной математики // Программирование. – 1997. – №4. – С. 14-25.
169. Семенов В.А. Объектно-ориентированная методология эволюционной разработки математического обеспечения. Дисс. … доктора физ.-мат. наук. – М.: ИСП РАН, 1998.
170. Семенов В.А., Тарлапан О.А. Методика разработки библиотеки шаблонов BLAS для разреженных матриц // Вопросы кибернетики. Приложения системного программирования / Под ред. В.П. Иванникова. – М.: Науч. совет по комплексной проблеме «Кибернетика» РАН. – 1997. – Вып. 3. – С. 227-239.
171. Семенов В.А., Тарлапан О.А. Объектно-ориентированный подход к программированию прямых методов линейной алгебры // Вопросы кибернетики. Приложения системного программирования / Под ред. В.П. Иванникова. – М.: Науч. совет по комплексной проблеме «Ки-

бернетика» РАН, 1996. – Вып. 2. – С. 147-170.

172. Семенов В.А., Тарлапан О.А. Технологии реализации разреженных матричных классов // Вопросы кибернетики. Приложения системного программирования / Под ред. В.П. Иванникова. – М.: Науч. совет по комплексной проблеме «Кибернетика» РАН, 1995. – С. 164-188.
173. Семенов В.А., Ширяева Е.Ю. Объектная классификация задач и методов нелинейной безусловной оптимизации // Вопросы кибернетики. Приложения системного программирования / Под ред. В.П. Иванникова. – М.: Науч. совет по комплексной проблеме «Кибернетика» РАН. – 1996. – Вып. 2. – С. 86-119.
174. Семеріков С.О., Завізєна Н.С. Розробка та використання електронних довідників з інформатики // Всеукраїнська конференція молодих науковців «Інформаційні технології в науці та освіті». – Черкаси: ЧДУ. – 1997. – С. 49.
175. Семеріков С.О., Завізєна Н.С. Розробка та використання електронних довідників з інформатики // Матеріали Всеукраїнської конференції молодих науковців «Інформаційні технології в науці та освіті». – Частина 1. – Черкаси: ЧДУ. – 1997. – С. 212-224.
176. Семеріков С.О. Методика викладання чисельних методів у об’єктній методології // Тезисы докладов научно-методического семинара «Информационные технологии в учебном процессе». – Одесса: ЮУГПУ им. К.Д. Ушинского. – 1999. – С. 24.
177. Семеріков С.О. Застосування об’єктно-орієнтованого програмування до викладання курсу чисельних методів // Математика, її застосування та викладання: Матеріали міжвузівської регіональної наукової конференції. – Кіровоград: РВГ ІЦ КДПУ ім. В. Винниченка. – 1999. – С. 134-137.
178. Семеріков С.О. Об’єктно-орієнтований підхід як засіб активізації пізнавальної діяльності // Творча особистість учителя: проблеми теорії

і практики. Збірник наукових праць. Випуск третій. – К.: НПУ ім. М.П. Драгоманова, 1999. – С. 69-75.

179. Семериков С.А. Объектный подход к преподаванию численных методов // VII Международная конференция «Математика. Компьютер. Образование», г. Дубна, 23-30 января 2000 г. Тезисы. – М.: Прогресс-Традиция, 1999. – С. 291.
180. Семеріков С.О. Чисельні методи: об’єктний підхід // Комп’ютерно-орієнтовані системи навчання. Збірник наукових праць. Випуск 2. – К.: НПУ ім. М.П. Драгоманова, 2000. – С. 122-128.
181. Сенько Ю.В. Формирование научного стиля мышления учащихся. – М.: Знание, 1986. – 80 с.
182. Скарборо Д.Г. Численные методы математического анализа. – М.-Л.: Гос. техн.-теоретич. изд., 1934. – 440 с.
183. Скаткин М.Н. Активизация познавательной деятельности учащихся в обучении. – М.: АПН РСФСР, 1965. – 48 с.
184. Слєпкань З.І. Наукові засади педагогічного процесу у вищій школі. Конспект лекцій. – К.: НПУ ім. М.П. Драгоманова, 1999. – 150 с.
185. Слепкань З.И. Психолого-педагогические основы обучения математике: Метод. пособие. – К.: Рад. школа, 1983. – 192 с.
186. Соловйов В.М., Семеріков С.О., Теплицький І.О. Інструментальне забезпечення курсу комп’ютерного моделювання // Комп’ютер у школі і сім’ї. – 2000. – № 4. – С. 28-31.
187. Сорокин А.С. Техника счета (Методы рациональных вычислений). – М.: Знание, 1976. – 120 с.
188. Спиваковский А. Педагогические программные средства: объектно-ориентированный подход // Информатика и образование. – 1990. – № 2. – С. 71-73.
189. Справочник по классам Borland C++ 3.1-4.0 /Под ред. Дериева И.И. – К.: Диалектика, 1994. – 256 с.

190. Страуструп Б. Язык программирования С++: В 2-х частях. – К.: Диасофт, 1993.
191. Сырнев Н.И. Вычислительная культура в средней школе: Автореферат дисс. … кандидата пед. наук. / Акад. пед. наук. РСФСР. Науч.-исслед. ин-т общего и политехн. образования, 1961. – 9 с.
192. Талызина Н.Ф. Технология обучения и ее место в педагогическом процессе // Современная высшая школа. – 1977. – Т. 1 (17).
193. Талызина Н.Ф. Управление процессом усвоения знаний. – М.: Изд-во МГУ, 1975. – 343 с.
194. Тарлапан О.А. Исследование и разработка объектно-ориентированного матричного обеспечения. Дисс. … кандидата физ.-мат. наук. – М.: ИСП РАН, 1998.
195. Телло Э. Объектно-ориентированное программирование в среде Windows. – М.: Наука-Уайли, 1993. – 347 с.
196. Теория и практика педагогического эксперимента / Под ред. А.И. Пискунова. – М.: Педагогика, 1979. – 208 с.
197. Токарь С., Штонда В. Объектно-ориентированный анализ для программистов // Soft Review. – 1993. – Октябрь. – С. 3-8.
198. Томащук О.П. Професійна спрямованість викладання математичного аналізу в умовах диференційованої підготовки вчителя математики: Дис. … канд. пед. наук: 13.00.02 / НПУ ім. М.П. Драгоманова – К., 1999. – 247 с.
199. Тополянский Д.Б. Зв’язок між математикою та виробництвом в сучасній школі. – Харків: Радянська школа, 1932. – 76 с.
200. Турчин В.М. Математична статистика. Навч. посіб. – К.: Видавничий центр «Академія», 1999. – 240 с.
201. Тыугу Э., Мацкин М., Мериcте М., Пеньям Я., Шмундак А. Парадигма объектно-ориентированного программирования // Всесоюзная конференция «Актуальные проблемы современного программирова-

ния». Секция «Объектно-ориентированное программирование» / АН Эстонии, Ин-т кибернетики. – Таллинн. – 1990. – С. 18-24.

202. Тюхтин В.С. О природе образа: Психическое отражение в свете идеи кибернетики. – М.: Высшая школа, 1963. – 123 с.
203. Уилкинсон Дж., Райнш К. Справочник алгоритмов на языке Алгол. Линейная алебра. – М.: Машиностроение, 1976.
204. Уткіна С.В., Наришкіна Л.С. Алгебра і числові системи: Навч. посібник. – К.: Вища школа, 1995. – 304 с.
205. Файнерман І. Виробнича математика: Підр. для шкіл, ФЗУ. – Харків-К.: Держтехвидав, 1931. – 304 с.
206. Файнерман І. Збірник виробничо-математичних завдань. – Вип. ІІ. – Харків-К.: Держ. вид. України, 1930. – 145 с.
207. Фейсон Т. Объектно-ориентированное программирование на Borland C++ 4.5. – К.: Диалектика, 1996. – 824 с.
208. Фільчаков П.Ф. Математичний практикум. Обчислення: Посібник для фізико-математичних факультетів пед. ін-тів УРСР. – К.: Рад. школа, 1958. – 278 с.
209. Фихтенгольц Г.М. Математика для инженеров. – М.: ГТТИ, 1933. – Ч. 1, 2.
210. Форсайт Д., Малькольм М., Моллер К. Машинные методы математических вычислений. – М.: Мир, 1980. – 320 с.
211. Хараджян А.А. Использование объектно-ориентированного подхода для моделирования электромеханических систем // Комп’ютерне моделювання та інформаційні технології в природничих науках. – Кривий Ріг: Видавничий відділ КДПУ, 2000. – С. 137-140.
212. Хараджян А.А. Использование объектно-ориентированного подхода для идентификации динамических систем // Комп’ютерне моделювання та інформаційні технології в природничих науках. – Кривий Ріг: Видавничий відділ КДПУ, 2000. – С. 137-140.

213. Хараджян А.А., Семериков С.А., Завизена Н.С. Использование алгебраических структур при моделировании атомных процессов в физике твёрдого тела (факультативный курс) // Международная научно-практическая конференция «ELBRUS'97 Новые информационные технологии и их региональное развитие». Тезисы докладов. – Нальчик: Кабардино-Балкарский государственный университет им. Х.М. Бербекова. – 1998. – С. 87-89.
214. Харламов И.Ф. Активизация учения школьников. – Минск: Народная асвета, 1970. – 158 с.
215. Хаусхолдер А.С. Основы численного анализа. – М.: Изд. иностр. лит., 1956. – 320 с.
216. Хемминг Р.В. Численные методы для научных работников и инженеров. – М.: Наука, 1968. – 400 с.
217. Цибко Г.Ю. Підвищення рівня теоретичної підготовки з інформатики на фізико-математичних факультетах педагогічних вузів: Дис. … канд. пед. наук: 13.00.02 / НПУ ім. М.П. Драгоманова. – К., 1998. – 200 с.
218. Черных Л.А. Теоретические основы разработки методической системы обучения // Евристика та дидактика точних наук: Збірник наукових робіт. – Вип. 3. – Донецьк: Донецька школа евристики та точних наук, 1995. – С. 15-19.
219. Шамова Т.И. Активизация учения школьников. – М.: Педагогика, 1982. – 208 с.
220. Швець В.О. Оновлення методичної системи навчання математики // Проблеми навчання математики в університеті й школі: Тези доповідей науково-методичної конференції математичного факультету. – Донецьк: Донецький державний університет, 1994. – С. 3-6.
221. Шиманский С.В., Шиманский Ю.А. Принципы числовых расчетов. Теория и практика ведения числовых расчетов. – СПб., 1909. – 70 с.

222. Широчин В.П. Моделювання на ЕОМ. Конспект лекцій для студентів спец. 22.01 «Обчислювальні машини, комплекси, системи і мережі» всіх форм навчання. – К.: КПІ, 1994. – 64 с.
223. Шкіль М.І., Слєпкань З.І., Дубинчук О.С. Алгебра і початки аналізу: Проб. підруч. для 10-11 класів серед. шк. – К.: Зодіак-ЕКО, 1995. – 608 с.
224. Шлеер С., Меллор С. Нотация для объектно-ориентированного проектирования (OOD), не зависимая от языка программирования // Soft Review. – 1993. – Декабрь. – С. 3-9.
225. Шлеер С., Меллор С. Объектно-ориентированный анализ: моделирование мира в состояниях. – К.: Диалектика, 1993. – 240 с.
226. Шмулевич П.К. Прикладная математика. – Ч. 1. – Л.: Прибой, 1931. – 236 с.
227. Шуп Т.Е. Прикладные численные методы в физике и технике. – М.: Высшая шк., 1990. – 254 с.
228. Щукина Г.И. Активизация познавательной деятельности учащихся в учебном процессе: Учебное пособие для студентов пед. ин-тов. – М.: Просвещение, 1979. – 160 с.
229. Щукина Г.И. Проблема познавательного интереса в педагогике. – М.: Педагогика, 1971. – 352 с.
230. Элементы вычислительной математики. Под. ред. С.Б. Норкина. – М.: Высшая школа, 1960. – 164 с.
231. Эсаулов А.Ф. Активизация учебно-познавательной деятельности студентов. – М.: Высшая школа, 1988. – 223 с.
232. Юрченко В.А., Семериков С.А. Эффективное использование ресурсов компьютера для решения прикладных задач (факультативный курс) // Комп’ютерне моделювання та інформаційні технології в природничих науках. – Кривий Ріг: Видавничий відділ КДПУ, 2000. – С. 270-274.

233. Aberth O., Schaefer M.J. Precise Computation Using Range Arithmetics, via C++ // j-TOMS. – 1992. – Vol. 18, No. 4. – P. 481-491.
234. Arndt J. Hfloat, a C++ library for high precision computations. – http://www.jjj.de/hfloat/, 1997. – 14 p.
235. Arndt J. Remarks on arithmetical algoritms and the computation of π. – http://www.jjj.de/hfloat/, 1997. – 55 p.
236. Barrett R., Berry M., Chan T.F., Demmel J., Donato J., Dongarra J., Eijkhout V., Pozo R., Romine C., Van der Vorst H. Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods. – 2nd Edition. – Philadelphia: SIAM, 1994. – 124 p.
237. Budge K.G. Just What is Object-Oriented Numerics? – Sandia National Laboratories: http://www.sandia.gov/OOP/, 1994. – 20 p.
238. Budge K.G. Effective Numerical C++. – Sandia National Laboratories: http://www.sandia.gov/OOP/, 1995. – 63 p.
239. Cox B.J. Object-oriented programming. – NY: Addison-Wesley, 1987. – 374 p.
240. Goldberg A. Programmer as a reader. – IEEE Software. – 1987. – Vol. 9. – P. 62-70.
241. Haible B. CLN, a Class Library for Numbers. – http://clisp.cons.org/~haible/packages-cln.html, 1999. – 70 p.
242. Haible B., Papanikolaou T. Fast multiprecision evaluation of series of rational numbers. // IMACS/ACA and ANTS III proceedings. – Honolulu. – 1997. – 8 p.
243. Haney S. Is C++ Fast Enough for Scientific Computing? // Computers in Physics. – 1994. – Vol. 8, No. 6. – P. 690-694.
244. Horstmann C.S. C++ class libraries for numerical programming // C++ Reports. – 1996. – Vol. 8, No. 1. – P. 61-66.
245. Jacky J.P., Kalet I.J. An object-oriented programming discipline for standard Pascal. – Comm. ACM. – 1987. – Vol. 30, No. 9. – P. 772-786.

246. Keefer T. Why C++ Will Replace Fortran // Dr. Dobb's Journal of Software Tools. – 1992. – Vol. 17, No. 12. – P. 39-47.
247. Machiels L., Deville M.O. Fortran 90: An Entry to Object-Oriented Programming for Solution of Partial Differential Equations // ACM Transactions on Mathematical Software. – 197. – Vol. 23, No. 1. – P. 32-49.
248. McDonald J.A. Object-oriented programming for linear algebra // ACM SIG-PLAN Notices. – 1989. – Vol. 24, No. 10. – P. 175-184.
249. Meyer B. Reusability: the case of object-oriented design. – IEEE Software. – 1987. – Vol. 3. – P. 50-64.
250. Miller G. The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information // The Psychological Review. – 1956. – Vol. 63 (2). – P. 86.
251. Miller G.R. An object-oriented approach to structural analysis and design // Computers and Structures. – 1991. – Vol. 40, No. 1. – P. 75-82.
252. Musser D.R., Stepanov A.A. Algorithm-oriented Generic Libraries // Software: Practice and Experience. – 1994. – Vol. 24, No. 7. – P. 632-642.
253. Pidaparti R.M.V., Hudli A.V. Dynamic analysis of structures using object-oriented techniques // Computers and Structures. – 1993. – Vol. 49, No. 1. – P. 149-156.
254. Pozo R., Remington K.A. C++ Programming for Scientist. – Gaithersburg: NIST, 1998. – 115 p.
255. Press W.H., Teukolsky S.A., Vetterling W.T., Flannery B.P. Numerical Recipes in C: The Art of Scientific Computing, 2nd Edition. – Cambridge–New York: Cambridge University Press, 1997. – 1009 p.
256. Robinson A.D. C++ Gets Faster for Scientific Computing // Computers in Physics. – 1996. – Vol. 10. – P. 458-462.
257. Saunders J. A Survey of Object-oriented Programming Languages // Journal of Object-oriented Programming. – March/April 1989. – Vol. 1(6).

258. Stefik M., Bobrow D. Object-oriented Programming: Themes and Variations. – AI Magazine. – 1986. – Vol. 6(4). – P. 41.
259. Stroustrup B. What is object-oriented programming? – IEEE Software. – 1988. – Vol. 5. – P. 10-20.
260. Sullivan S.J., Zorn B.G. Numerical Analysis Using Nonprocedural Paradigms // ACM Transactions on Mathematical Software. – 1995. – Vol. 21, No. 3. – P. 267-298.
261. The Object-Oriented Numerics Page. – http://oonumerics.org, 2000.
262. Tommila M. APFLOAT. A C++ High Performance Arbitrary Precision Arithmetic Package. Version 1.50. – http://www.iki.fi/~mtommila/apfloat/, 1998. – 35 p.
263. Veldhuizen T. Techniques for Scientific C++. – Indiana University Computer Science Department: http://seurat.uwaterloo.ca/blitz/papers/techtalk.ps, 1998. – 80 p.
264. Veldhuizen T. Expression Templates // C++ Report. – 1995. – Vol. 7, No. 5. – P. 26-31.
265. Veldhuizen T. Using C++ template metaprograms // C++ Report. – 1995. – Vol. 7, No. 4. – P. 36-43.
266. Veldhuizen T., Ponnambalam K. Linear algebra with C++ template metaprograms // Dr. Dobb's Journal of Software Tools. – 1996. – Vol. 21, No. 8. – P. 38-44.
267. Wong M.K.W., Budge K.G., Peery J.S., Robinson A.C. Object-Oriented Numerics: A Paradigm to Numerical Object-Oriented Programming // Computers In Physics. – 1993. – Vol. 7, No. 6. – P. 655-663.
268. Zeglinski G.W., Han R.S., Aitchison P. Object-oriented matrix classes for use in a finite element code using C++ // International Journal for Numerical Methods in Engineering. – 1994. – Vol. 37. – P. 3921-3937.

ДОДАТКИ

Додаток А. Програмна реалізація типу «арифметичний вектор»

```
#ifndef __VECTOR_H
#define __VECTOR_H
#ifndef __FSTREAM_H
#include <fstream.h>
#endif
#ifndef __IOMANIP_H
#include <iomanip.h>
#endif
#ifndef __STDLIB_H
#include <stdlib.h>
#endif
#ifndef __MATH_H
#include <math.h>
#endif
#ifndef __EXCEPT_H
#include <except.h>
#endif
#ifndef __CSTRING_H
#include <cstring.h>
#endif

/*параметризований клас для роботи з векторними об'єктами */

template <class YourOwnFloatType>/* підставте свій тип */
class vector
{ //приватні дані
  long m;             //розмірність (довжина) вектора
  YourOwnFloatType *vec; /*покажчик на елементи вектора */
  /*віртуальні функції читання з потоку і запису в потік; їх необхі-
дно перевизначити в похідних класах для забезпечення можливості ви-
користання єдиних перевантажених операторів << і >> */
  virtual void In(istream &);
  virtual void Out(ostream &);
 public://загальнодоступні дані і функції
  /*завантаження вектора з файлу: dimension data1 data2... */
  vector(char *);
  vector();/*створення порожнього вектора одиничної розмірності */
  vector(long);/*створення порожнього вектора заданої розмірності */
  vector(long, YourOwnFloatType *); /*створення вектора заданої роз-
мірності, заповнюваного даними з масиву */
  //конструктор копіювання
  vector(vector<YourOwnFloatType> &);
  ~vector();//деструктор
  friend vector<YourOwnFloatType> operator+ (vector
<YourOwnFloatType> &, vector<YourOwnFloatType> &); //додавання двох
векторів
  friend vector<YourOwnFloatType> operator+= (vector
<YourOwnFloatType> &, vector <YourOwnFloatType> &);//додавання з
```

```
присвоюванням
  friend vector<YourOwnFloatType> operator-(vector
<YourOwnFloatType> &, vector<YourOwnFloatType> &); //віднімання
  friend vector<YourOwnFloatType> operator-= (vector
<YourOwnFloatType> &, vector <YourOwnFloatType> &);//віднімання з
присвоюванням
  friend YourOwnFloatType operator* (vector <YourOwnFloatType> &,
vector<YourOwnFloatType> &); //скалярне множення
  friend vector<YourOwnFloatType> operator* (YourOwnFloatType,
vector<YourOwnFloatType> &); //множення числа на вектор
  friend vector<YourOwnFloatType> operator*
(vector<YourOwnFloatType> &, YourOwnFloatType ); //множення вектора
на число
  friend vector<YourOwnFloatType> operator*=
(vector<YourOwnFloatType> &, YourOwnFloatType ); //множення вектора
на число з присвоюванням
  friend ostream &operator<<(ostream &, vector <YourOwnFloatType>
&);//висновок вектора в потік
  friend istream &operator>>(istream &, vector <YourOwnFloatType>
&);//уведення вектора з потоку
  vector<YourOwnFloatType> operator= (vector <YourOwnFloatType>
&);//присвоювання
  vector<YourOwnFloatType> operator-();/*унарний мінус*/
  vector<YourOwnFloatType> operator+();/*унарний плюс*/
  vector<YourOwnFloatType> operator~(); /*нормування (визначення на-
правляючих косинусів) */
  YourOwnFloatType operator!();//модуль вектора
  virtual long IsEqual(void *);
/*ця віртуальна функція порівнює поточний вектор з об'єктом, що ле-
жить за адресою, переданому через узагальнений покажчик */
  friend long operator==(vector<YourOwnFloatType> &,
vector<YourOwnFloatType> &);/*перевірка на рівність */
  friend long operator!=(vector<YourOwnFloatType> &,
vector<YourOwnFloatType> &);/*перевірка на нерівність */
  YourOwnFloatType &operator[](long a);
     //індексування елементів вектора
  long getm() { return m; }//розмірність вектора
};

/*Тепер реалізація оголошених методів класу vector */

//індикатор помилки - деяка константа
const long double MAX_LONGDOUBLE=1.7976931348e308;

/*
  Створювати вектор можна по-різному. Наприклад, якщо він знаходить-
ся на зовнішньому пристрої у форматі m d1 d2 ... dm, де m - розмір-
ність вектора, а di - його компоненти, то маємо наступний конструк-
тор:
*/
template <class YourOwnFloatType>
vector<YourOwnFloatType>::vector(char *f) /*ім'я файлу */
{
  long i;

  ifstream fp=f;//намагаємося відкрити файл
  if(!fp)//якщо не вдалося
```

```
    throw xmsg("Не можу відкрити файл "+string(f)+ "\n");
  fp>>m;//уводимо розмірність
  if(m<=0)//перевірка на коректність
    throw xmsg("Розмірність вектора некоректна \n");
//діагностика
  try
  {
   vec=new YourOwnFloatType[m];/*спроба виділення пам'яті */
  }
  catch(xalloc)
  {
    throw xmsg("Не вистачає пам'яті \n");
  }
  for(i=0;i<m&&fp>>vec[i];i++);/*зчитування з файлу */
}


/*
  У випадку, коли нам відома лише розмірність вектора, але невідомі
його складові, припускаємо, що даний вектор є нульовим:
*/
template <class YourOwnFloatType>
vector<YourOwnFloatType>::vector(long a):m(a)
//розмірність вектора
{
  long i;

  if(m<=0)//перевірка розмірності
    throw xmsg("Розмірність вектора некоректна \n");
  //діагностика
  try
  {
   vec=new YourOwnFloatType[m];/*спроба виділення пам'яті */
  }
  catch(xalloc)
  {
    throw xmsg("Не вистачає пам'яті \n");
  }
  for(i=0;i<m;vec[i++]=0);/*обнуління компонентів вектора */
}


/*
  Нарешті, нам можуть бути відомі як розмірність, так і компоненти
вектора:
*/
template <class YourOwnFloatType>
vector<YourOwnFloatType>::vector(long a,
YourOwnFloatType *v):m(a)
/*цей конструктор приймає розмір і покажчик на дані */
{
  long i;

  if(m<=0)//перевірка розмірності
    throw xmsg("Розмірність вектора некоректна \n");
  //діагностика
  try
```

```
  {
   vec=new YourOwnFloatType[m];/*спроба виділення пам'яті */
  }
  catch(xalloc)
  {
    throw xmsg("Не вистачає пам'яті \n");
  }
  for(i=0;i<m;i++)
    vec[i]=v[i];/*копіювання з зовнішнього масиву у вектор */
}


/*
  Є ще один випадок, коли ми нічого не можемо сказати про розмір-
ність і компоненти вектора - при створенні масиву векторів, тобто
матриці, коли для оператора new потрібно конструктор без параметрів
чи коли розмір вектора заздалегідь невідомий.
*/
template <class YourOwnFloatType>
vector<YourOwnFloatType>::vector():m(1)
{
  try
  {
   vec=new YourOwnFloatType[m];/*спроба виділення пам'яті */
  }
  catch(xalloc)
  {
    throw xmsg("Не вистачає пам'яті \n");
  }
  *vec=0;//обнуляємо єдиний наявний елемент
}


/*
  У реальних розрахунках можуть використовуватися вектори великих
розмірностей, тому розміщаються вони у вільній пам'яті комп'ютера, а
коли необхідність у них відпадає - знищуються.
*/
template <class YourOwnFloatType>
vector<YourOwnFloatType>::~vector()
{
  delete []vec;//знищення динамічного масиву
}


/*
  Необхідність в індексації вектора виникає в двох випадках: при
одержанні складової вектора по її номері і при зміні не усього век-
тора, а тільки однієї його складової.
  При цьому, звичайно, варто враховувати можливість помилкового за-
вдання номера складової: припустимий діапазон значень [0,m).
*/
template <class YourOwnFloatType>
YourOwnFloatType & vector<YourOwnFloatType>::operator[](long a)
{
  static YourOwnFloatType error=MAX_LONGDOUBLE;
  if(a>=0&&a<m)//якщо все гаразд
```

```
    return vec[a];
  else//при виході за межі вектора лаємося
  {
    cerr<<"Індекс "<<a<<" поза діапазоном вектора\n";
    return error;
  }
}


/*
  Створюючи вектор, можна попутно ініціювати його даними з вже існу-
ючого:
*/
template <class YourOwnFloatType>
vector<YourOwnFloatType>::vector( vector<YourOwnFloatType> &ex) :
m(ex.m)
/*це конструктор копіювання, що приймає посилання на вектор */
{
  try
  {
   vec=new YourOwnFloatType[m];/*спроба виділення пам'яті */
  }
  catch(xalloc)
  {
    throw xmsg("Не вистачає пам'яті \n");
  }
  for(long i=0;i<m;i++)
    vec[i]=ex[i];
  /*тут при копіюванні ex використовується вже індексація */
}


/*
  Додавання векторів є алгебраїчною операцією тільки тоді, коли век-
тора однакової розмірності. Результатом додавання є вектор тієї ж
розмірності, що і вихідні, компонентами якого є сума відповідних
компонентів вихідних векторів.
*/
template <class YourOwnFloatType>
vector<YourOwnFloatType> operator+
(vector<YourOwnFloatType> &f, vector<YourOwnFloatType> &s)
{
  if(f.m!=s.m)//перевірка на рівність розмірностей
    throw xmsg("Вектори різних довжин складати не можна \n");
  //діагностика
  vector<YourOwnFloatType> temp(f.m);
  //створюємо тимчасовий вектор
  /*тут працюють операції індексування для всіх трьох векторів */
  for(long i=0;i<f.m;i++)
    temp[i]=f[i]+s[i];
  return temp;//повертаємо результуючий вектор
}


//скорочена операція "додавання з присвоюванням"
template <class YourOwnFloatType>
vector<YourOwnFloatType> operator+=
```

```
(vector<YourOwnFloatType> &f, vector<YourOwnFloatType> &s)
{
  return f=f+s;
}


/*
  Введемо декілька допоміжних унарних операцій:
- "мінус":
*/
template <class YourOwnFloatType>
vector<YourOwnFloatType> vector<YourOwnFloatType>:: operator-()
{
  vector<YourOwnFloatType> temp(m);
  //створюємо тимчасовий вектор
  /*Якщо this - це покажчик на поточний об'єкт векторного класу, то
*this - це сам поточний об'єкт класу vector, тобто той, з яким ми
зараз працюємо. А до будь-якого векторного об'єкта ми можемо засто-
сувати операцію індексування */
  for(long i=0;i<m;i++)
    temp[i]=-(*this)[i];
  //temp[i]=-vec[i];
  //temp.vec[i]=-vec[i];
  //temp.operator[](i)=-operator[](i); etc...
  return temp;//повертаємо результуючий вектор
}


//унарний плюс
template <class YourOwnFloatType>
vector<YourOwnFloatType> vector<YourOwnFloatType>:: operator+()
{
  return *this;//повертаємо самого себе
}


/*
  Операція, що алгебраїчної назвати не можна - це, скоріше, приклад
дуже розповсюдженого тернарного відношення "скалярний добуток двох
векторів":
*/
template <class YourOwnFloatType>
YourOwnFloatType operator*(vector<YourOwnFloatType> &f,
vector<YourOwnFloatType> &s)
{
  if(f.m!=s.m)
    throw xmsg("Множення векторів з неспівпадаючими розмірами немож-
ливе \n"); //діагностика
  YourOwnFloatType temp=0;
  for(long i=0;i<f.m;i++)
    temp+=f[i]*s[i];
    //підсумовуємо добутку складових векторів
  return temp;
}


/*
```

```
  Модуль вектора як квадратний корінь скалярного добутку вектора на
самого себе:
*/
template <class YourOwnFloatType> inline YourOwnFloatType
vector<YourOwnFloatType>:: operator!()
{
  return sqrt((*this)*(*this));
}


/*
  нормування вектора за модулем
*/
template <class YourOwnFloatType>
vector<YourOwnFloatType> vector<YourOwnFloatType>::operator~()
{
  vector<YourOwnFloatType> temp(m);
  /*скалярний добуток поточного об'єкта на самого себе */
  YourOwnFloatType modul=!(*this);
  for(long i=0;i<m;i++)
    temp[i]=(*this)[i]/modul;/* направляючі косинуси */
  return temp;
}


/*
  множення числа на вектор:
*/
template <class YourOwnFloatType>
vector<YourOwnFloatType> operator*
(YourOwnFloatType ld, vector<YourOwnFloatType> &v)
{
  vector<YourOwnFloatType> temp=v;
  for(long i=0;i<v.getm();i++)
    temp[i]=temp[i]*ld;/* скоріше, це навіть "подовження" вектора */
  return temp;
}


/*
  множення вектора на число:
*/
template <class YourOwnFloatType>
inline vector<YourOwnFloatType> operator*
(vector<YourOwnFloatType> &v, YourOwnFloatType ld)
{
  return ld*v; /*викликали вже визначену функцію */
}


//операція скороченого множення вектора на число
template <class YourOwnFloatType>
vector<YourOwnFloatType> operator*=
(vector<YourOwnFloatType> &v, YourOwnFloatType ld)
{
  return v=v*ld;
}
```

```
/*
  Маючи визначені бінарну операцію додавання векторів і унарну одер-
жання вектора, протилежного до даного, можна векторною мовою, не
звертаючи до компонентів векторів, визначити операцію віднімання:
*/
template <class YourOwnFloatType>
vector<YourOwnFloatType> operator-
(vector<YourOwnFloatType> &f, vector<YourOwnFloatType> &s)
{
  return f+(-s);
  //return operator+(f,-s);
  //return operator+(f,s.operator-());
}


//операція скороченого віднімання
template <class YourOwnFloatType>
vector<YourOwnFloatType> operator-=
(vector<YourOwnFloatType> &f, vector<YourOwnFloatType> &s)
{
  return f=f-s;
}


/*
  При переписуванні одного вектора в інший можливі два випадки:
  1. якщо розмірність обох векторів збігається, то просто заміняємо
складові першого вектора компонентами другого;
  2. у противному випадку знищуємо перший вектор і створюємо знову,
використовуючи другий як будівельний матеріал.
*/
template <class YourOwnFloatType>
vector<YourOwnFloatType> vector<YourOwnFloatType>::
operator=(vector<YourOwnFloatType> &x)
{
  if(m!=x.m)//якщо розміри не збігаються
  {
    delete []vec; /*знищуємо вміст поточного вектора */
    m=x.m;//установлюємо новий розмір
    try
    {
      vec=new YourOwnFloatType[m];/*спроба виділення пам'яті */
    }
    catch(xalloc)
    {
      throw xmsg("Не вистачає пам'яті \n");
    }
  }
  for(long i=0;i<m;i++)
    vec[i]=x[i];/*копіюємо дані з вектора x у поточний */
/*присвоювання - це бінарна операція, першим параметром якої є об'-
єкт, якому привласнюють, другим - об'єкт, що привласнюють. При цьому
перший об'єкт, на відміну від всіх інших бінарних операцій, міняєть-
ся, і він же повертається як результат (це буває необхідним для опе-
рацій виду a=b=c;)*/
```

```
  return *this;
}


/*Ця функція порівнює поточний вектор з вектором, що лежить за адре-
сою x. Для кожного класу, похідного від векторного, не має сенсу пе-
ревизначати операторні функції перевірки на рівність і нерівність -
досить перевизначити цю віртуальну функцію */
template <class YourOwnFloatType>
long vector<YourOwnFloatType>::IsEqual(void *x)
{
  if(m!=((vector<YourOwnFloatType>*)x)->m)
    //при розбіжності розмірностей
    return 0; //констатуємо розбіжність векторів
  for(long i=0;i<m;i++)
    if((*this)[i]!=(*(vector<YourOwnFloatType>*)x)[i])
      return 0;//якщо хоч один елемент не збігся
  return 1;
}


/*
  Порівняння векторів є тернарним відношенням, результатом якого є
число нуль, якщо вектори не рівні й одиниця в противному випадку.
  Два вектори будемо вважати рівними, якщо вони мають однакові дов-
жини і їхні відповідні складові збігаються:
*/
template <class YourOwnFloatType>
long operator==(vector<YourOwnFloatType> &f,
vector<YourOwnFloatType> &s)
{
  return f.IsEqual(&s);
}


/*
  Нерівність векторів визначимо через рівність і операцію заперечен-
ня:
*/
template <class YourOwnFloatType> inline long
operator!=(vector<YourOwnFloatType> &f, vector<YourOwnFloatType> &s)
{
  return !(f==s);//логічне заперечення рівності
}


/*Потужний I/O-механізм C++ дозволяє в природній формі виводити
(уводити) вектори на будь-який пристрій відображення інформації. Для
універсалізації зчитування і запису вектора в потік знову удамося до
механізму віртуальних функцій. З цією метою, за аналогією з printOn,
визначимо дві функції - одну для введення, іншу - для виведення */

template <class YourOwnFloatType>
void vector<YourOwnFloatType>::In(istream &is)
{
  for(long i=0;i<m;i++)
    is>>(*this)[i];
```

```
}

template <class YourOwnFloatType>
void vector<YourOwnFloatType>::Out(ostream &os)
{
  for(long i=0;i<m;i++)
  {
    os.precision(100);
    os<<(*this)[i]<<" ";/*компоненти розділяємо пробілами */
  }
}

//виведення у потік
template <class YourOwnFloatType>
ostream &operator<<(ostream &os, vector<YourOwnFloatType> &x)
{
  x.Out(os);
  return os;
}

/*  - уведення з потоку */
template <class YourOwnFloatType>
istream &operator>>(istream &is, vector<YourOwnFloatType> &x)
{
  x.In(is);
  return is; /*приймаємо і повертаємо посилання на потік уведення */
}

#endif
```

Додаток Б. Програмна реалізація типу «поліном»

```
#ifndef __POLYNOM_H
#define __POLYNOM_H
#ifndef __VECTOR_H
#include "vector.h"
#endif
#ifndef __IOSTREAM_H
#include <iostream.h>
#endif
#define max(a,b)    (((a) > (b)) ? (a) : (b))
#define min(a,b)    (((a) < (b)) ? (a) : (b))

/*
 Параметризований клас для поліномів
 Внутрішній формат:
 a0+a1*x+a2*x^2+a3*x^3+a4*x^4+a5*x^5+...+a(n-1)*x^(n-1)
 Введення і виведення виконуються, починаючи з коефіцієнта при най-
вищому степені
*/
//Наш поліном буде базуватися на векторі
template <class YourOwnFloatType>
class polynom: public vector<YourOwnFloatType>
{
  //ці функції є внутрішніми
  void optimize();/*перетворення полінома в канонічну форму */
  polynom<YourOwnFloatType> reverse();/*запис полінома в зворотному
порядку */
  void In(istream &);/*за аналогією з векторами - функції введення
*/
  void Out(ostream &);/*    і виведення */
 public:
  polynom(char *);//поліном з файлу
  polynom(long, YourOwnFloatType *);/*поліном з масиву */
  polynom(polynom<YourOwnFloatType> &); /*конструктор копіювання */
  polynom();//конструктор за замовчуванням
  polynom(long);//поліном заданого степеня
  polynom<YourOwnFloatType> operator-();/*унарний мінус */
  polynom<YourOwnFloatType> operator+();/*унарний плюс */
  friend polynom<YourOwnFloatType> operator+(polynom
<YourOwnFloatType> &, polynom<YourOwnFloatType> &); //додавання
  friend polynom<YourOwnFloatType> operator+= (polynom
<YourOwnFloatType> &, polynom <YourOwnFloatType> &); //скорочене до-
давання
  friend polynom<YourOwnFloatType> operator-(polynom
<YourOwnFloatType> &, polynom<YourOwnFloatType> &); //віднімання
  friend polynom<YourOwnFloatType> operator-= (polynom
<YourOwnFloatType> &, polynom <YourOwnFloatType> &); //скорочене
віднімання
  friend polynom<YourOwnFloatType> operator* (polynom
<YourOwnFloatType> &, polynom <YourOwnFloatType> &); /*множення по-
ліномa на поліном */
  friend polynom<YourOwnFloatType> operator*= (polynom
```

```
<YourOwnFloatType> &, polynom <YourOwnFloatType> &); /*скорочене
множення на поліном */
  friend polynom<YourOwnFloatType> operator* (YourOwnFloatType,
polynom<YourOwnFloatType> &); //множення числа на поліном
  friend polynom<YourOwnFloatType> operator*(polynom
<YourOwnFloatType> &, YourOwnFloatType); //множення полінома на чис-
ло
  friend polynom<YourOwnFloatType> operator*= (polynom
<YourOwnFloatType> &, YourOwnFloatType);//скорочене множення на чис-
ло
  //набір операцій для порівняння поліномів
  friend long operator<(polynom<YourOwnFloatType> &,
polynom<YourOwnFloatType> &);
  friend long operator>(polynom<YourOwnFloatType> &,
polynom<YourOwnFloatType> &);
  friend long operator<=(polynom<YourOwnFloatType> &,
polynom<YourOwnFloatType> &);
  friend long operator>=(polynom<YourOwnFloatType> &,
polynom<YourOwnFloatType> &);
  YourOwnFloatType &operator[](long);/*індексація полінома */
  YourOwnFloatType operator()(YourOwnFloatType); /*значення полінома
в заданій точці */
  friend long div(polynom<YourOwnFloatType> &,
polynom<YourOwnFloatType>&, polynom<YourOwnFloatType> &,
polynom<YourOwnFloatType> &);//ділення з остачею
  friend polynom<YourOwnFloatType> operator/(polynom
<YourOwnFloatType> &, polynom <YourOwnFloatType> &); //ціла частина
від ділення
  friend polynom<YourOwnFloatType> operator%(polynom
<YourOwnFloatType> &, polynom<YourOwnFloatType> &); //остача від ді-
лення
  long IsEqual(void *);//перевірка на рівність
  polynom<YourOwnFloatType> operator=
(polynom<YourOwnFloatType> &);//присвоювання
  friend polynom<YourOwnFloatType> derive
(polynom<YourOwnFloatType>,long);//похідна
  friend polynom<YourOwnFloatType> integral
(polynom<YourOwnFloatType>,long);//інтеграл
  friend polynom<YourOwnFloatType> pow
(polynom<YourOwnFloatType>,unsigned int);//ступінь
  polynom<YourOwnFloatType> operator^(unsigned int); /*степінь як
операція */
  polynom<YourOwnFloatType> operator^=(unsigned int); /*скорочений
степінь */
};


/*конструктори полінома будуть аналогічні конструкторам вектора */
//поліном з файлу
template <class YourOwnFloatType>
polynom<YourOwnFloatType>:: polynom(char *f):
vector<YourOwnFloatType>(f)
{
}


//поліном степеня a-1
```

```
template <class YourOwnFloatType>
polynom<YourOwnFloatType>::polynom(long a):
vector<YourOwnFloatType>(a)
{
}


//нуль-поліном
template <class YourOwnFloatType>
polynom<YourOwnFloatType>::polynom(): vector<YourOwnFloatType>()
{
}


/*поліном (a-1)-го степеня з коефіцієнтами з масиву v*/
template <class YourOwnFloatType> polynom <YourOwnFloatType>::
polynom(long a, YourOwnFloatType *v): vector<YourOwnFloatType>(a,v)
{
}


//конструктор копіювання
template <class YourOwnFloatType> polynom <YourOwnFloatType>::
polynom(polynom<YourOwnFloatType> &ex): vector<YourOwnFloatType>(ex)
{
}


//для індексації полінома викликаємо відповідний метод векторного
класу
template <class YourOwnFloatType> YourOwnFloatType
&polynom<YourOwnFloatType>::operator[](long a)
{
  return (*(vector<YourOwnFloatType>*)this)[a];
}


//обчислення значення полінома в точці x
template <class YourOwnFloatType> YourOwnFloatType
polynom<YourOwnFloatType>:: operator()(YourOwnFloatType x)
{
/*
  YourOwnFloatType temp=0,px=1;
  for(long i=0;i<getm();i++,px*=x)
    temp+=(*this)[i]*px;
  return temp;
*/
  polynom<YourOwnFloatType> temp=2;
  temp[0]=-x, temp[1]=1;
  return ((*this)%temp)[0];
}


//унарний мінус
template <class YourOwnFloatType>
polynom<YourOwnFloatType> polynom<YourOwnFloatType>:: operator-()
{
```

```
  return *(polynom*) &(-(*(vector<YourOwnFloatType>*) this));
}


//унарний плюс - повертаємо самого себе
template <class YourOwnFloatType>
polynom<YourOwnFloatType> polynom<YourOwnFloatType>::operator+()
{
  return *this;
}


//додавання двох поліномів
template <class YourOwnFloatType> polynom <YourOwnFloatType>
operator+(polynom<YourOwnFloatType> &f, polynom<YourOwnFloatType>
&s)
{
  long ms=max(f.getm(),s.getm());
  polynom<YourOwnFloatType> temp(ms);
  //створюємо тимчасовий поліном
  /*тут працюють операції індексування для всіх трьох поліномів */
  for(long i=ms-1;i>=min(f.getm(),s.getm());i--)
    temp[i]=(f.getm()>s.getm())?f[i]:s[i];
  for(long i=min(f.getm(),s.getm())-1;i>=0;i--)
    temp[i]=f[i]+s[i];
  temp.optimize();
  /*поки є, видаляємо 0-коефіцієнт при старшому степені */
  return temp;
}


//скорочене додавання, виражене через звичайне
template <class YourOwnFloatType> polynom <YourOwnFloatType>
operator+=
(polynom <YourOwnFloatType> &f, polynom <YourOwnFloatType> &s)
{
  return f=f+s;
}


/*віднімання поліномів, виражене через додавання і заперечення */
template <class YourOwnFloatType> polynom<YourOwnFloatType>
operator-
(polynom <YourOwnFloatType> &f, polynom<YourOwnFloatType> &s)
{
  return f+(-s);
}


//скорочене віднімання
template <class YourOwnFloatType> polynom<YourOwnFloatType>
operator-=
(polynom <YourOwnFloatType> &f, polynom<YourOwnFloatType> &s)
{
  return f=f-s;
}
```

```
//множення поліномів
template <class YourOwnFloatType>
polynom<class YourOwnFloatType> operator*(polynom<YourOwnFloatType>
&f, polynom<YourOwnFloatType> &s)
{
  polynom<YourOwnFloatType> temp(f.getm()+s.getm());
  for(long i=0;i<f.getm();i++)
    for(long j=0;j<s.getm();j++)
      temp[i+j]+=f[i]*s[j];
  temp.optimize();
  return temp;
}


//скорочене множення
template <class YourOwnFloatType>
polynom<class YourOwnFloatType> operator*=(polynom<YourOwnFloatType>
&f, polynom<YourOwnFloatType> &s)
{
  return f=f*s;
}


//множення числа на поліном
template <class YourOwnFloatType> polynom<YourOwnFloatType>
operator*(YourOwnFloatType ld, polynom<YourOwnFloatType> &v)
{
  polynom<YourOwnFloatType> temp=v;
  temp=*(polynom<YourOwnFloatType>*)
    &((*(vector<YourOwnFloatType>*)(&temp))*ld);
  temp.optimize();
return temp;
}


//множення полінома на число
template <class YourOwnFloatType>
polynom <YourOwnFloatType> operator*(polynom<YourOwnFloatType> &v,
YourOwnFloatType ld)
{
  return ld*v;
}


//скорочене множення на число
template <class YourOwnFloatType>
polynom <YourOwnFloatType> operator*=
(polynom <YourOwnFloatType> &v,YourOwnFloatType ld)
{
  return v=v*ld;
}


//присвоювання
template <class YourOwnFloatType> polynom <YourOwnFloatType>
polynom<YourOwnFloatType>:: operator=(polynom<YourOwnFloatType> &x)
```

```
{
  //привласнюємо як вектора
  (*(vector<YourOwnFloatType>*)this)=(*(vector
<YourOwnFloatType>*)&x);
  optimize();//приводимо до канонічного виду
  return *this;//і повертаємо результат
}


//встановлення актуального степеня полінома
template <class YourOwnFloatType>
void polynom<YourOwnFloatType>::optimize()
{
  if(getm()!=1)//якщо поліном не нульового степеня
  {
    if((*this)[getm()-1]==(YourOwnFloatType)0)/*якщо коефіцієнт при
найвищому степені нульовий */
    {
      polynom<YourOwnFloatType> temp(getm()-1);
      //створюємо тимчасовий поліном степеня на 1 менше
      for(long i=0;i<getm()-1;i++)
        temp[i]=(*this)[i];
      *this=temp;//переписуємо його в поточний
      optimize();//знову перевіряємо поліном
    }
  }
}


/*у дуже рідких випадках може бути необхідним реверсувати поліном */
template <class YourOwnFloatType>
polynom<YourOwnFloatType> polynom<YourOwnFloatType>::reverse()
{
  polynom temp=*this;
  for(long i=0;i<getm();i++)
    temp[i]=(*this)[getm()-i-1];
  return temp;
}


//невід'ємний степінь полінома як функція
template <class YourOwnFloatType>
polynom<YourOwnFloatType> pow(polynom<YourOwnFloatType> x, unsigned
int p)
{
  polynom<YourOwnFloatType> temp;
  if(p==0)
    temp[0]=1;/* поліном у нульовому степені - це число 1*/
  else
  {
    temp=x;
    for(long i=0;i<p-1;i++)
      temp*=x;
  }
  return temp;//повертаємо результуючий поліном
}
```

```
//похідна j-го порядку
template <class YourOwnFloatType>
polynom<YourOwnFloatType> derive(polynom<YourOwnFloatType> p, long
j)
{
  if(j==0)/*нульова похідна полінома є сам поліном */
    return p;
  if(j<0)/*негативна похідна інтерпретується як інтеграл */
    return integral(p,-j);
  if(p.getm()==1)
    return polynom<YourOwnFloatType>();
    //якщо більш знижувати нікуди
  polynom<YourOwnFloatType> result=p.getm()-1;
  for(long i=0;i<result.getm();i++)
  //обчислюємо першу похідну
    result[i]=(i+1)*p[i+1];
  if(j==1)//якщо її і треба було знайти -
    return result;//повертаємо результат
  else//інакше
    return derive(result,j-1);  //обчислюємо похідну від даної
}


//інтеграл від полінома
template <class YourOwnFloatType>
polynom<YourOwnFloatType> integral(polynom<YourOwnFloatType> p, long
j)
{
  if(j<=0)
  /*негативна кратність інтегрування трактується як похідна */
    return derive(p,-j);
  polynom<YourOwnFloatType> result=p.getm()+1;
  for(long i=0;i<p.getm();i++)
    result[i+1]=p[i]/(i+1);
  if(j==1)
    return result;
  else
    return integral(result,j-1);
}


//степінь як операція
template <class YourOwnFloatType>
polynom<YourOwnFloatType> polynom<YourOwnFloatType>::
operator^(unsigned int p)
{
  return pow(*this,p);
}


//скорочений ступінь
template <class YourOwnFloatType>
polynom<YourOwnFloatType> polynom<YourOwnFloatType>::
operator^=(unsigned int p)
{
  return (*this)=pow(*this,p);
```

```
}


/*Перевантажувати оператори введення з потоку і виведення в потік
необхідності немає - досить перевантажити дві віртуальні функції
введення полінома з потоку, починаючи з коефіцієнтів при старших
степенях */
template <class YourOwnFloatType>
void polynom<YourOwnFloatType>::In(istream &is)
{
  for(long i=getm()-1;i>=0;i--)
    is>>(*this)[i];
  optimize();
}


/*виведення полінома в потік, починаючи з коефіцієнтів при старших
степенях */
template <class YourOwnFloatType>
void polynom<YourOwnFloatType>::Out(ostream &os)
{
  for(long i=getm()-1;i>=0;i--)
  {
    os.precision(100);
    //установлюємо величезну точність висновку
    os<<(*this)[i]<<" ";/*компоненти розділяємо пробілами */
  }
}


//порівняння поточного поліном з поліномом за адресою x
template <class YourOwnFloatType>
long polynom<YourOwnFloatType>::IsEqual(void *x)
{
  polynom<YourOwnFloatType> test1=*(polynom<YourOwnFloatType>*)x,
                            test2=(*this);
  test1.optimize();
  test2.optimize();
  return ((vector<YourOwnFloatType>*)&test1)->
     vector<YourOwnFloatType>::IsEqual(&test2);
}


/*
  З двох поліномів однакової довжини менше той, у якого коефіцієнт
при старшому степені менше. При рівності розглядаємо більш низький
степінь і т.д.
*/
template <class YourOwnFloatType>
long operator<(polynom<YourOwnFloatType> &f, polynom
<YourOwnFloatType> &s)
{
  polynom<YourOwnFloatType> test1=f,test2=s;
  test1.optimize();
  test2.optimize();
  if(test1.getm()<test2.getm())
    return 1;
```

```
  if(test1.getm()>test2.getm()||test1==test2)
    return 0;
  //точно не рівні - з'ясовуємо, хто ж з них менше
  for(long i=test1.getm()-1;i>=0;i--)
    if(test1[i]<test2[i])
      return 1;
    else
      if(test1[i]>test2[i])
        return 0;
  return 0;
}


/*всі інші операції визначаємо через уже відомі */
template <class YourOwnFloatType>
long operator>(polynom<YourOwnFloatType> &f, polynom
<YourOwnFloatType> &s)
{
  return (!(f<s))&&(f!=s);
}


template <class YourOwnFloatType>
long operator<=(polynom<YourOwnFloatType> &f, polynom
<YourOwnFloatType> &s)
{
  return (f<s)||(f==s);
}


template <class YourOwnFloatType>
long operator>=(polynom<YourOwnFloatType> &f, polynom
<YourOwnFloatType> &s)
{
  return (f>s)||(f==s);
}


/* ділення полінома на поліном за алгоритмом Евкліда */
template <class YourOwnFloatType> long div(polynom
<YourOwnFloatType> &f, polynom<YourOwnFloatType> &g,
polynom<YourOwnFloatType> &l, polynom<YourOwnFloatType> &r)
{
  polynom<YourOwnFloatType> tf=f,tg=g;
  tf.optimize();
  tg.optimize();
  if(tg==polynom<YourOwnFloatType>())
  //спроба ділення на нуль
    throw xmsg("Спроба ділення на 0\n");
  if(tf.getm()<tg.getm())
  {
    l=polynom<YourOwnFloatType>();
    r=tf;
    return 1;
  }
  for(l=polynom<YourOwnFloatType>();tf.getm()>= tg.getm();)
  {
```

```
    polynom<YourOwnFloatType> x(2);
    x[1]=1;
    x^=tf.getm()-tg.getm();
    x*=tf[tf.getm()-1]/tg[tg.getm()-1];
    r=tf-=tg*x;
    l+=x;
  }
  return 1;
}


//ціла частина від ділення
template <class YourOwnFloatType> polynom <YourOwnFloatType>
operator/(polynom<YourOwnFloatType> &f, polynom<YourOwnFloatType>
&s)
{
  polynom<YourOwnFloatType> l,r;
  div(f,s,l,r);
  return l;
}


//остача від ділення
template <class YourOwnFloatType> polynom <YourOwnFloatType>
operator%(polynom<YourOwnFloatType> &f, polynom<YourOwnFloatType>
&s)
{
  polynom<YourOwnFloatType> l,r;
  div(f,s,l,r);
  return r;
}


#endif


#ifndef __EQUATION_H
#define __EQUATION_H

/*Цей файл включення містить оголошення двох типів - комплексного
полінома і комплексного вектора, а також заголовки чотирьох функцій
для рішення рівнянь 2-го, 3-го, 4-го і вищих степенів */


#ifndef __COMPLEX_H
#include <complex.h>
#endif
#ifndef __POLYNOM_H
#include "polynom.h"
#endif
#ifndef __VECTOR_H
#include "vector.h"
#endif
#ifndef __MATRIX_H
#include "matrix.h"
#endif
```

```
typedef polynom<complex> cpolynom;
typedef vector<complex> cvector;
typedef matrix<complex> cmatrix;

cvector square(complex, complex, complex);
cvector kardano(double, double, double, double);
cvector ferrary(double, double, double, double, double);
cvector newton(cpolynom);

#endif


#ifndef __EQUATION_H
#include "equation.h"
#endif
/*Рішення квадратного рівняння з комплексними коефіцієнтами */
cvector square(complex a2,complex a1,complex a0)
{
  complex a=a2,b=a1,c=a0;
 /*Знайдені комплексні корені повинні бути записані в двохкомпонент-
ний комплексний вектор-результат */
  cvector res(2);
  //перевіримо, чи не нульовий коефіцієнт при x^2
  if(a!=complex(0,0))
  {
    /*якщо так, шукаємо корені через квадратичний дискримінант */
    complex D=b*b-4*a*c;
    res[0]=(-b+sqrt(D))/(2*a);     //і заносимо їх у відповідні
    res[1]=(-b-sqrt(D))/(2*a);     //компоненти вектора-результату
  }
  else
    res[0]=res[1]=-c/b; /*інакше розв'язуємо рівняння першого степе-
ня */
  return res;
}


/*Рішення кубічного рівняння з дійсними коефіцієнтами методом Карда-
но-Тартальї*/
cvector kardano(double a3, double a2, double a1, double a0)
{
  /*Загальний вид рівняння, корені якого ми шукаємо -
a3*x^3+a2*x^2+a1*x+a0=0. Так як рівняння має третій степінь, то і
коренів у нього три, що і визначає розмірність вектора-результату */
  cvector res=3;
  if(a3==0)//якщо коефіцієнт при x^3 нульовий,
  {
    //розв'язуємо відповідне квадратне рівняння
    cvector res2=square(a2,a1,a0);
    /*у цьому випадку приймаємо, що два корені є співпадаючими */
    res[0]=res[1]=res2[0];
    res[2]=res2[1];
  }
  else
  {
    /*інакше - приводимо кубічне рівняння до канонічного виду, коли
коефіцієнт при третьому ступені дорівнює 1 */
```

```
    double a=a2/a3,b=a1/a3,c=a0/a3;
    //знаходимо доданки кубічного дискримінанта
    double p=b-a*a/3,q=2*a*a*a/27-a*b/3+c;
    /*виконавши перепозначення x=y-a/3, розв'язуємо надалі рівняння
y^3+p*y+q=0 */
    double diskr=pow(q/2,2)+pow(p/3,3); //знаходимо кубічний дискри-
мінант
    /*якщо дискримінант дорівнює 0, то маємо 3 дійсних корені, з них
два співпадаючих */
    if(diskr==0)
    {
      res[0]=3*q/p;
      res[1]=res[2]=-res[0]/2;
    }
    /*один дійсний корінь і два комплексно спряжених */
    if(diskr>0)
    {
      double what=-q/2+sqrt(diskr);
      double u0=(what>0)?pow(what,1.0/3.0):-pow(-what, 1.0/3.0);
      double v0=-p/(3*u0);
      res[0]=u0+v0;
      res[1]=res[2]=-res[0]/2;
      complex k3(0,sqrt(3)*(u0-v0)/2);
      res[1]+=k3;
      res[2]-=k3;
    }
    //три різних дійсних корені
    if(diskr<0)
    {
      cvector zkub12=square(1,q,-p*p*p/27);
      complex u0=pow(zkub12[0],1/3.);
      res[0]=2*real(u0);
      res[1]=-real(u0)-imag(u0)*sqrt(3);
      res[2]=-real(u0)+imag(u0)*sqrt(3);
    }
    //переходимо назад від y к x
    for(int i=0;i<3;i++)
      res[i]-=a/3;
  }
  return res;//повертаємо вектор-результат
}


/*Метод Феррарі для рішення рівнянь четвертого степеня з дійсними
коефіцієнтами */
cvector ferrary(double a4, double a3, double a2, double a1, double
a0)
{
  cvector res=4;//усього коренів буде 4
  if(!a4)/*якщо коефіцієнт при четвертому ступені x нульовий, нама-
гаємося розв'язати рівняння третього степеня */
  {
    cvector res3=kardano(a3,a2,a1,a0);
    res[0]=res[1]=res3[0];
    for(int i=1;i<3;i++)
      res[i+1]=res3[i];
  }
```

```
  else
  {
    double a=a3/a4,b=a2/a4,c=a1/a4,d=a0/a4;
    /*складаємо і знаходимо корені кубічної резольвенти */
    cvector cy0=kardano(1,-b,-4*d+a*c,-d*a*a+4*b*d-c*c);
    double y0;
    /*шукаємо хоча б один дійсний корінь, за допомогою якого зводимо
рівняння четвертого ступеня до сукупності квадратних */
    for(int i=0;i<3;i++)
      if(!imag(cy0[i]))
      {
        y0=real(cy0[i]);
        break;
      }
    double A=a*a/4-b+y0,B=a*y0/2-c;
    double x12=-B/(2*A);
    //вирішуємо квадратні рівняння
    cvector sq1=square(1,a/2-sqrt(A),y0/2+sqrt(A) *x12);
    cvector sq2=square(1,a/2+sqrt(A),y0/2-sqrt(A) *x12);
    for(long i=0;i<2;i++)
      res[i]=sq1[i],res[i+2]=sq2[i];
  }
  return res;//повертаємо результат
}


/*модифікований метод Ньютона пошуку комплексних коренів полінома */
cvector newton(cpolynom p)
{
  double t=1,eps=1e-8,j;

  p*=1;/* виконуємо множення на 1 для того, щоб відкинути ведучі ну-
лі */
  //результуючий вектор буде мати розмірність,
  //рівну порядку полінома
  cvector result=p.getm()-1;
  /*якщо вектор-результат одномірний, те маємо справа з поліномом
першого степеня */
  if(result==cvector())
  {
    result[0]=-p[0]/p[1];
    return result;
  }
  //якщо двовимірний - із квадратним рівнянням і т.д.
  if(result.getm()==2)
    return square(p[2],p[1],p[0]);
  if(result.getm()==3&&!imag(p[3])&&!imag(p[2])&&
!imag(p[1])&&!imag(p[0]))
    return kardano(real(p[3]), real(p[2]), real(p[1]), real(p[0]));
  if(result.getm()==4&&!imag(p[4])&&!imag(p[3])&&
!imag(p[2])&&!imag(p[1])&&!imag(p[0]))
    return ferrary(real(p[4]), real(p[3]), real(p[2]), real(p[1]),
real(p[0]));
  complex z=0;/*початкове наближення до кореня покладемо рівним
(0,0)*/
  do
  {
```

```
    complex dz=0;
    /*знаходимо порядок і значення ненульової похідної в точці z */
    for(j=1;dz==complex(0);dz=derive(p,j++)(z));
    z+=t*pow(-p(z)/dz,1/(-j));/*модифікуємо наближення */
    t*=(1-eps);
  }while(abs(p(z))>=eps);
  //повторюємо до досягнення заданої точності
  /*У цьому циклі знаходиться тільки один корінь z. Для виділення
інших ділимо вихідний поліном на x-z, знижуючи тим самим степінь на
одиницю, і знаходимо ще один корінь, і т.д. до першого степеня */
  result[0]=z;
  cpolynom z1=2;
  z1[0]=-z,z1[1]=1;
  cvector more=newton(p/z1);
  for(long i=1;i<result.getm();i++)
    result[i]=more[i-1];
  return result;//повертаємо вектор результату
}
```

Додаток В. Структура матричного класу

```
template <class YourOwnFloatType> class matrix
{
```

Як і векторний, клас для роботи з матричними об'єктами – параметризований; це лише шаблон, за яким для конкретних типів, що підставляються замість YourOwnFloatType, компілятор автоматично генерує клас. Для матричного типу типами, що підставляються, можуть бути типи, для яких визначені стандартні арифметичні операції.

```
long m, n;
```

У цих двох приватних змінних будемо зберігати розмірність матриці – кількість рядків (у першій змінній) і стовпців (у другій). Цілком зрозумілою вимогою до цих чисел є те, що вони повинні бути натуральними.

```
vector<YourOwnFloatType> *mtr;
```

Розглянемо матрицю $\mathbf{A}=\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}$.

Кожен рядок (так само, як і стовпець) цієї матриці являє собою упорядковану послідовність числових об'єктів, тобто вектор. Традиційно вважають, що векторами є рядки матриці:

$$\mathbf{A}=\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}, \begin{array}{l} \mathrm{A}_1 = (a_{11}, a_{12}, \dots, a_{1n}), \\ \mathrm{A}_2 = (a_{21}, a_{22}, \dots, a_{2n}), \\ \dots\dots\dots\dots\dots\dots\dots\dots, \\ \mathrm{A}_m = (a_{m1}, a_{m2}, \dots, a_{mn}). \end{array} , \rightarrow \mathbf{A}=\begin{pmatrix} \mathrm{A}_1 \\ \mathrm{A}_2 \\ \dots \\ \mathrm{A}_m \end{pmatrix}.$$

Можливість такого подання матриці і зручність зберігання векторів-рядків дає нам підстави використовувати як внутрішній тип для збереження матриці покажчик на вектор.

```
public:
```

Загальнодоступні методи (функції-члени) і дружні функції, що використовуються для роботи з матричними об'єктами. Більшість з них є реалізаціями відповідних операцій матричної алгебри, тому вони подані у вигляді перевизначених операцій.

```
matrix(char *);
```

Як параметр цей конструктор приймає ім'я текстового файлу, у якому зберігається матриця. При цьому передбачається, що перші два числа у файлі – це розмірність матриці, тобто кількість її рядків і стовпців. Після них йдуть числа, які є елементами матриці; кількість яких повинна бути *m*x*n*. При зчитуванні числа розміщаються порядково: спочатку заповнюється перший рядок, потім – другий і т.д. Надлишок даних у файлі ігнорується, при нестачі даних матриця заповнюється нулями, починаючи з позиції, що слідує за останнім числом.

```
matrix();
```

Для створення масивів матриць нам необхідний конструктор за замовчуванням. Він створює порожню матрицю розміром 1x1, викликаючи конструктор векторного класу за замовчуванням. Ця матриця фактично моделює один елемент типу, що підставляється.

```
matrix(long, long);
```

Цей конструктор приймає два параметри – натуральні числа, перше з який задає кількість рядків у матриці, а друге – кількість стовпців. Так як жодних відомостей про елементи матриці не дається, вважається, що даний конструктор призначений для створення нульових матриць заданого розміру. Отже, при використанні цього конструктора не виникає необхідності очищати матрицю перед початком роботи з нею.

```
matrix(long, long, YourOwnFloatType *);
```

Цей конструктор приймає три параметри – натуральні числа, перше з який задає кількість рядків у матриці, а друге – кількість стовпців, і покажчик на елементи того типу, що використовується для зберігання даних у векторах.

Після створення матриці заданих розмірів, відбувається спроба заповнення її *size1*x*size2* елементами з масиву, на який вказує третій параметр. Якщо цей покажчик не ініційований чи вказує на область пам'яті, яка не містить об'єктів типу, що підставляється, або кількість таких об'єктів менше, ніж добуток кількості рядків на кількість стовпців, то частина чи вся матриця, у залежності від ситуації, буде містити «сміття», над яким і будуть виконуватися подальші операції в припущенні про коректність даних, що зберігаються.

```
matrix(matrix &);
```

Конструктор копіювання матричного класу використовується тоді, коли необхідно «зліпити» матрицю «за образом та подобою» вже існуючої. Це означає, що конструйована матриця буде мати таку саму розмірність, що й копійована матриця-параметр, а всі елементи їх будуть збігатися. Зрозуміло, дані цих матриць зберігаються в різних областях пам'яті, і дії над однією матрицею не відбиваються на іншій, як це відбувалося б у синтаксично схожій ситуації оголошення посилального об'єкта:

matrix a=e; /*матриці a й e однакові, але в різних областях пам'яті */

matrix &c=a; /*матриці c і a однакові й в одній області пам'яті */

```
~matrix();
```

Деструктор. Якщо матриця була розміщена в оперативній пам'яті як динамічний об'єкт, то він буде викликатися при виконанні оператора delete, у всіх інших випадках – при виході матричного об'єкта з області видимості. Зважаючи на те, що єдине поле матричного класу, яке розподіляється динамічно – це покажчик на вектор, то саме з-під нього пам'ять і буде звільнена. При цьому викличуться деструктори кожного вектора-рядка, що зберігається в матриці, знищуючи самі дані з рядків.

```
friend matrix<YourOwnFloatType> operator+(matrix
<YourOwnFloatType> &, matrix<YourOwnFloatType> &);
```

Дружня функція додавання матриць приймає два параметри – матриці од-

накового типу (однакової розмірності), а матрицю-суму повертає як результат. Сумою двох матриць **A** і **B** однакового типу називається матриця **C** того ж типу, елементи якої c_{ij} дорівнюють сумам відповідних елементів a_{ij} і b_{ij} матриць **A** и **B**, тобто $c_i=a_{ij}+b_{ij}$. З іншого боку, оскільки в якості одного з подань матриці ми маємо векторне, то суму матриць легко представити через суму векторів, що складають рядки матриць доданків.

```
friend matrix<YourOwnFloatType> operator-(matrix
<YourOwnFloatType> &, matrix<YourOwnFloatType> &);
```

Дружня функція віднімання матриць приймає два параметри – матриці однакового типу (розмірності), а матриця-різниця повертається як результат.

```
friend matrix<YourOwnFloatType> operator*(matrix
<YourOwnFloatType> &, matrix<YourOwnFloatType> &);
```

Дружня функція некомутативного множення матриць, як і будь-яка матрична бінарна операція, приймає два параметри – матриці, що перемножуються, і повертає матрицю-результат, якщо типи матриць такі, що останній визначений.

Нехай **A** і **B** – матриці типів відповідно *m*x*n* і *p*x*q*. Якщо число стовпців матриці **A** дорівнює числу рядків матриці **B**, тобто $n=p$, то для цих матриць визначена матриця **C** типу *m*x*q*, що називається їхнім добутком: **C**, де $c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + ... + a_{in}b_{nj} = \sum_{k=1}^{n} a_{ik}b_{kj}$ (i=1, 2, …, m; j=1, 2, …, q).

З визначення випливає наступне правило множення матриць: щоб одержати елемент, що стоїть в *i*-ому рядку і *j*-ому стовпці добутку двох матриць, потрібно елементи *i*-го рядка першої матриці помножити на відповідні елементи *j*-го стовпця другої і отримані добутку скласти.

Добуток **AB** має сенс тоді і лише тоді, коли матриця **A** містить у рядках стільки елементів, скільки елементів міститься в стовпцях матриці **B**. Зокрема, можна перемножувати квадратні матриці лише однакового порядку. У тих окремих випадках, коли **AB**=**BA**, матриці **A** і **B** називаються *комутативними*. Так, наприклад, одинична матриця **E** відіграє роль оди-

ниці при множенні – вона комутативна з будь-якою квадратною матрицею **A** того ж порядку, причому **AE=EA=A**.

```
friend matrix<YourOwnFloatType> operator* (YourOwnFloatType,
matrix<YourOwnFloatType> &);
friend matrix<YourOwnFloatType> operator*(matrix
<YourOwnFloatType> &, YourOwnFloatType );
```

На відміну від множення матриці на матрицю, множення матриці на скаляр – операція, результат якої однозначний і визначений завжди: це матриця тієї ж розмірності, що і вихідна. Добутком матриці **A** на число α (чи добутком числа α на матрицю **A**) називається матриця, елементи яка отримані множенням всіх елементів матриці **A** на число α, тобто

$$\alpha\mathbf{A} = \mathbf{A}\alpha = \begin{pmatrix} \alpha a_{11} & \alpha a_{12} & \dots & \alpha a_{1n} \\ \alpha a_{21} & \alpha a_{22} & \dots & \alpha a_{2n} \\ \dots & \dots & \dots & \dots \\ \alpha a_{m1} & \alpha a_{m2} & \dots & \alpha a_{mn} \end{pmatrix} \text{ чи } \alpha\mathbf{A} = \mathbf{A}\alpha = \begin{pmatrix} \alpha \mathrm{A}_1 \\ \alpha \mathrm{A}_2 \\ \dots \\ \alpha \mathrm{A}_\mathrm{m} \end{pmatrix}.$$

```
friend ostream &operator<<(ostream &, matrix<YourOwnFloatType>
&);
friend istream &operator>>(istream &, matrix<YourOwnFloatType>
&);
```

Виведення матриці в потік і введення матриці з потоку – дві операції, що класично перевантажуються для кожного типу. Першим параметром кожної з функцій є потік виведення (уведення), другим – матричний об'єкт, сконструйований раніше. Особливий інтерес являє операція введення з потоку: при зчитуванні даних уміст поточної матриці заміщується об'єктами, що зчитуються з потоку – саме з цією метою передача матриці у функцію організована за посиланням. При записі (вставці) матриці в потік використовується відповідна перевантажена операція векторного класу для кожного з векторів-рядків матриці.

```
friend matrix<YourOwnFloatType> SLAE_Orto(matrix
<YourOwnFloatType> &, matrix<YourOwnFloatType> &);
```

Перший параметр цієї функції – квадратна матриця *N*x*N*, другий – *N*x1, значення, що повертається – матриця *N*x1. Розглянемо детальніше, що це за функція, для чого вона призначена і чому її аргументи саме такі.

Нехай нам необхідно розв'язати систему лінійних рівнянь виду:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n = b_2 \\ ..... \\ a_{n1}x_1 + a_{n2}x_2 + ... + a_{nn}x_n = b_n \end{cases}.$$

З елементів цієї системи складемо три матриці:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & ... & a_{1n} \\ a_{21} & a_{22} & ... & a_{2n} \\ ... & ... & ... & ... \\ a_{n1} & a_{n2} & ... & a_{nn} \end{pmatrix}, \ \mathbf{X} = \begin{pmatrix} x_1 \\ x_2 \\ ... \\ x_n \end{pmatrix} \text{ і } \mathbf{B} = \begin{pmatrix} b_1 \\ b_2 \\ ... \\ b_n \end{pmatrix}.$$

Помноживши матрицю **A** на матрицю **X**, ми одержимо матрицю-стовпець, елементи якої являють собою ліву частину записаної системи. Так як ліва частина системи дорівнює правій, то отримана матриця-стовпець не може бути нічим іншим, як матрицею-стовпцем **B**. Таким чином, ми маємо право записати систему лінійних алгебраїчних рівнянь у матричній формі як

$$\mathbf{AX=B}.$$

У записаній системі матриця-стовпець **X** являє собою вектор невідомих, котрі необхідно буде визначити, **A** – матриця коефіцієнтів при невідомих і **B** – права частина СЛАР (матриця-стовпець вільних членів, узятих зі зворотним знаком). Отже, задачу розв'язання системи рівнянь можна звести до задачі визначення такої матриці **X**, що при множенні праворуч на матрицю **A** дає матрицю **B**. Цим і пояснюються вимоги до параметрів розглянутої функції: перша матриця містить коефіцієнти системи рівнянь, друга – стовпець вільних членів, узятий зі зворотним знаком, а значенням даної функції, що повертається, буде стовпець невідомих.

Будемо виходити з припущення, що така матриця існує. Тоді помно-

жимо це рівняння по обидва боки ліворуч на деяку матрицю $\mathbf{A}^{-1}$ таку, котра при множенні на матрицю **A** ліворуч дає одиничну матрицю:

$$\mathbf{A}^{-1}\mathbf{AX}=\mathbf{A}^{-1}\mathbf{B},\ \mathbf{EX}=\mathbf{A}^{-1}\mathbf{B},\ \mathbf{X}=\mathbf{A}^{-1}\mathbf{B}.$$

Знаючи спосіб одержання матриці $\mathbf{A}^{-1}$ з матриці **A**, ми можемо вирішити поставлену задачу. Цей шлях ми, однак, розглядати зараз не будемо, а підійдемо до проблеми з іншого боку, для чого розглянемо *метод ортогоналізації*. Нехай вихідна система переписана у виді:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n - b_1 = 0 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n - b_2 = 0 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n - b_n = 0 \end{cases}.$$

Увівши позначення $a_{i(n+1)}=-b_i$, одержимо

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n + a_{1(n+1)} = 0 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n + a_{2(n+1)} = 0 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n + a_{n(n+1)} = 0 \end{cases}.$$

Ліву частину кожного рівняння системи можна розглядати як скалярний добуток двох векторів: $\boldsymbol{A}_i=(a_{i1}, a_{i2}, \ldots, a_{in}, a_{i(n+1)})$ і $\boldsymbol{X}=(x_1, x_2, \ldots, x_n, 1)$. За такої постановки рішення системи зводиться до побудови вектора, ортогонального до кожного вектора $\boldsymbol{A}_i$. Додамо до системи векторів $\boldsymbol{A}_i$ лінійно незалежний від них вектор $\boldsymbol{A}_{n+1}=(0, 0, \ldots, 0, 1)$. У векторному просторі розмірності n+1 будемо будувати такий його ортонормований базис $\boldsymbol{B}_1, \boldsymbol{B}_2, \ldots, \boldsymbol{B}_{n+1}$, щоб при будь-якому $k=1, 2, \ldots, n+1$ вектори $\boldsymbol{B}_1, \boldsymbol{B}_2, \ldots, \boldsymbol{B}_k$ утворювали ортонормований базис підпростору P_k, породженого розглянутими векторами $\boldsymbol{A}_1, \boldsymbol{A}_2, \ldots, \boldsymbol{A}_k$. Для цього досить будувати в підпросторі P_k деякий ортогональний базис $\boldsymbol{U}_1, \boldsymbol{U}_2, \ldots, \boldsymbol{U}_k$, а потім нормувати його. Звичайно цей процес рекомендується повторити 3-4 рази з метою як можна більш точної ортогоналізації і, відповідно, більш точного рішення розглянутої системи. Більш просто розглянутий алгоритм можна сформулювати так:

1. Нормуємо перший вектор-рядок системи.
2. Ортогоналізуємо другий вектор до першого, а потім нормуємо його.
3. Ортогоналізуємо третій вектор до першого і до другого, а потім нормуємо, і т.д., із всіма іншими векторами.

При цьому самий останній вектор, яким ми доповнили систему, буде ортогональним до усіх попередніх, а, отже, і буде її рішенням.

```
friend matrix<YourOwnFloatType> SLAE_Gauss(matrix
<YourOwnFloatType> &, matrix<YourOwnFloatType> &);
```

Розглянемо ще один метод рішення систем лінійних рівнянь – метод Гаусса, чи метод послідовного виключення невідомих. Суть його складається в перетворенні системи

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n \end{cases}$$

до системи з трикутною матрицею, з якої потім послідовно (зворотним ходом) виходять значення всіх невідомих.

Піддамо цю систему наступному перетворенню. Вважаючи, що $a_{11}\neq 0$ (ведучий елемент), розділимо на a_{11} коефіцієнти першого рівняння (виконання умови $a_{11}\neq 0$ можна домогтися завжди шляхом перестановки рівнянь системи):

$$x_1+\alpha_{12}x_2+\ldots+\alpha_{1n}x_n=\beta_1.$$

Користуючись цим рівнянням, легко виключити невідоме x_1 з інших рівнянь системи (для цього досить з кожного рівняння відняти його, попередньо помножене на відповідний коефіцієнт при x_1). Потім над іншими рівняннями системи зробимо аналогічне перетворення: виберемо з їхнього числа рівняння з ведучим елементом і виключимо з його допомогою з інших рівнянь невідоме x_2. Повторюючи цей процес, замість вихідної системи одержимо рівносильну їй систему з трикутною матрицею коефіцієнтів при невідомих:

$$\begin{cases} x_1 + \alpha_{12}x_2 + \alpha_{13}x_3 + ... + \alpha_{1n}x_n = \beta_1 \\ x_2 + \alpha_{23}x_3 + ... + \alpha_{2n}x_n = \beta_2 \\ ..... \\ x_n = \beta_n \end{cases},$$

з якої послідовно знаходяться значення всіх невідомих x_n, x_{n-1}, …, x_1.

Таким чином, процес розв'язання за методом Гаусса розпадається на два етапи. Перший етап, що складається в послідовному виключенні невідомих, називають прямим ходом. Другий – визначення значень невідомих – прийнято називати зворотним ходом.

```
matrix<YourOwnFloatType> minor(long, long);
```

Ця функція в програмах в явному вигляді звичайно не застосовується. Вона має два параметри – номер рядка і стовпця даної матриці, що не записуються в матрицю-результат (створення матриці з наявної без заданих рядка і стовпця). При цьому кожна з розмірностей матриці (кількість рядків і кількість стовпців) зменшується на одиницю.

```
friend YourOwnFloatType det2(matrix<YourOwnFloatType> &);
```

Обчислення значення визначника квадратної матриці є важливою задачею лінійної алгебри. Так, чисельне рішення системи рівнянь має сенс лише в тому випадку, коли матриця, складена з коефіцієнтів при невідомих цієї системи, невироджена, тобто коли її визначник відмінний від нуля. Однак і в тому випадку, коли визначник системи відмінний від нуля, але дуже малий за абсолютною величиною, до отриманих в ході рішення значень коренів слід відноситися з обережністю, оскільки вони можуть значно відрізнятися від справжніх значень невідомих. Тому розв'язання системи лінійних рівнянь корисно супроводжувати обчисленням визначника цієї системи. Обчислення визначника може становити і самостійний інтерес.

З курсу лінійної алгебри відоме правило, за яким визначник порядку n можна виразити через n визначників порядку на одиницю нижче:

$$\Delta = a_{i1}A_{i1} + a_{i2}A_{i2} + ... + a_{in}A_{in}.$$

Це операція розкладання визначника за елементами i-го рядка; $A_{ij}=(-1)^{i+j}M_{ij}$ – алгебраїчне доповнення елемента a_{ij}, M_{ij} – мінор елемента a_{ij}, тобто визначник $(n-1)$-го порядку, одержуваний з визначника Δ викреслюванням i-го рядка та j-го стовпця.

Описана процедура дозволяє реалізувати дуже просту рекурсивну функцію для обчислення детермінанта розкладанням по одному з рядків; при цьому, однак, число множень і ділень, необхідних для обчислення визначника n-го порядку, дорівнює $(n-1)(n^2+n+3)/3$, що стримує застосування такого методу для обчислення детермінантів високих порядків.

```
friend YourOwnFloatType det(matrix<YourOwnFloatType> &);
```

Для системи рівнянь з матриці коефіцієнтів ми можемо скласти детермінант і обчислити його значення. При цьому будь-які зміни матриці коефіцієнтів ведуть до зміни її детермінанта. Розглянемо, як у методі Гаусса змінюється визначник вихідної системи. Позначимо визначник системи рівнянь через D. Після того, як ми розділимо ліву і праву частини першого рівняння на ведучий елемент a_{11}, визначник перетвореної системи дорівнюватиме D/a_{11}, наступні перетворення, пов'язані з виключенням x_1 з інших рівнянь системи, величину визначника не змінюють. На другому кроці, коли ми розділимо обидві частини перетвореного другого рівняння на другий ведучий елемент a_{22}, визначник отриманої системи дорівнюватиме $D/(a_{11}\cdot a_{22})$. Операції по виключенню x_2 з рівнянь системи знову не змінюють величини визначника. Здійснюючи аналогічні дії, ми на n-му кроці прийдемо до системи, визначник якої буде $D/(a_{11}\cdot a_{22}\cdot\ldots\cdot a_{nn})$. Але матриця коефіцієнтів при невідомій системі – трикутна, з одиницями на головній діагоналі, тому її визначник дорівнює 1. Одержуємо, що $D/(a_{11}\cdot a_{22}\cdot\ldots\cdot a_{nn})=1$, тобто $D=a_{11}\cdot a_{22}\cdot\ldots\cdot a_{nn}$. Таким чином, для обчислення визначника системи рівнянь потрібно одержати добуток ведучих елементів, використовуваних на кожному кроці методу Гаусса.

```
matrix<YourOwnFloatType> operator=(matrix<YourOwnFloatType>
```

```
&);
```

Присвоювання матриць – бінарна операція, яку реалізують як метод класу. Якщо розмірність поточної матриці не збігається з розмірністю присвоюваної, ми спочатку перерозподіляємо пам’ять під її вектори таким чином, щоб кількість векторів і їхні розмірності в обох матрицях збігалися. Потім відбувається повекторне копіювання рядків копійованої матриці в рядки поточної. Значення, що повертається – копія поточної матриці.

```
matrix<YourOwnFloatType> operator*=(matrix<YourOwnFloatType>
&x);
```

При роботі з убудованими типами даних зручною можливістю є використання набору скорочених операцій, у яких дія комбінується зі знаком присвоювання. Наприклад, ця функція перевантажує операцію скороченого множення. Реалізована як метод класу, вона множить поточну матрицю на передану як параметр, а результат множення не лише повертається, але й перезаписується в поточну матрицю. Дія цієї операції, зрозуміло, має ті ж обмеження, що накладаються на множення матриць.

```
matrix<YourOwnFloatType> operator+= (matrix<YourOwnFloatType>
&x);
matrix<YourOwnFloatType> operator-= (matrix<YourOwnFloatType>
&x);
```

Як і скорочене множення, скорочені операції додавання і віднімання реалізовані як методи матричного класу. Обидві ці функції працюють, використовуючи раніше визначені операції додавання, віднімання та присвоювання.

```
matrix<YourOwnFloatType> operator^(long);
```

Для квадратних матриць ми можемо визначити таку операцію, як піднесення матриці до цілого невід’ємного степеня.

```
matrix<YourOwnFloatType> operator^=(long);
friend matrix<YourOwnFloatType> pow(matrix<YourOwnFloatType>
&, long);
```

Для зручності роботи визначимо також функцію-член для скороченої операції піднесення до невід'ємного степеня, а також дружню функцію для піднесення матриці до степеня.

```
matrix<YourOwnFloatType> operator~();
```

Замінивши в матриці **A** типу *m*x*n* рядки відповідно стовпцями, одержимо так звану *транспоновану* матрицю $\mathbf{A}^T$ типу *n*x*m*, що має наступні властивості:

- двічі транспонована матриця збігається з вихідною;
- транспонована матриця суми дорівнює сумі транспонованих матриць доданків;
- транспонована матриця добутку дорівнює добутку транспонованих матриць співмножників, узятому в зворотному порядку.

Якщо матриця **A** квадратна, то $\det \mathbf{A}^T = \det \mathbf{A}$. Матриця **A** називається *симетричною*, якщо вона збігається зі своєю транспонованою, тобто якщо $\mathbf{A}^T = \mathbf{A}$. Звідси випливає, що симетрична матриця – квадратна й елементи її, симетричні щодо головної діагоналі, рівні між собою. Добуток $\mathbf{C}=\mathbf{A}\mathbf{A}^T$ являє собою симетричну матрицю, тому що $\mathbf{C}^T=(\mathbf{A}\mathbf{A}^T)^T=(\mathbf{A}^T)^T\mathbf{A}^T=\mathbf{A}\mathbf{A}^T=\mathbf{C}$.

Розглянемо систему лінійних алгебраїчних рівнянь, записаних у матричній формі, і припустимо, що кількість рівнянь у цій системі перевищує кількість невідомих. Зрозуміло, що при використанні методу послідовного виключення невідомих після приведення системи до трикутного виду одержимо «зайві» рівняння. Якщо ця система рівнянь визначає якусь залежність, то точне відновлення цієї залежності стає проблематичним. Однак, використовуючи операцію транспонування, ми можемо одержати наближений розв'язок цієї системи, причому такий, що обчислена за ним права частина кожного рівняння буде досить мало відрізнятися від вихідної. Запишемо цю процедуру у формі матричного рівняння:

$$\mathbf{A}\mathbf{X}=\mathbf{B}$$

$$\mathbf{A}^T\mathbf{A}\mathbf{X}=\mathbf{A}^T\mathbf{B}$$

$$\mathbf{A}^T\mathbf{A}=\mathbf{A}', \mathbf{A}^T\mathbf{B}=\mathbf{B}'$$

$$\mathbf{A}'\mathbf{X}=\mathbf{B}'$$

Якщо розв'язок вихідної системи за методом Гаусса знайти було неможливо, то розв'язати перетворену систему цілком можливо. Розмірність матриці **A** – *m*x*n*, розмірність матриці $\mathbf{A}^T$ – *n*x*m*, розмірність матриці **X** – *n*x1, розмірність матриці **B** – *m*x1. Розмірність добутку $\mathbf{A}^T\mathbf{A}$, відповідно до правила множення матриць, буде *n*x*n*, розмірність добутку $\mathbf{A}^T\mathbf{B}$ – *n*x1, що задовольняє параметрам функції для розв'язання системи рівнянь. Система, визначена таким способом, називається *нормальною*; матриця цієї системи називається *матрицею плану*, а розв'язок цієї системи буде оптимальним за критерієм мінімізації суми квадратів відхилень від справжніх значень (критерію МНК – *методу найменших квадратів*).

```
matrix<YourOwnFloatType> operator!();
```

Нехай у нас є набір значень деякої невідомої функції, що пов'язує набір змінних зі значенням-результатом. Це може бути, приміром, функція залежності швидкості росту стебла від вологості, температури, тиску і якихось ще параметрів. Висунемо гіпотезу про вид зв'язку цих параметрів з результатом, причому будемо вважати, що ця залежність задається у вигляді лінійної комбінації відомих функцій, що пов'язують ці параметри.

Підставляючи конкретні значення у функцію-гіпотезу, ми одержимо систему рівнянь, у правій частині якої буде лінійна комбінація невідомих коефіцієнтів функції-гіпотези з числовими значеннями, отриманими при підстановці, а в лівій – результати вимірів. Якщо кількість рівнянь у цій системі в нас буде менше, ніж число коефіцієнтів у функції-гіпотезі, то в нас просто недостатньо даних, щоб обчислити ці коефіцієнти. Якщо ці значення збігаються чи рівнянь більше, ніж невідомих, таку систему приводять до нормальної і визначають коефіцієнти функції-гіпотези – моделі даного процесу. Позначаючи через **X** матрицю системи, складеної за модельною функцією, через **Y** – матрицю-стовпець вимірюваних значень, а че-

рез **A** – матрицю-стовпець невідомих коефіцієнтів моделі, маємо $\mathbf{XA=Y} \rightarrow \mathbf{X^TXA=X^TY} \rightarrow \mathbf{(X^TX)A=X^TY} \rightarrow \mathbf{A=(X^TY)\cdot(X^TX)^{-1}}$.

І знову ми зустрічаємося з особливою матрицею, яку позначили як матрицю в мінус першому степені, причому вона дуже явно пов'язана із знаходженням розв'язку системи рівнянь. Цю властивість ми використаємо в наступній функції, а поки розглянемо поняття оберненої матриці.

Оберненою матрицею стосовно даної називається матриця, що, будучи помноженою як праворуч, так і ліворуч на дану матрицю, дає одиничну матрицю. Для матриці **A** позначимо обернену їй матрицю через $\mathbf{A}^{-1}$. Тоді за визначенням маємо $\mathbf{AA}^{-1}=\mathbf{A}^{-1}\mathbf{A=E}$, де **E** – одинична матриця.

Квадратна матриця називається *неособливою*, якщо визначник її відмінний від нуля. У противному випадку матриця називається *особливою*, чи сингулярною. Усяка неособлива матриця має обернену матрицю, що записується як

$$\mathbf{A}^{-1} = \begin{pmatrix} \frac{A_{11}}{\Delta} & \frac{A_{21}}{\Delta} & \cdots & \frac{A_{n1}}{\Delta} \\ \frac{A_{12}}{\Delta} & \frac{A_{22}}{\Delta} & \cdots & \frac{A_{n2}}{\Delta} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{A_{1n}}{\Delta} & \frac{A_{2n}}{\Delta} & \cdots & \frac{A_{nn}}{\Delta} \end{pmatrix},$$

де A_{ij} – алгебраїчні доповнення відповідних елементів a_{ij}.

Особлива квадратна матриця оберненої не має, тому що її визначник дорівнює нулю. Основні властивості оберненої матриці:

1. Визначник оберненої матриці дорівнює оберненій величині визначника вихідної матриці.
2. Обернена матриця добутку квадратних матриць дорівнює добутку обернених матриць співмножників, узятому в зворотному порядку.
3. Транспонована обернена матриця дорівнює оберненій від транспонованої даної матриці.

Визначивши в такий спосіб обернену для даної матрицю, ми можемо

розв’язати систему рівнянь двома операціями – оберненням і множенням.

```
matrix<YourOwnFloatType> operator*();
```

Обернення матриці описаним вище способом – операція досить трудомістка. Згадаємо, що в процесі розв’язання системи рівнянь ми фактично неявно знаходимо обернену матрицю. Для того, щоб знайти її в явному вигляді, застосуємо наступний прийом: запишемо разом квадратну матрицю й одиничну. Будемо здійснювати такі елементарні перетворення над ними, щоб привести першу матрицю до діагональної форми. В результаті цих перетворень друга матриця буде містити матрицю, обернену до першої.

```
YourOwnFloatType operator&();
operator YourOwnFloatType();
```

Для зручності використання складемо дві операторні функції, що визначають чисельний детермінант як унарну операцію. Тому що детермінант можна розглядати як оператор перетворення матриці до числового значення, запишемо його ще як функцію перетворення до типу.

```
matrix<YourOwnFloatType> operator-();
```

Унарний мінус.

```
matrix<YourOwnFloatType> operator+();
```

Унарний плюс.

```
friend long operator==(matrix<YourOwnFloatType> &, matrix
<YourOwnFloatType> &);
```

Дружня функція перевірки на рівність порівнює дві матриці і повертає ненульове значення, якщо вони рівні. Дві матриці вважаються рівними, якщо вони мають однакові розмірності, і відповідні їхні елементи (вектори-рядки) рівні.

```
friend long operator!=(matrix<YourOwnFloatType> &, matrix
<YourOwnFloatType> &);
```

Дружня функція перевірки на нерівність порівнює дві матриці і повертає ненульове значення, якщо вони не рівні. Операція полягає в перевірці цих матриць на рівність і логічне заперечення отриманого результату.

```
vector<YourOwnFloatType> &operator[](long a);
```

Метод матричного класу для індексування елементів матриці приймає як параметр номер вектора-рядка матриці, яку необхідно одержати. Якщо цей номер коректний, повертається посилання на відповідний вектор, який, у свою чергу, теж можна проіндексувати. Таким чином, одноразове застосування операції індексування дозволяє одержати вектор, а дворазове – відповідний елемент цього вектора. Так як дана операція повертає посилальне значення, її можна використовувати в операціях присвоювання як ліворуч, так і праворуч. При виході індексу за припустимий діапазон значень у стандартний потік виведення вставляється спеціальне діагностичне повідомлення і повертається спеціальна вектор-ознака помилки.

```
long getm() { return m; }
long getn() { return n; }
```

Два службових методи класу для визначення числа рядків і числа стовпців.

```
};
```

Інтерфейсний файл реалізації матричного класу matrix.h наведено у п. 2.3.2 підручника [143].