МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ПЕДАГОГІЧНИЙ УНІВЕРСИТЕТ
імені М. П. ДРАГОМАНОВА

На правах рукопису

ЯЩИК ОЛЕКСАНДР БОГДАНОВИЧ

УДК 37 004(07)

МЕТОДИКА НАВЧАННЯ АЛГОРИТМІЗАЦІЇ ТА ПРОГРАМУВАННЯ СТАРШОКЛАСНИКІВ НА РІВНІ ПОГЛИБЛЕНОГО ВИВЧЕННЯ ІНФОРМАТИКИ

13.00.02 – теорія та методика навчання (інформатика)

ДИСЕРТАЦІЯ
на здобуття наукового ступеня
кандидата педагогічних наук

Науковий керівник
доктор педагогічних наук,
професор
Рамський Юрій Савіянович

Київ – 2016

ЗМІСТ

Список скорочень

ЗНЗ – загальноосвітній навчальний заклад

ЗОШ – загальноосвітня школа

ІКТ – інформаційно-комунікаційні технології

ІТ – інформаційні технології

КС – комп’ютерні системи

НІС – навчально-інформаційне середовище

ООП – об’єктно-орієнтований підхід

ППЗ – педагогічні програмні засоби

СКМ – система комп’ютерної математики

FAQ – відповіді на найбільш часті запитання

LMS – Learning Management Systems (системи управління навчанням)

LCMS – Learning Content Management Systems (системами управління навчанням і навчальним контентом)

MOODLE – Modular Object-Oriented Dynamic Learning Environment (середовище навчання з використанням ІКТ)

SCORM – Sharable Content Object Reference Model (міжнародна основа обміну електронними курсами)

ВСТУП

Інформаційні технології розвиваються дуже стрімко, кількість різноманітних прикладних засобів з року в рік зростає, досягнення інформатики широко проникають в різні сфери людської діяльності. Інформатизація освіти як визначальна ланка загального процесу інформатизації суспільства спрямована на підвищення якості змісту освіти, а також впровадження ІКТ в усі види діяльності в її системі, в тому числі у навчальний процес.

Колись інформатику вивчали лише в старших класах загальноосвітніх закладів. Сьогодні ж вивчення інформатики відіграє провідну роль не тільки у змісті шкільної освіти, а й у подальшому навчанні. Сучасний курс інформатики є результатом великого спектра досліджень, відображених в роботах С. А. Бешенкова [63], В. Ю. Бикова [19], А.Ф. Верланя [7], Ю.В. Горошка [39; 41; 68], А. Г. Гейна [34; 33], А. П. Єршова [62], М. І. Жалдака [64; 65; 73; 74; 70; 69; 75; 68; 67; 72], В.І. Клочка [109; 111], О. А. Кузнєцова [121], А. Г. Кушніренка [128; 129], М. П. Лапчика [131], В.В. Лапінського [72; 205], Ю.І. Машбиця [151], Н. В. Морзе [155; 156; 157], В.М. Монахова [63], Ю. С. Рамського [192; 193; 195; 196; 199; 200; 201; 194], С.А. Ракова [189], В. Д. Руденко [206], З. С. Сейдаметової [212], О. М. Спіріна [234], Ю.В. Триуса [223; 246; 245], Є. К. Хеннера [132; 214], С. М. Яшанова [269] та багатьох інших.

Нині настав новий період розвитку інформатики як міждисциплінарного наукового напряму, що виконує інтеграційні функції для інших напрямів науки, – як природничих, так і гуманітарних. Проникнення ідей і засобів інформатики в ці галузі продиктовано потребами і логікою розвитку власне фундаментальної науки, а також необхідністю розв’язання ряду важливих прикладних проблем. Варто очікувати, що це не тільки дасть новий імпульс для розвитку наукових досліджень інформатики у зв’язку з іншими науками, й збагатить її новими перспективними ідеями.

Однак ці дослідження торкаються лише окремих компонентів системи підготовки з основ інформатики у контексті загальної середньої освіти та підготовки вчителів до використання ІКТ. Крім того, у них не повною мірою представлено зміст предметної галузі «Інформатика», який змінився протягом останніх років, а також соціальний контекст розвитку освіти в Україні наприкінці ХХ ст. – на початку ХХІ ст. Розвиток засобів інформатизації, ІКТ приводить до суттєвих змін інформатики як навчальної дисципліни, що вимагає переосмислення цілей, змісту, засобів і форм підготовки учнів та вчителів з інформатики на сучасному рівні. Ці зміни повинні знайти відображення як у системі загальної освіти, так і у підготовці майбутніх фахівців. Зв'язку з цим міністр освіти і науки України Л. М. Гриневич зазначає: «В програмах необхідно враховувати компетентності, які необхідні дітям для сучасного життя, зокрема критично мислити, аналізувати, працювати в команді, навички фінансової грамотності та підприємництва».

Головною причиною постійного вдосконалення методики навчання інформатики є розвиток цієї науки, сучасних ІКТ та засобів зв'язку. Перехід середніх навчальних закладів до профільної освіти та вищих навчальних закладів до ступеневої освіти є ще однією умовою, яка повинна суттєво вплинути на методику навчання інформатики, розробку нових технологій і систем навчання та удосконалення існуючих, традиційних. Наявність такої умови вимагає перегляду всіх методологічних і концептуальних основ традиційної педагогіки та переходу до неперервної відкритої освіти, заснованої на особистісно орієнтованому навчанні.

Поява навчальних закладів нового типу і перехід загальноосвітніх навчальних закладів на новий зміст і структуру спричинили поглиблення змісту основного курсу інформатики та посилення його прикладної спрямованості. Інформатика на сьогодні є одним із засобів формування не тільки освітнього, а й розвивального та інтелектуального потенціалу особистості. У процесі поглибленого вивчення інформатики основні завдання курсу суттєво розширюються та доповнюються, що обумовлено необхідністю

виявлення та розвитку в учнів логічних здібностей, підготовки їх до участі в олімпіадних змаганнях та наукових дискусіях, формування в них стійкого інтересу до інформатики і пов'язаної з нею професійної діяльності, підготовки до навчання у ВНЗ. Одним із навчально-дослідницьких закладів нового типу є Мала академія наук України, яка забезпечує організацію і координацію науково-дослідницької діяльності учнів, з використанням новітніх ІКТ (зокрема мережні електронні майданчики – В. Б. Дем'яненко).

Проблема нових підходів до вивчення алгоритмізації та основ програмування знайшла відображення в багатьох дослідженнях, більша частина яких спрямована на старшу аудиторію – студентів, а шкільному процесу навчання приділено меншу увагу. Тому сьогодні актуальності набувають розробки нового змісту, форм і засобів навчання алгоритмізації та програмування в ЗОШ, зокрема в школі з поглибленим вивченням інформатики, відповідного дидактичного забезпечення та його науково-методичного і психолого-педагогічного обґрунтування.

Варто зазначити, що на початку історії навчання інформатики навчальні програми в основному були орієнтовані на вивчення тільки мов програмування [181]. Такий підхід не враховував інтереси учнів, яким в подальшій професійній діяльності необхідні були тільки навички користувача.

Нині ситуація різко змінилася. У зв'язку із швидкими темпами розвитку предметної галузі, стрімким розвитком інформаційних технологій, парадигм програмування, удосконаленням комп'ютерної техніки, зростанням потоку надходження даних та удосконаленням можливостей їх опрацювання задачний підхід в навчанні інформатики перестав себе виправдовувати і з'явилася тенденція до вивчення тільки середовищ користувачів. Такі підходи не дають змоги забезпечити відповідну фундаментальну підготовку учнів.

На основі аналізу наукових досліджень, зокрема дисертаційних досліджень, виконаних упродовж 1990–2014 р. р., у процесі навчання алгоритмізації та програмування виявлено низку суперечностей, з яких випливає проблема підвищення якості навчання алгоритмізації та

програмування старшокласників на рівні поглибленого вивчення інформатики (рис. 1).

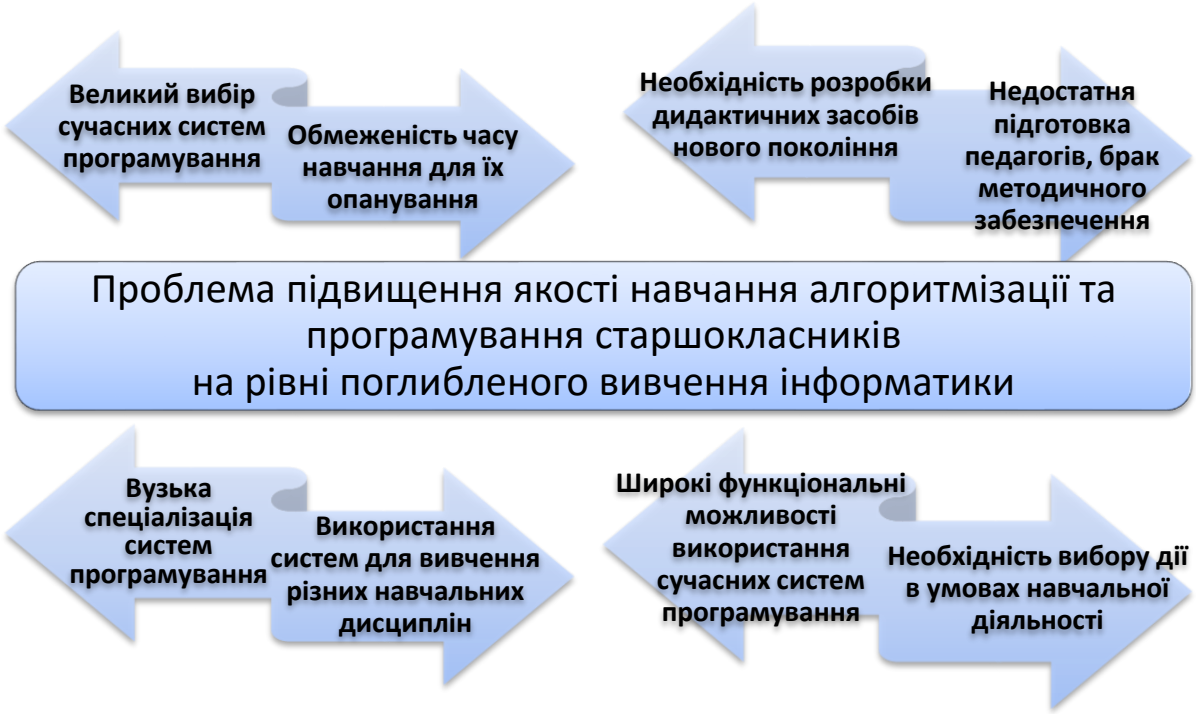


Рис. 1. Суперечності, що підтверджують актуальність дослідження

У зв’язку з цим виникає необхідність вивчати як засоби сучасних ІКТ, які мають суто спеціальну спрямованість (текстові редактори, електронні таблиці, бази даних і т. д.), так і засоби алгоритмізації та програмування. Щоб забезпечити фундаментальну підготовку учнів, навчання потрібно проводити на якісно новому рівні у тісному зв’язку з новітніми ІКТ. Тому актуалізується питання наукового обґрунтування концептуальних підходів, формування пріоритетних ідей в галузі розробки засобів навчання нового покоління, дидактичних засад формування сучасного навчального середовища, забезпечення навчальних закладів разом з апаратними засобами, педагогічними програмними засобами, розробленими відповідно до чинних програм вивчення навчальних предметів, розробки науково-методичного забезпечення та засобів експериментального дослідження для вивчення закономірностей навчально-виховного процесу з використанням засобів

навчання нового покоління, виявлення їх впливу на формування системно-логічного мислення у процесі вивчення шкільних предметів, зокрема інформатики.

Зазначені чинники обумовлюють **актуальність нашого дослідження**, спрямованого на розробку методичної системи, яка в контексті поглибленого курсу інформатики забезпечує засвоєння школярами теоретичних основ алгоритмізації та програмування (понять, принципів, засобів) та їх практичної реалізації з використанням навчально-інформаційних середовищ та програм спеціального призначення.

У проекті Державного стандарту освіти в галузі інформатики зазначено, що розвиток мислення школярів разом з формуванням наукового світогляду і підготовкою учнів до практичної діяльності є однією з функцій освітньої галузі. Проте «програмістський» аспект змісту базового курсу більшою мірою орієнтований на розвиток системно-логічного мислення школярів [181].

У більшості сучасних підручників з інформатики програмування зводиться в основному до навчання складання алгоритмів та знання визначень мови програмування. Такий підхід передбачає вивчення конкретної мови програмування, що можна виконати за кілька годин, натомість розуміння відповідних парадигм програмування вимагає набагато більше часу.

Майбутній фахівець повинен володіти не тільки основами алгоритмізації та програмування, але й уміти використовувати різні середовища програмування, зокрема системи комп’ютерної математики (СКМ). Вміння використовувати такі системи у навчальному процесі ЗОШ дає змогу реалізовувати міжпредметні зв’язки інформатики, математики, фізики тощо і загалом підвищити якість навчального процесу. Це дає підстави вважати тему дослідження актуальною.

Питання вдосконалення змісту й засобів навчання алгоритмізації та основ програмування в шкільному курсі інформатики розглянуто в працях Н.Р. Балик [190], В.Ю. Бикова [19], Л. І. Білоусової [21], В.П. Вембер [89; 92; 93], А.Ф. Верланя [7], Ю.В. Горошка [39; 68; 41], А.М. Гуржія [32],

М.І. Жалдака [64; 73; 74; 75], І.Т. Зарецької [91], І.С. Іваськіва [88], О.Г. Кузьмінської [90; 89; 92; 93], С.О. Лещук [138; 136; 192], М.С. Львова [142; 233], І.М. Лукаш [139; 140], П.М. Маланюка [143; 144], Н.В. Морзе [64; 90; 89; 158], Ю.С. Рамського [195; 196; 191; 222], В.Д. Руденка [206], З.С. Сейдаметової [212], С. О. Семерікова [216; 241], О.В. Співаковського [226; 232; 231; 233], Ю.В. Триуса [223; 246; 245], Г. Ю. Цибко [41] та ін.

Зв'язок роботи з науковими програмами, планами, темами. Дисертаційне дослідження виконане відповідно до тематичного плану науково-дослідної роботи кафедри теоретичних основ інформатики та кафедри інформаційних технологій і програмування Національного педагогічного університету імені М. П. Драгоманова (державна реєстрація № 0105U000448). Тему дисертації затверджено Вченою радою НПУ імені М. П. Драгоманова (протокол № 11 від 27 квітня 2010 р.) та узгоджено Радою з координації наукових досліджень у галузі педагогіки і психології при НАПН України (протокол №7 від 27 вересня 2011 р.)

Об'єктом дослідження є процес навчання алгоритмізації та програмування старшокласників у школах з поглибленим вивченням інформатики.

Предмет дослідження – методична система навчання алгоритмізації та програмування старшокласників на рівні поглибленого вивчення інформатики, орієнтована на використання систем комп'ютерної математики (СКМ).

Мета дослідження полягає в теоретичному обґрунтуванні доцільності використання систем комп'ютерної математики та розробці на їх основі відповідної методичної системи навчання алгоритмізації і програмування, використання якої забезпечить формування системно-логічного мислення та формування компонентів інформатичних компетентностей старшокласників у процесі поглибленого вивчення інформатики.

В основу дослідження покладено **гіпотезу** про те, що методично обґрунтоване цілеспрямоване використання для навчання алгоритмізації та

програмування систем комп’ютерної математики сприяє формуванню інформатичних компетентностей, системно-логічного мислення учнів, активізації навчального інтересу, розвитку особистості в умовах сьогодення і тим самим підвищенню ефективності навчального процесу.

Завдання дослідження:

1. Обґрунтувати доцільність навчання старшокласників алгоритмізації та програмування під час поглибленого вивчення інформатики із застосуванням об’єктно-орієнтованого підходу (ООП).

2. Визначити компоненти інформатичних компетентностей старшокласників, що формуються в процесі навчання алгоритмізації та програмування.

3. Проаналізувати чинні методичні системи навчання алгоритмізації та програмування у старшій школі та з’ясувати шляхи їх вдосконалення.

4. Створити компоненти власної методичної системи навчання алгоритмізації та програмування із застосуванням СКМ, використання якої сприяє розвитку системно-логічного мислення та формуванню компонентів інформатичних компетентностей у процесі навчальної діяльності учнів.

5. Експериментальним шляхом перевірити ефективність використання запропонованої методичної системи та апробувати окремі її компоненти.

Методологічною основою дослідження є Закон України «Про національну програму інформатизації», Державна національна програма «Освіта» (Україна XXI століття), нормативні документи Міністерства освіти та науки України [180]; положення теорії пізнання про взаємозв’язок теорії та практики, про пізнання як активну перетворювальну діяльність людини; дидактичні ідеї особистісно зорієнтованого навчання; принцип психології про єдність свідомості і діяльності; теорія діяльнісного та поетапного підходу до формування прийомів розумової діяльності; основні положення концепції загальної середньої освіти як базової в єдиній системі неперервної освіти та вивчення дисциплін у їх логічному взаємозв’язку, розвитку шкільного курсу інформатики, інформатизації освіти та профільного навчання.

Для розв’язування поставлених завдань використовувались такі **методи дослідження**:

– ***теоретичні:***

1) аналіз філософської, наукової, навчально-методичної та психолого-педагогічної літератури з теми дослідження;

2) аналіз нормативних і програмно-методичних документів у сфері освіти, державних галузевих стандартів середньої та вищої освіти, навчальних програм підготовки вчителя інформатики;

3) аналіз монографій, дисертацій із проблеми дослідження;

4) узагальнення педагогічного досвіду навчання дисциплін інформатичного циклу у педагогічних університетах в умовах впровадження компетентнісного підходу в освіті;

– ***емпіричні:*** спостереження, анкетування, тестування, бесіди з учителями та учнями;

– ***експериментальні:*** констатувальний, пошуковий, формувальний експерименти; статистичне опрацювання результатів педагогічного експерименту та їх аналіз.

Наукова новизна дослідження полягає в теоретичному та експериментальному обґрунтуванні компонентів власної методичної системи навчання алгоритмізації і програмування під час поглибленого вивчення інформатики із застосуванням СКМ, використання якої у навчальному процесі сприятиме формуванню складових системи інформатичних компетентностей старшокласників та ефективному розвитку системно-логічного мислення.

У дисертації досліджено роль інформатики у формуванні мислення учнів, зокрема вплив вивчення алгоритмізації та програмування як невід’ємної частини курсу інформатики в старшій школі на формування типу мислення старшокласників, що називають системно-логічним мисленням.

Теоретичне значення дослідження полягає в розробці компонентів власної методичної системи навчання, використання якої сприяє розвитку системно-логічного мислення учнів під час вивчення алгоритмізації і

програмування на основі об’єктно-орієнтованого підходу, дає змогу створити умови для формування інформатичних компетентностей старшокласників.

Практичне значення дослідження характеризується такими результатами:

– *розроблено* окремі компоненти власної методичної системи навчання алгоритмізації та програмування старшокласників (мета, зміст, засоби навчання, форми) на основі використання навчально-інформаційних середовищ із застосуванням СКМ;

– *розроблено* навчальні матеріали, які містять теоретичні положення, систему практичних завдань, комп’ютерні засоби для вивчення алгоритмізації та програмування з допомогою системи управління навчальними курсами «ІнфоНІС», MOODLE;

– *розроблено* систему практичних робіт для ефективного навчання алгоритмізації та програмування, що сприяє розвитку системно-логічного мислення старшокласників у процесі поглибленого вивчення інформатики.

Апробація результатів дослідження. Основні результати дослідження представлено і обговорено впродовж 2010–2015 р. р. на різних науково-практичних і науково-методичних конференціях і семінарах:

– міжнародних:

1. Міжнародній науково-практичній конференції «Освітні вимірювання в інформаційному суспільстві» (м. Київ, 26-29 травня 2010 р.).

2. Міжнародній інтернет конференції «Впровадження електронного навчання в освітній процес: концепції, проблеми, рішення» (м. Тернопіль, 21-22 жовтня 2010 р.).

3. IV Міжнародній науково-практичній конференції «Актуальні проблеми та перспективи технологічної і професійної освіти» (м. Тернопіль, 23-24 вересня 2011 р.).

– ***всеукраїнських:***

4. Всеукраїнській науково-практичній конференції «Проблеми та перспективи наук в умовах глобалізації» (м. Тернопіль, 7 листопада 2006 р.).

5. VIII Всеукраїнській науково-практичній конференції «Проблеми та перспективи наук в умовах глобалізації» (м. Тернопіль, 15 листопада 2012 р.).

6. Регіональному науково-практичному семінарі «Підготовка фахівців інженерно-педагогічних спеціальностей: досвід, проблеми, перспективи» (м. Тернопіль, 18 квітня 2013 р.).

7. IV Науково-технічній конференції «Інформаційні моделі, системи та технології» (м. Тернопіль, 15-16 травня 2014 р.).

8. Всеукраїнському науково-методичному семінарі «Методика навчання алгоритмізації та програмування старшокласників на рівні поглибленого вивчення інформатики» (м. Київ, 19 травня 2015 р., НПУ імені М. П. Драгоманова).

Результати дослідження **впроваджено** у навчальний процес Тернопільського національного педагогічного університету імені Володимира Гнатюка (довідка № 1066-33/03 від 21.12.2015 р.), Зеленогаєвської ЗОШ І-ІІ ст., Білозерського р-ну, Херсонської обл. (№ 70 від 18.12.2015), Підгаєцької гімназії імені Маркіяна Паславського, Тернопільської обл. (№ 255 від 29.12.2015), Лозівської ЗОШ І-ІІІ ст., Тернопільського р-ну (№ 01-12/2208 від 21.12.2015 р.), Острівської ЗОШ І-ІІІ ст. Тернопільського р-ну (№ 01/12-2207 від 21.12.2015).

Публікації. Основні результати дисертації відображено в 11 публікаціях автора загальним обсягом 8 друкованих аркушів, з них 5 статей у фахових педагогічних виданнях, 1 – іноземна публікація, 1 – публікація у виданнях, віднесених до міжнародних наукометричних баз даних.

Обсяг і структура дисертації. Дисертація складається зі вступу, трьох розділів, висновків до розділів, загальних висновків, списку використаних джерел з 292 найменуваннями (з них 14 іноземними мовами), 8 додатків на 120 сторінках. Загальний обсяг дисертації 352 сторінки, з яких 202 сторінки основного тексту. Робота містить 4 таблиці і 23 рисунки.

РОЗДІЛ 1.
ТЕОРЕТИЧНІ ТА МЕТОДОЛОГІЧНІ АСПЕКТИ НАВЧАННЯ АЛГОРИТМІЗАЦІЇ ТА ПРОГРАМУВАННЯ УЧНІВ СТАРШИХ КЛАСІВ З ПОГЛИБЛЕНИМ ВИВЧЕННЯМ ІНФОРМАТИКИ

1.1. Навчання алгоритмізації та програмуванню як ефективний засіб формування системно-логічного мислення старшокласників

В умовах орієнтації освіти на всебічний і гармонійній розвиток особистості учнів, створення умов для повного розкриття їхніх вподобань і здібностей, посилення зв'язку змісту навчання з повсякденним життям, формування загальних прийомів наукового пізнання та творчого використання сучасних засобів дослідження оточуючого середовища (включаючи засоби ІКТ) особливого значення набуває роль інформатики не тільки як науки, і як навчального предмета. Невід'ємною складовою навчального процесу є використання новітніх ІКТ у навчальному процесі, що відкриває перспективи якісного вдосконалення навчання на основі інтеграції навчальних дисциплін, інтенсифікації навчального процесу й гуманізації його на основі диференціації, професійної спрямованості та індивідуалізації навчання. Тому широке використання в навчальному процесі засобів сучасних комп'ютерних технологій вимагає принципово нового підходу до навчання інформатики [265, с. 137-145].

Важливе завдання при цьому – організація навчання, в якому комп'ютер є не тільки об'єктом, а й засобом навчання. Цей аспект використання обчислювальної техніки в навчальному процесі ґрунтується на тому, що комп'ютер – інструмент опрацювання даних, використання якого може сприяти розумовому розвитку учнів. Основною метою навчання інформатики в школі є – не лише ознайомити дітей з будовою комп'ютера і навчити ним користуватися, а й розвивати інтелект учнів, підвищити продуктивність

виконання поставлених задач за допомогою використання комп'ютерних систем (КС).

Як відомо, на сучасному етапі розвитку суспільства змінюється поняття навчання: просте засвоєння знань поступається місцем умінню користуватися набутими знаннями на практиці. Інформатизація освіти відкриває нові засоби здобуття і використання знань, водночас з цим процесом пов'язаний вагомий психолого-педагогічний аспект: «інформатизація накладає свій відбиток не лише на організацію знання в сучасній картині світу, а й на способи і прийоми мислення» [78, с. 129]. Тому саме у розвитку таких способів і прийомів – перспектива розвитку інформаційного суспільства.

Сьогодні основою високого статусу людини поступово стає розум, талант, уміння опрацьовувати дані і вирішувати складні інтелектуальні завдання. Відбувається перехід від епохи технологічної спадковості, коли майбутнє легко «прораховувалось», до епохи інтелектуальних технологій. Учні мають бути готовими використовувати та створювати ці технології, а завдання вчителя – сприяти цьому.

Відомі вчені, наукові організації та провідні компанії світу запропонували перелік навичок, завдяки яким у XXI сторіччі люди зможуть стати успішними в професійній діяльності та особистому житті. Цей перелік отримав назву «Навички XXI сторіччя». Застосування інформаційних і комунікаційних технологій входить у цей перелік і відноситься до блоку навичок у галузі інформації, засобів зв'язку й технологій, який також включає інформаційну грамотність, компетентність у питаннях передавання даних. Навички в галузі цифрових технологій життєво необхідні, щоб опановувати інструментарій інформаційних, медіа- і комунікаційних технологій, які постійно розширюється. Крім того, ці компетенції активно сприяють розвиткові інших навичок XXI століття.

Відповідно до «Навичок XXI сторіччя» для ефективного формування компетентності в питаннях ІКТ учень повинен уміти:

− застосовувати технології як інструменти пошуку, організації, оцінювання і передачі повідомлень;

− належним чином використовувати цифрові технології (комп'ютери, персональні цифрові секретарі, медіа-плейєри, супутникові навігаційні системи, засоби комунікації та соціальні мережі для отримання, управління, інтегрування, оцінки і опрацювання даних;

− при отриманні доступу до інформаційних ресурсів і їх використанні керуватись основоположними принципами етики та законності [256].

Вивчення алгоритмізації в шкільному курсі інформатики має два цільових аспекти: перший – розвивальний, під яким розуміється розвиток алгоритмічного (ще кажуть – операційного) мислення учнів; другий – програмістський. Якщо в перших шкільних підручниках інформатики у вивченні алгоритмізації превалював другий аспект, то згодом розвиваючий аспект став актуальнішим. Програмістський аспект має профорієнтаційний характер, оскільки професія програміста в наш час дуже поширена і престижна, а вивчення програмування в межах шкільного курсу інформатики дає змогу учням адаптувати свої здібності до такої діяльності. Безумовно, більшою мірою це завдання може вирішити поглиблений курс інформатики в старших класах.

Насамперед визначимо специфіку системно-логічного мислення. Воно є однією з нових і найменш досліджених категорій сучасної педагогічної теорії і практики, тому для виявлення її особливостей необхідний аналіз смислової значущості термінів, що визначають її вихідні складові: «мислення», «логічне мислення», «системний підхід» [265, с. 137-145].

Мислення є вищим пізнавальним процесом. У філософському розумінні мислення – це «вища форма активного віддзеркалення об'єктивної реальності, що полягає в цілеспрямованому, опосередкованому і узагальненому пізнанні суб'єктом істотних зв'язків і співвідношень предметів і явищ в творчому творенні нових ідей, в прогнозуванні подій і дій» [115]. Мислення трактують як продукт історичного розвитку суспільної практики, як особливу теоретичну

форму людської діяльності. У філософії, психології і педагогіці розрізняють такі форми мислення: наочно-образне, наочно-дієве і словесно-логічне. Форми мислення трактуються філософією «як способи і види формальної організації розумового процесу, абстраговані від його змістової компоненти» [148]. Якщо ж враховується змістова компонента, то тут вже активізується мислення теоретичне і практичне, теоретичне та емпіричне, логічне (аналітичне) й інтуїтивне, продуктивне та репродуктивне, мимовільне й довільне. Зокрема, логічне мислення, яке у широкому розумінні називають дискурсивним, передбачає логічний перехід від одного певного визначення до іншого. Логічне мислення істотно відрізняється від інтуїтивного – такого, що пізнає світ шляхом споглядання і встановлює істину без доказу. Вивчення алгоритмізації та програмування сприяє формуванню в учнів логічного мислення.

У Короткому словнику системи психологічних понять – логічне мислення тлумачиться як «тип мислення, суть якого полягає в оперуванні поняттями, думками і висновками з використанням законів логіки» [116].

Н.А. Підгорецька конкретизує це тлумачення і додає, що «уміння логічно мислити включає ряд компонентів: уміння орієнтуватися на істотні ознаки об’єктів і явищ, уміння підкорятися законам логіки, будувати свої дії відповідно до них, уміння проводити логічні операції, усвідомлено їх аргументувати, уміння будувати гіпотези і робити висновки з даних наслідків і т.д.» [175]. Цим тлумаченням ми і керуватимемося в даному дослідженні.

Логічне мислення вивчається в багатьох науках: філософії, психології, кібернетиці, педагогіці і кожна з них досліджує його в певному аспекті, який характерний саме для неї. Проаналізуємо ці аспекти інтерпретації суті та розвитку логічного мислення.

У філософії розглядають логічне мислення з позиції теорії пізнання, формальної логіки, що вивчає форми і закони правильного мислення, і за допомогою діалектики, що дає загальний засіб дослідження логічного мислення як розвиваючого процесу.

З психологічного погляду, логічне мислення – це активна діяльність суб'єкта; виявлення спонукальних мотивів, цілей, які мають індивідуальну значущість та особливості логічного мислення; дослідження розумових операцій з метою усвідомлення суб'єктом логічних принципів, що лежать в їх основі. Тому психологічний аспект розвитку логічного мислення передбачає цілеспрямовану діяльність у вищеназваних напрямах.

У кібернетиці логічне мислення необхідне для розв'язування задач технічного моделювання розумових операцій у формі «штучного інтелекту», а також в тих аспектах мислення, які пов'язані зі швидким й ефективним опрацюванням даних за допомогою комп'ютерних систем. Таким чином, кібернетичний аспект розвитку логічного мислення обумовлений насамперед процесами моделювання розумових операцій.

В педагогіці, як відзначає А. Д. Гетманова [38], логічне мислення вивчається у контексті здійснення процесу пізнання під час навчання і виховання підростаючого покоління. Отже, педагогічний аспект розвитку логічного мислення учнів полягає в розробці і експериментальній перевірці необхідних педагогічних умов організації процесу навчання.

Розглянемо наступні характеристики логічного мислення, що розкривають особливості протікання розумових процесів і забезпечують продуктивність розумової діяльності:

• гнучкість (рухливість), пов'язана з умінням змінити намічений шлях розв'язування задачі, якщо він не відповідає тим умовам, які плануються в процесі розв'язування і не можуть бути враховані від початку, що виявляється в активній перебудові вихідних даних, розумінні і використанні їх відносності;

• швидкість – протікання розумових процесів;

• самостійність – вміння побачити і поставити нове запитання, а потім розв'язати його своїми силами;

• економічність мислення – визначається числом логічних ходів, за допомогою яких засвоюється нова закономірність;

• ширина розуму – вміння охопити широке коло питань у різних галузях знань та практики;

• глибина – вміння вникати в суть, розкривати причини, передбачати наслідки;

• послідовність думки – уміння дотримувати чіткого порядку в розгляді того чи іншого питання;

• критичність – якість мислення, що дає змогу здійснювати строге оцінювання результатів розумової діяльності, знаходити в них сильні і слабкі сторони, доводити істинність висунутих положень.

Разом з тим, варто зазначити, що більшість так званих «якостей розуму» належать не стільки до характеристик продуктивності власне розумової діяльності, скільки до властивостей особистості більш високого рівня, що обумовлюють продуктивність пізнавальної діяльності в цілому.

Визначимо суть будь-якого пізнавального процесу, який базується на логічному мисленні, використовуючи схему процесу пізнання за Р.Ф. Абдєєвим [1, с. 58], наведену на рис. 1.1.

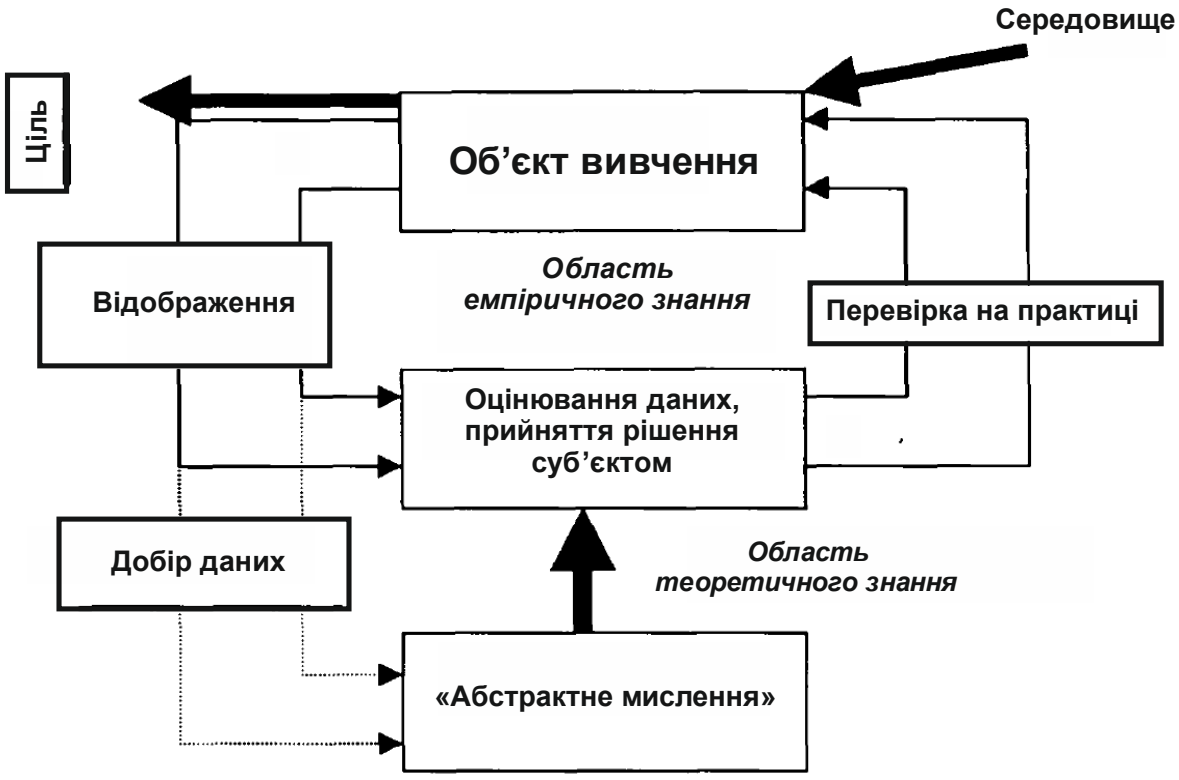


Рис. 1.1 Схема процесу пізнання

В основу процесу пізнання покладено два контури зворотного зв'язку людського мислення: перший – середовище емпіричного мислення; другий – добір та узагальнення повідомлень, спроба виявити чергову відносну істину, тобто середовище логічного мислення, яке сприяє формуванню теоретичних знань. Інакше кажучи, в першому контурі сприймається явище, а в другому пізнається сутність об'єкта. Така схема відображає механізм руху пізнання від відносної істини до наукових фактів, від конкретизації до узагальнення, де кожний поточний результат оцінюється на тлі постійно оновлюваних знань.

Таким чином, щоб в учнів сформувалася здатність логічно думати, потрібно навчитися систематизувати набуті знання.

«Системний підхід – напрям методології наукового пізнання і соціальної практики, в основі якого лежить розгляд об'єктів як систем; орієнтує дослідження на розкриття цілісності об'єкта, виявлення багатообразних типів зв'язку в ньому і групування їх в єдину теоретичну картину. Системний підхід нерозривно пов'язаний з діалектикою, є конкретизацією її основних принципів» [126, с. 12]. Принцип системності широко використовується в інформатиці, математиці, біології, екології, психології, кібернетиці, техніці, економіці, управлінні та інших галузях.

У психології і педагогіці визначають системний підхід як положення про те, що специфіка складного системного об'єкта (системи) не вичерпується особливостями складових її елементів, а пов'язана передусім з характером взаємодій між елементами. Тому на перший план висувається завдання пізнання характеру і механізму цих зв'язків і співвідношень. У процесі системного аналізу виявляються не лише причини явищ, а й зворотна дія результату на породжувані його причини.

«Системний аналіз – це процес створення певного «узагальнюючого образу». Він вбирає в себе сукупності взаємопов'язаних етапів, які розробляються в різних галузях науки для опису об'єктів», – зазначає Ю.А. Самоненко [211]. Серед таких етапів вчений виокремлює: визначення якісної своєрідності системи, тобто сукупності її основних властивостей;

встановлення функцій системи; виокремлення структури системи з вичленennям її елементів, що мають «горизонтальний» і «вертикальний» характер; встановлення серед зв'язків, необхідних для існування системи, так званих системо-утворюючих; встановлення компонентів і їх властивостей, складових як внутрішніх так і зовнішніх середовищ стосовно системи; визначення меж стійкості системи; опис динаміки кількісної міри властивостей системи, яку можна назвати «поведінкою» системи і т. д. [210].

Застосування системного підходу в процесі навчання алгоритмізації та програмування активізує розвиток логічного мислення старшокласників, дає змогу розглядати його як цілісну багаторівневу категорію, як специфічний вид діяльності, що забезпечується різними психічними процесами: сприйняттям, пам'яттю, уявою, з обов'язковим використанням мови. У сучасній психологічній і педагогічній літературі наголошується на впливі вивчення інформатики, зокрема алгоритмізації та програмування, на розвиток логічного мислення [21, с. 74-82; 48; 85; 185; 160].

Зазначимо, що суть системного підходу відображається в наступних його характеристиках, які допомагають встановлювати властивості системних об'єктів і вдосконалювати їх:

1. доцільність системи відносно зовнішнього середовища, її вивчення в єдності з середовищем;

2. розчленовування цілого, що приводить до виокремлення елементів. Властивості елементів залежать від їх належності до певної системи, а властивості системи не зводяться до властивостей її елементів;

3. всі елементи системи перебувають в складних зв'язках і взаємодіях, серед яких потрібно виокремити найбільш актуальні, визначаючі для даної системи, тобто системо-утворюючу ланку;

4. сукупність елементів і зв'язків дає уявлення про структуру і організацію системних об'єктів. Ці поняття виражають певну впорядкованість системи, взаємозв'язок і взаємозалежність її елементів;

5. спеціальним способом регулювання зв’язків між елементами системи і тим самим зміни і самих елементів є управління, що включає постановку цілей, добір засобів, контроль і корекцію, аналіз результатів [265, с. 6].

У світлі вимог системного підходу для теоретичного дослідження формування системно-логічного мислення учнів старшої профільної школи важливого значення набуває також розвиток системної дедукції. Така дедукція передбачає рух думки від вихідних теоретичних передумов до узагальнених висновків, а від них – до усе більш конкретних наслідків, які перевіряються як доведенням, так і досвідом. Це дає підставу для цілісного розуміння предмету дослідження на теоретичному рівні. Як бачимо, принципове значення для педагогічних досліджень розвитку мислення, зокрема системно-логічного мислення учнів, має вимога вивчення даного процесу як цілого.

Як наголошується в останніх філософських дослідженнях, «розвиток цілісності здійснюється за рахунок:

– розвитку кожного елемента;

– зростання значення кожного елемента для розвитку і функціонування один одного;

– підпорядкування кожного елемента системі;

– набування здатності кожного елемента виконувати функцію заради розвитку системи в цілому;

– виникнення нових властивостей системи, якими не володіють окремі елементи;

– появи у системі змістовно нових елементів для більш повного виконання функцій іншими елементами і системою в цілому» [166].

Поняття «системний стиль мислення» увів І. Новік [161, с. 54], науковець відзначає важливі характерні риси цього стилю мислення:

– він відображає загальнонаукові тенденції процесу пізнання певної епохи;

– він є своєрідним компасом, критерієм науковості, використовуючи

який можна достатньо ефективно оцінювати міру відповідності конкретному історичному періоду тих або інших наукових ідей, моделей, гіпотез, досліджень в різних галузях науки;

– він виражає світоглядну позицію людини;

– він виступає об'єктивною інтеграційною тенденцією нашої епохи;

– він науково обґрунтованою стратегією з управління складними і надскладними об'єктами будь-якої природи.

В. В. Черников під системним мисленням розуміє «віддзеркалення об'єктивної реальності, що полягає в цілеспрямованому пізнанні суб'єктом істотних зв'язків і співвідношень, які існують в певному явищі, що обумовлюють єдність його форми і змісту, його стійке функціонування в зовнішньому середовищі, а також полягає в творчому пошуку нових ідей, в прогнозуванні подій і дій через ідеї системного підходу, системного аналізу» [263].

Таким чином, підсумовуючи, можна стверджувати, що системно-логічне мислення – це нова категорія свідомості, підґрунтям якої є вміння швидко і якісно, враховуючи швидкий розвиток сучасних ІТ, розв'язувати складні завдання. Водночас, це новий щабель психологічних і психофізіологічних можливостей людини в галузі мислення, який робить можливим опрацювання та засвоєння великих масивів даних, визначення відповідних певній ситуації відомостей, отримання нових знань. Системний підхід є продуктивним у формуванні такого типу мислення учнів, оскільки належить до так званих нелінійних систем (при зміні одного з елементів структури інші змінюються не пропорційно, а за складнішим законом).

Вивчення алгоритмізації та програмування в шкільному курсі інформатики спрямоване на підвищення інтересу учнів до навчання, збільшення їхньої активності і мотивації. Розробка алгоритмів і програм «має велике значення для розумового розвитку школярів» [166]. У процесі навчання алгоритмів формується як аналітичне, так і системно-логічне мислення, яке, зрештою, служить запорукою формування інформаційної культури учнів.

Розглянемо, як саме здійснюється розвиток системно-логічного мислення у ході формування в учнів навичок створення алгоритмів та написання програм. Розв'язування будь-якої задачі починається визначення проблеми і добору даних із сформульованої задачі. У процесі добору даних і з визначення послідовності кроків, які відповідають вимогам до майбутнього розв'язку завдань, в учнів формуються уміння бачити в будь-якому явищі його інформаційну складову виокремлювати фактори, які впливають на розв'язування поставленої задачі та правильне її оцінювання. Під час формалізації поставленої проблеми, тобто при розробці інформаційної моделі і структуризації даних, формується розуміння та сприйняття даних. Побудова алгоритму сприяє розвитку навичок подання цього процесу у вигляді скінченої послідовності кроків, які ведуть до мети, і екстраполяції вказаного підходу на інші задачі процесу навчання. Відбувається формування навичок впорядковування, систематизації, структуризації даних, способів їх подання.

Створюючи алгоритм чи програму, учень передбачає, як вони повинні працювати. Проте часто, через певні логічні помилки або неточності в розрахунках, реалізація створеного продукту надає результат, що не відповідає очікуванням учня. Протиріччя, яке виникло, змушує його аналізувати свою роботу та знаходити помилки. Для розвитку мислення поява «протиріччя» – це сигнал появи проблеми, нерозв'язної за допомогою вже відомих інтелектуальних дій, сигнал для включення мислення як самостійного осмислення предмета [37, с. 21–30.]. Відповідно, це стимулює мислення учнів та систематизує уже наявні знання з метою осягнення чогось нового. Крім того, різноманітність виконавців алгоритмів і мов програмування спонукає учня чітко формулювати свою думку рідною мовою, а тоді вже записувати її за допомогою конкретного виконавця чи мови програмування. У класах поглибленого вивчення інформатики, орієнтованих на алгоритмізацію та програмування, корисно зупинитися на розгляді прикладу роботи машини Тюрінга. Побудована під керівництвом Алана Тюрінга у Великій Британії

обчислювальна система ACE (завершена в 1950 році), була одним з перших універсальних обчислювальних пристроїв, яка повністю відповідав вимогам обчислювальності. Описуючи різноманітні алгоритми для машин Тюрінга і стверджуючи реалізованість усіляких композицій алгоритмів, А. Тюрінг переконливо показав розмаїтість можливостей використання запропонованої ним конструкції, що дало змогу йому заявити: «Будь-який алгоритм може бути реалізований відповідною машиною Тюрінга» [253, с. 12–15]. Це основна гіпотеза теорії алгоритмів у формі Тюрінга. Одночасно ця теза є формальним визначенням алгоритму. Завдяки їй можна доводити існування або неіснування алгоритмів, створюючи відповідні машини Тюрінга або доводячи неможливість їхньої побудови. Завдяки цьому з'являється загальний підхід до пошуку алгоритмічних розв'язків. Вивчення машини Тюрінга дає змогу розвивати операційне мислення, а процес складання алгоритму як послідовного ланцюжка елементарних дій дисциплінує за розробкою конкретних алгоритмів і програм.

Таким чином, структурованість та чіткість алгоритмічного виконавця чи мови програмування, які використовуються старшокласниками у процесі роботи, позитивно впливають на формування і розвиток їхнього логічного мислення. Вивчення алгоритмізації та програмування сприяє формуванню в учнів:

- загальних логічних прийомів мислення (індукція, дедукція, аналіз, синтез, узагальнення, абстракція та інші);
- спеціальних для інформатики прийомів розумової діяльності, які є основою засобів пізнання інформаційного простору (різні способи абстракції, узагальнення, конкретизації, засіб побудови моделей процесів, прийом прогнозування та виокремлення смислових опорних пунктів та інші);
- системи знань як результату діяльності вчителя і учня, здобутий мисленєвою діяльністю.

Оволодіння структурним стилем мислення формує в учнів певні уміння і навички наприклад, використання знання технологій низхідного і висхідного

проектування програм є неможливим без оволодіння засобами індукції та дедукції. Програма будується з логічно пов'язаних компонентів, використовуючи обмежений набір конструкцій, тому без оволодіння прийомами аналізу та синтезу не можна забезпечити повну відповідність даних між програмними компонентами різних рівнів. Таким чином, формування системно-логічного мислення – це процес розвитку здібностей до роздумів, який відбувається в процесі створення алгоритмів і програм шляхом мисленєвих дій. Отже, системно-логічне мислення – це мисленєва діяльність управління пошуком розв'язування задач, результатом якого є алгоритми та програми як специфічні продукти.

Проведений аналіз специфіки формування системно-логічного мислення учнів старших класів та місця алгоритмізації і програмування в цьому процесі дає змогу зробити висновок, що ґрунтовне вивчення цих розділів інформатики є інструментом розвитку розумових здібностей школярів, точніше – інструментом пізнання в процесі навчання. Створення ефективної методики їх навчання в школі вимагає врахування системно-логічного аспекту, що передбачає вміння учнів здійснювати добір та аналіз даних, виявляти внутрішні ієрархічні зв'язки між інформаційними одиницями в інформаційних масивах, використовуючи відповідні засоби і теоретичні способи спеціальної науки, практичні засоби автоматизації інформаційних процесів, комп'ютерні і мікропроцесорні системи. Отже, наявність системно-логічного мислення виступає передумовою формування інформаційної культури як системи інформатичних компетентностей старшокласників.

1.2. Формування інформатичних компетентностей учнів старшої школи в процесі навчання алгоритмізації та програмування

Використання компетентнісного підходу в освіті зумовлено необхідністю визначити адекватність сучасної системи освіти потребам завтрашнього дня. Мета цього підходу – встановити взаємозв'язок між вимогами ринку праці і результатами шкільного навчання та виховання. У зв'язку з цим важливими стають уміння працювати в команді, творчо мислити, здійснювати самоосвіту, брати на себе відповідальність за прийняття рішень, володіти навичками використання інформаційних технологій тощо. Усі ці характеристики особистості мають бути складовими системи освітніх компетентностей учнів нової формації.

Однак не завжди компетентнісний підхід в освіті був визначальним: він практично не використовувався при побудові типових навчальних програм, стандартів, оцінювальної системи. Тільки в середині 1990-х років поняття «компетентнісний підхід» починає визначати вимоги до підготовки учнів в школі. Під цим терміном розумівся «підхід, при якому результати освіти визнаються значущими за межами системи освіти» [31].

Компетентнісний підхід у визначенні цілей і змісту освіти, орієнтованих на набуття вмінь, узагальнених способів діяльності, є провідним в роботах таких педагогів, як Н. Р. Балик [197], Н. С. Веселовської [31], В. Б. Дем'яненко [53], М. І. Жалдака [69, 75], Кузьміної Н. М. [124], О. В. Овчарук [114], Т. В. Підгорної [173], Рамського Ю. С. [197; 201], О. В. Співаковського [171], О. М. Спіріна [234], С. В. Трішіна [248], А. П. Тряпіціної [179], А. В. Хуторського [261] та інших. Проте в педагогічній науці досі відсутнє однозначне визначення поняття «компетентність», немає чітких критеріїв для виявлення основних компонентів системи компетентностей старшокласників, не достатньо вивченими залишаються роль та місце інформатики у формуванні ключових та власне інформатичних компетентностей учнів. Тому в ракурсі нашого дослідження актуальним є

питання специфіки формування інформатичних компетентностей (знань, умінь, навичок, здібностей, особистісних якостей) учнів старших класів у процесі навчання алгоритмізації та програмування.

Розглядаючи компетентнісний підхід, необхідно розрізняти два поняття: компетенція і компетентність. Компетенція «включає сукупність взаємозалежних якостей особистості, що задаються стосовно визначеного кола предметів і процесів». До неї також відносять коло повноважень певної особи, організації, установи. У межах своєї компетенції особа може бути компетентною або не компетентною у тих чи інших питаннях [69, с. 4].

Компетентність проявляється у «володінні людиною відповідною компетенцією, що включає її особистісне відношення до неї і предмету діяльності» [187]. Під компетентністю розуміють комплекс знань, умінь, навичок, досвіду застосування їх для здійснення діяльності, метою якої є досягнення певних цілей, ставлення до процесу та результатів виконання цієї діяльності [224, с. 66].

У цьому ж контексті функціонує поняття «освітня компетентність», що розуміється як «сукупність значеннєвих орієнтацій, знань, умінь, навичок і досвіду діяльності учня стосовно визначеного кола об’єктів реальної дійсності, необхідних для здійснення особистісної і соціально-значущої продуктивної діяльності» [114]. Освітні компетентності диференціюються за рівнями, як і зміст освіти:

– ключові (реалізовані на метапредметному, загальному для всіх предметів змісті);

– загальнопредметні (реалізовані на змісті, що є інтегративним для сукупності предметів, освітньої галузі);

– предметні або специфічні (сформовані в межах окремих предметів).

Ключові компетентності необхідні для продуктивної суспільно корисної діяльності будь-якого учня. Вони проявляються в «здатності вирішувати професійні завдання на основі використання даних, комунікації...» [115, с. 9]. Професійні компетентності передбачають окрім

технологічної підготовки інші компоненти, які мають, переважно, позапрофесійний або надпрофесійний характер, але необхідні сьогодні певною мірою кожному фахівцю. Це якості мислення (гнучкість, абстрактність, системне й експериментальне мислення), якості особистості (самостійність, здатність приймати відповідальні рішення, творчий підхід до справи, вміння доводити її до кінця, вміння постійно вчитися і оновлювати свої знання), комунікативні якості (комунікабельність, здатність до співпраці, вміння вести діалог) та інші. Такі якості, необхідні для вивчення алгоритмізації та програмування в старшій школі і належать до ключових компетентностей. Педагог сприятиме розвитку компетентностей в учня за умови, що він сам володіє ними на достатньо високому рівні. Тому варто говорити про компетентності, якими повинні володіти вчителі інформатики у галузі алгоритмізації та програмування. Детально це питання висвітлено у статті М. І. Жалдака, Ю. С. Рамського, М. В. Рафальської [75]. Автори зазначають, що для здійснення успішної індивідуальної, суспільної, трудової діяльності кожен учитель інформатики має набути соціально та професійно значущих компетентностей.

У структурі системи соціально-професійних компетентностей фахівця науковці виокремлюють чотири блоки, перші два з яких є базовими і необхідними для формування двох наступних:

1) *інтелектуальний* – сукупність сформованих у випускника прийомів розумової діяльності (аналіз, синтез, порівняння, співставлення, класифікація, систематизація, узагальнення та ін.);

2) *особистісний* – особистісні якості випускника (відповідальність, цілеспрямованість, самостійність, організованість тощо);

3) *соціально-значущі компетентності* – компетентності, володіння якими дає змогу забезпечити життєдіяльність випускника у сучасному світі та його взаємодію з іншими людьми, групою, колективом;

4) *професійні компетентності* – компетентності, набуття яких дає змогу випускнику виконувати професійну діяльність.

Отже, погоджуємося із словами М. І. Жалдака про те, що формування професійних компетентностей вчителя інформатики передбачає набуття ним компетентностей у галузі інформатики та суміжних з нею дисциплін, методики навчання та дидактики, психологічних і педагогічних основ здійснення навчально-виховного процесу, дослідницької діяльності та педагогічного спілкування, що визначає якість його професійної діяльності [75 с. 44-52].

Розглянемо детальніше проблему формування інформатичних компетентностей в учнів під час навчання алгоритмізації і програмування у класах з поглибленим вивченням інформатики.

Як відомо, «базові компетентності відображають специфіку певної професійної діяльності» [179, с. 9]. А. П. Тряпіцина характеризує базові компетентності як розвиток інтегральної характеристики особистості, здатність вирішувати завдання, спираючись на особистий досвід, набутий в різних ситуаціях, не тільки в навчанні, й у процесі спілкування з друзями, дозвільної діяльності і т. д.; здатність вирішувати завдання, які спираються, безперечно, на засвоєні знання і сформовані вміння, і, нарешті, це здатність людини адекватно оцінювати себе в ситуації, коли вона повинна вирішувати завдання та приймати рішення [179, с. 9].

Спеціальні або предметні компетентності відображають специфіку конкретної предметної або надпредметних сфер діяльності. Спеціальні компетентності можна розглядати як реалізацію ключових і базових компетентностей в сфері навчального предмета, конкретної наукової галузі. Спеціальні компетентності складають варіативну частину структури компетентностей учнів, тому що пов’язані з певним видом діяльності.

Розвиток науки і техніки, автоматизація всіх сфер людської діяльності вимагає від учнів уміння здобувати нові знання шляхом використання ІТ в навчальній та майбутній професійній діяльності. Тому, в сучасних педагогічних дослідженнях набуває актуальності проблема формування інформатичної компетентності учнів, що належить до ключових

компетентностей та відображає рівень знань та умінь учнів в галузі інформатики. Зазначимо, що компетентність у сфері інформатичних та комунікаційних технологій включена у перелік ключових компетентностей, які визначені науковцями міжнародної спільноти на Лісабонській конференції 2001 року [168].

На думку Є. М. Смирнової-Трибульської, інформатичні компетентності – це знання і необхідні уміння застосування ІКТ для розв'язування особистісно значимих задач в галузі освіти і майбутньої професійної діяльності. Вчена виводить таку формулу компетентності***: Компетентність = Мобільність знання + Гнучкість методів + Критичність.*** Крім цього, вона акцентує увагу на двох послідовних напрямах формування інформатичних компетентностей: *по-перше,* учень здобуває елементарні знання та необхідні уміння застосування комп'ютера для розв'язування різноманітних завдань; *по-друге,* він самостійно розширює і поглиблює знання в галузі інформатичної освіти, здобуті раніше [225 с. 17].

Сформованість інформатичних компетентностей вказує на здатність учня орієнтуватися в інформаційному середовищі, тобто володіти вмінням працювати з різноманітними інформаційними ресурсами, знаходити необхідний матеріал, опрацьовувати його та вміти вирішувати поставлену проблему. Погоджуючись з думкою Н. Баловсяк [18, с. 22], розглянемо в інформатичних компетентностях наступні три основні складові:

– *інформаційні компетентності* (здатність ефективної роботи з даними у всіх формах подання);

– *комп'ютерні або комп'ютерно-технологічні компетентності* (уміння та навички роботи з сучасними комп'ютерними засобами та програмним забезпеченням);

– *процесуально-діяльнісні компетентності* (здатність застосовувати сучасні засоби ІКТ у роботі з даними та розв'язуванням різноманітних задач). Ця класифікація засвідчує, що рівень інформатичної компетентності залежить від рівня володіння ІКТ.

Розглянуті нами інформатичні компетентності інтегрують *знання* про основні засоби інформатики та інформаційних технологій, *уміння* використовувати наявні знання для розв'язування прикладних задач, *навички* використання комп'ютера і технологій зв'язку, *здатності* подавати повідомлення і дані у зрозумілій для всіх формі, які *виявляються* у *прагненні*, *здатності* і *готовності* до ефективного застосування сучасних можливостей використання ІКТ для розв'язування завдань як у навчальній діяльності, так і в повсякденному житті.

Одним із ефективних засобів формування в учнів старших класів інформатичних компетентностей є вивчення алгоритмізації та програмування. Розвиток умінь та навичок роботи з ІКТ є складним процесом, тому структура системи інформатичних компетентностей має кілька компонентів.

1. Мотиваційний компонент – внутрішня переконаність в необхідності здобуття нових знань у галузі застосування, а також можливостей використання ІТ в процесі навчання алгоритмізації та програмування; розуміння значущості цих знань для успішності навчання та подальшого ефективного формування професійних навичок.

2. Когнітивний (інформаційний, змістовий) компонент – знання основ інформатики, новітніх ІТ; розуміння закономірностей та особливостей протікання інформаційних процесів; знання сфер застосування і можливостей використання сучасних ІТ для розв'язання поставлених завдань; знайомство з технологіями об'єктно-орієнтованого програмування.

3. Технологічний (поведінковий, діяльнісний) компонент – вміння вибирати і формулювати мету своєї діяльності і здатність у відповідності з поставленими цілями впорядковувати, систематизувати, структурувати дані; вміння пов'язувати розрізнені факти; володіння основами алгоритмізації для виконання постановки завдання перед програмістом; володіння основами технологій створення програмного забезпечення; вміння правильно використовувати інформаційні ресурси, використовувати інформатичні технології для підвищення ефективності своєї освітньої діяльності.

4. Особистісний (емоційно-вольовий) – брати на себе відповідальність і приймати рішення; вміння приймати рішення про застосування того чи іншого програмного забезпечення; психологічна готовність до використання комп'ютера для вдосконалення власної праці; розвиток рефлексивних здібностей: вміння оцінювати свою діяльність і рівень власного розвитку [120, с. 31].

С. В. Трішіна розглядає ще один компонент інформатичної компетентності – комунікативний, який передбачає уміння використовувати технічні засоби комунікації в процесі передачі відомостей від однієї людини до іншої за допомогою різних форм і способів спілкування [248].

Розглянемо специфіку зазначених компонентів структури інформатичної компетентності стосовно навчання алгоритмізації та програмування у старшій школі.

Зазначимо, що поряд з наведеними вище компонентами вагому роль відіграє розвинуте системно-логічне мислення учнів. За його допомогою учень може сприймати і розуміти навчальний матеріал, задачу тощо, розкривати зв'язки та закономірності між досліджуваними об'єктами, що допомагає сформувати цілісне уявлення про дану проблему і способи її розв'язання. На цій основі формується переконаність в значущості інформаційних технологій для здобуття нових знань, тобто реалізовується мотиваційний компонент.

Загальними завданнями програмування є уявне виокремлення і поєднання окремих властивостей об'єкта, скріплення розрізнених фактів і подій в єдиний ланцюг, а це в процесі роботи з даними означає використання різноманітних способів опрацювання даних, їх структуризацію і систематизацію. Таким чином, будується модель реальної системи та відбувається приріст знань (когнітивний компонент).

У процесі діяльності над створенням моделі об'єкта освоюються технології опрацювання даних, що сприяє формуванню технологічного компонента. Виконання системою інформатичних компетентностей операцій

узагальнення як операції системно-логічного мислення, що виражається в навчанні системного підходу до розв'язування поставлених задач, розкритті зв'язків та закономірностей в роботі з даними, сприяє формуванню нових знань, навиків рефлексії. Під час пошуку і виправлення помилок при складанні програм розвивається відповідальність до роботи з будь-якими даними. Для цього етапу характерний прояв позитивних емоцій на основі особистих перемог і досягнень при одержанні очікуваного результату, що сприяє підвищенню самооцінки особистості, зростанню мотивації для використання алгоритмізації та програмування у процесі навчання як в школі, так і в майбутній професійній діяльності.

Щоби старшокласники були здатні до кваліфікованого розв'язання алгоритмічних завдань та успішної навчальної діяльності, на уроках інформатики під час поглибленого вивчення алгоритмізації та основ програмування, вони повинні набути певних ключових інформатичних компетентностей. Ю.С. Рамський та М.А. Умрик [191, с. 16-25] детально розглядають інформатичні компетентності у галузі алгоритмізації і програмування. Для учнів старших класів, які поглиблено вивчають інформатику, актуальними будуть такі:

– володіти поняттям алгоритму та знати його властивості;

– володіти уявленнями про формалізацію понять алгоритм, алгоритмічні системи, алгоритмічно нерозв'язні проблеми;

– вміти обирати способи та форми подання алгоритму;

– вміти оперувати основними базовими структурами при складанні алгоритмів;

– вміти класифікувати алгоритми за змістом виконуваних дій та за структурою;

– вміти застосовувати метод послідовного уточнення алгоритму при розв'язуванні задач;

– вміти встановлювати порядок складання та правила запису алгоритмів та програм;

– вміти застосовувати різні форми опису алгоритмів і переходити від однієї форми опису до іншої;

– вміти використовувати прості й складні умови при побудові алгоритмів і програм;

– вміти описувати алгоритми розв’язування задач різних типів навчальною алгоритмічною мовою та мовою програмування;

– вміти складати й реалізовувати алгоритми з різними типами даних;

– вміти визначати можливості застосування виконавців для розв’язування задачі на основі системи команд виконавця;

– вміти розробляти алгоритми для навчальних виконавців, використовувати оператори мови програмування високого рівня для розв’язування задач;

– вміти аналізувати складність алгоритму;

– вміти проводити лабораторні експерименти для оцінювання алгоритмічної ефективності, аналізувати загальні підсумки роботи, порівнювати ці результати з наміченими на початку роботи, виявляти причини відхилень і намічати шляхи їх усунень;

– вміти оцінювати свою діяльність і діяльність інших, розподіляти роботу при спільній діяльності;

– володіти навичками користувача персонального комп’ютера;

– володіти програмістськими вміннями;

– володіти сучасними технологіями програмування;

– володіти засобами програмування машинно-орієнтованих мов;

– володіти основами логічного програмування;

– володіти технологією об’єктно-орієнтованого програмування;

– володіти основами систем візуального програмування.

Варто зазначити, що сукупність інформатичних компетентностей динамічна, оскільки вона передбачає функціонування, тобто постійну зміну та розвиток. Саме у властивостях, зв’язках, функціях та їх поєднанні полягають витоки розвитку інформатичної компетентності як цілісної

системи. Для формування зазначених компетентностей необхідно виконувати репродуктивні, проблемні та евристичні (частково-пошукові) завдання, зокрема індивідуальні і групові проекти, компетентнісні задачі, реалізація яких передбачає використання кількох різних інформаційних середовищ або програмних середовищ.

Таким чином, навчання алгоритмізації та основ програмування сприяє розвитку стійких навиків і умінь самостійно, у новій для себе ситуації, визначати завдання, системно розглядати їх, визначати вхідні і вихідні дані, висувати власні гіпотези, обґрунтовувати їх, пропонувати ефективне розв'язання, перевіряти його на практиці та вміти пояснити отриманий результат. Етапи алгоритмізації та складання програм впливають на процес самоосвіти і загалом сприяють формуванню інформатичних компетентностей на основі розвитку системно-логічного мислення, що відповідно служить головній меті – формуванню фахівця.

Отже, формування та розвиток інформатичних компетентностей є важливим завданням сучасної освітньої поглибленої підготовки учнів з інформатики, та інших дисциплін. Ці компетентності за своєю сутністю є одними з головних ключових компетентностей майбутнього фахівця. Вони передбачають формування початкових навиків роботи з комп'ютерною технікою і програмним забезпеченням, засвоєння основ алгоритмізації та програмування. Система інформатичних компетентностей особистості характеризує її знання, вміння, навички, прагнення, мотиви, інтереси, здатність і готовність до використання ІКТ у навчальній і подальшій професійній діяльності.

Розуміючи психологічні особливості учнів, учитель повинен уміти дієво сприяти формуванню і розвитку цих компетентностей, допомогти школярам розкрити власний творчий потенціал, вибрати індивідуальний освітній шлях. Опираючись на інформатичні компетентності здобуті у школі, студент вишу зможе в подальшому розвивати їх як складову системи професійних компетентностей фахівця, що сприятиме ефективному оволодінню майбутньою професією.

1.3. Дидактична доцільність застосування об’єктно-орієнтованого підходу до навчання алгоритмізації та програмування старшокласників

У цьому розділі обґрунтуємо доцільність використання об’єктно-орієнтованого підходу (ООП) до навчання алгоритмізації та основ програмування в курсі інформатики старшокласників. Згодом цей матеріал зможе стати підґрунтям для розробки положень нового, значно ефективнішого, курсу інформатики, що включатиме елементи ООП.

На сучасному етапі розвитку програмування ООП є одним із його найперспективніших напрямів. До підходу відносять: природну методологію програмування; подання розв’язків у зрозумілих термінах; гармонійне включення розвиненого, дружнього до користувача інтерфейсу; концептуальну єдність і невелику кількість основних конструкцій; абстрактність даних; відкритість, легку розширюваність, універсальність і можливість багаторазового використання створюваних модулів; ООП також дає змогу створювати модульні програми з поданням даних на певному рівні абстракції.

Об’єктно-орієнтована методологія, як і структурна, була створена з метою дисциплінувати процес розробки великих програмних комплексів і тим самим знизити їх складність і вартість. Її використання спрямоване на ті ж цілі, але не відмінну від структурної методології, воно дає змогу управляти складнішими проектами. Особливо треба відзначити, що використання ООП спрощує технологію створення програм, а не саму програму.

Більшість сучасних мов і систем програмування розвивається у напрямку все більшого використання об’єктної методології у створенні програм. Інтуїтивність опису об’єктів у технології ООП дала змогу вченим застосувати його і в системах опрацювання візуальних даних.

Зауважимо, що об’єктний підхід був відомий ще давньогрецьким філософам. Вони розглядали світ як у термінах об’єктів, так і подій. У XVII ст. Рене Декарт **[**119, с. 147**]** зазначав, що люди зазвичай мають об’єктно-

орієнтований погляд на світ. У XX ст. ця тема знайшла своє відображення у філософії об'єктивістської епістемології А. Ренда [282, с. 136]. Модель людського мислення, в якій розум людини розглядається як об'єднання мислячих агентів, запропонував М. Мінскі. Він стверджував, що тільки спільна дія мислячих агентів призводить до осмисленої поведінки людини [25].

Можливість застосування ООП очевидна для завдань найрізноманітнішого характеру. Нині об'єктно-орієнтоване проектування визнане єдиною методологією, яка дає змогу упорядковувати дуже великі й складні системи.

Основною ідеєю ООП є об'єднання в один змістовий блок дій та даних, необхідних для розв'язування поставленої задачі. В об'єктно-орієнтованій моделі відбувається відмова від традиційних уявлень про програму як набір підпрограм і даних. Головними складовими нової моделі є об'єкти, що поділяються на різноманітні класи або типи залежно від характерних для них властивостей та функцій. Інтерфейси об'єктів і зв'язок між ними визначаються на основі притаманних конкретному типові дій і відповідних цим діям повідомлень.

Кожний об'єкт відрізняється від інших об'єктів, хоча й може мати з ними спільні дані або взаємодії. Операції, дозволені над об'єктом або пов'язані з ним, називаються методами. У кожного об'єкта є свій набір методів. У ході виконання програми відбуваються певні події, передбачені описом об'єктів. Обмін даними між об'єктами відбувається з допомогою механізму повідомлень. Методи, які складають конкретне повідомлення, можуть супроводжуватися спеціальними аргументами. Таким чином, посилання повідомлення об'єкту призводить до пересилання одного або кількох методів та використовуваних ними аргументів. Якщо об'єкт не має метода, то повідомлення втрачається. Після опрацювання повідомлення і після того, як об'єкт перестає бути вибраним, він та його стан залишаються стабільними і згодом він знову в змозі отримувати нові повідомлення [188].

Набір об’єктів, притаманних їм методів і обмін повідомленнями між ними, призводить до можливості використання нового підходу в алгоритмізації та програмуванні. Адже об’єкти можуть містити відомості про власну будову і функціонування, обмінюватися повідомленнями з іншими об’єктами.

При конструюванні програм за допомогою системи об’єктів можливості визначення методів набагато багатогранніші, доступніші для розробника та мають значно простішу форму виклику. Механізм використання наслідування істотно спрощує процес розробки нових типів даних у разі проведення необхідних змін алгоритму [272].

За визначенням Р. Буча: ООП – це «методологія програмування, основана на зображенні програми у вигляді сукупності об’єктів, кожен з яких є реалізацією певного класу (типу особливого вигляду), а класи утворюють ієрархію спадковості» [26, с. 152].

Базовими елементами об’єктно-орієнтованої методології програмування є класи і об’єкти. Під об’єктом розуміють елемент, що відображає реальний компонент задачі, якому притаманні чіткі функції та властивості [13; 14; 17; 18]. Об’єкт моделює частину оточуючої дійсності. Термін «об’єкт» в програмному забезпеченні вперше був введений в мові Simula і застосовувався для моделювання реальності. Типи об’єктів, що розглядаються при проектуванні програмних систем, не обмежуються об’єктами реального світу. Виникають інші важливі типи об’єктів, взаємодія між якими служить механізмом відображення взаємодії більш високого рівня. Г. Буч дає наступне визначення: «Об’єкт – це реальність, що володіє станом, поведінкою та ідентичністю; структура і поведінка схожих об’єктів визначає загальний для них клас; терміни «екземпляр класу» і «об’єкт» однозначні» [26, с. 157].

Значення ключових слів даного визначення можна розкрити наступним чином. Стан об’єкта характеризується переліком (як правило, статичним) всіх властивостей цього об’єкта і поточними (як правило, динамічними) значеннями кожної з них.

Об’єкти не існують ізольовано, а піддаються дії або самі впливають на інші об’єкти. Під поведінкою розуміють те, як об’єкт діє і реагує; поведінка виражається в типах стану об’єкта і передачі повідомлень, тобто це діяльність об’єкта, яка зовнішньо спостерігається та перевіряється. Вплив одного об’єкта на інший з метою викликати відповідну реакцію називається операцією або передачею повідомлень між об’єктами. У об’єктно-орієнтованих мовах програмування операції, що виконуються над об’єктом, називаються методами і входять у визначення класу об’єкта.

Під ідентичністю розуміється така властивість об’єкта, яка відрізняє його від всіх інших об’єктів.

Поняття класу і об’єкта тісно позв’язані, неможливо говорити про об’єкт, не відносячи його до класу. Однак існує важлива відмінність цих понять: об’єкт визначає конкретну сутність, реальну в часі і в просторі, а клас визначає тільки абстракцію істотного в об’єкті. Таким чином, клас розглядається як деяка множина об’єктів, яким властиві загальна структура і загальні властивості. Будь-який конкретний об’єкт є екземпляром класу.

Класи не існують ізольовано. У кожній проблемній сфері ключові абстракції взаємодіють багатьма способами, що повинно бути відображено в моделі та програмі. Виокремлюють три основні типи взаємозв’язків між класами [4; 13; 14; 15]: 1) узагальнення – спеціалізація; 2) ціле – частина; 3) семантичні співвідношення.

На етапі аналізу предметної сфери і на ранніх стадіях проектування вирішуються дві основні задачі:

- виявлення класів і об’єктів, складових предметної галузі (ключові абстракції задачі);

- побудова структур, що забезпечують взаємодію об’єктів, при якій виконуються вимоги завдання (механізми реалізації).

Такий підхід створює логічну основу системи (структуру класів і об’єктів), після чого увага переключається на внутрішню поведінку ключових абстракцій і механізмів та на їх фізичне представлення.

Об’єктна модель є концептуальною базою об’єктно-орієнтованого підходу. Багато науковців виокремлють різноманітні елементи та дають їм різні назви. Аналіз вітчизняної літератури показує, що серед українських видань переважають навчальні посібники, орієнтовані на освоєння конкретної мови об’єктно-орієнтованого програмування. У зарубіжній літературі можна виокремити значно ширший блок праць, присвячених ідеології і методології ООП [2; 4; 13; 14; 15; 16; 18]. Такі праці містять глибокий аналіз основних ідей, принципів і понять, які лежать в основі об’єктно-орієнтованої парадигми програмування. На нашу думку, при створенні методики навчання алгоритмізації та програмування на основі ООП необхідно опанувати подібний теоретичний матеріал з метою аналізу змісту навчання із наукових позицій та з точки зору передових досягнень інформатики як науки.

Проаналізувавши думки різних дослідників [2; 4; 12; 13; 14; 15; 16; 18], можна зробити висновок, що об’єктна модель повинна мати наступні елементи:

1. **Абстрагування.** Один з основних методів, що використовується при розв’язанні складних завдань, полягає у виокремленні істотних характеристик деякого об’єкта, які відрізняють його від всіх інших видів об’єктів, і чіткому визначенні його концептуальних границь з точки зору спостерігача.

2. **Інкапсуляція.** Процес відділення один від одного елементів об’єкта, визначаючих його властивості та функції. «Ніяка частина складної системи не повинна залежати від внутрішньої будови якоїсь іншої частини» [26, с. 262]. Абстракція і інкапсуляція доповнюють одне одного: абстрагування спрямоване на зовнішні властивості об’єкта, а інкапсуляція займається його внутрішнім складом. Найчастіше інкапсуляція виконується за допомогою приховування даних, тобто маскуванням всіх внутрішніх елементів, які не впливають на зовнішню поведінку об’єкта. Зазвичай, ховаються внутрішня структура об’єкта і реалізація його методів. Реалізується механізм інкапсуляції наявністю у визначенні класу двох частин: інтерфейсу (відображає зовнішні властивості об’єкта, описуючи абстракцію властивостей всіх об’єктів даного

класу) і реалізації (описує механізми досягнення потрібних властивостей об’єкта).

3. **Модульність.** У мовах програмування, які підтримують модульність (Object Pascal, C++, Ada), модуль – це самостійна мовна конструкція. У цих мовах класи і об’єкти складають логічну структуру системи, вони поміщаються в модулі, утворюючи її фізичну структуру. Під модульністю варто розуміти розбиття програми на фрагменти, які компілюються окремо, але можуть встановлювати зв’язки один з одним. У мовах програмування, що підтримують принцип модульності, інтерфейс модуля відокремлений від його реалізації. Правильний поділ програми на модулі є складним завданням. Для добре відомих (типових) задач цей процес можна стандартизувати, але для нових завдань трансформація на модулі може бути дуже складною.

4. **Ієрархія.** Число абстракцій в складній системі може значно перевищувати розумові можливості людини. Значне спрощення в розумінні складних завдань досягається за рахунок утворення з абстракцій ієрархічної структури. Ієрархія передбачає впорядкування абстракцій, розміщення їх по рівнях. Основними видами ієрархічних структур, які використовують складні системи, є структура класів (ієрархія типу «узагальнення – спеціалізація») і структура об’єктів (ієрархія типу «ціле – частина»).

5. **Типізація.** Поняття типу сформульоване в теорії абстрактних типів даних. Типізація дозволяє запобігти використанню об’єктів одного класу замість іншого або управляти таким використанням. Центральне місце в понятті типізації посідає ідея узгодження типів.

6. **Паралелізм.** Кожна програма має принаймні один потік управління (процес), а може мати і багато таких потоків. Багато сучасних операційних систем (зокрема, Windows, Linux) передбачають підтримку паралелізму, що сприятливо відображається на можливості використання паралелізму в об’єктно-орієнтованих системах. Паралелізм підтримує абстрагування і синхронізацію процесів. Він синхронізує співвідношення активних об’єктів

один з одним. У паралельних системах визначається поведінка об’єкта за наявності декількох незалежних (паралельних) процесів.

7. **Зберігання (стабільність).** Об’єкт здатний існувати в часі, після завершення процесу, який його створив, і (або) переміщатись в просторі зі свого початкового адресного простору. Принцип стабільності дозволяє зберігати не лише дані, а й класи, щоб програми могли правильно інтерпретувати дані. Як правило, цей принцип використовується в об’єктно-орієнтованих базах даних.

Перші чотири з перерахованих елементів є головними, тобто без будь-якого з них модель не вважатиметься об’єктно-орієнтованою. Останні три елементи – додаткові, вони істотно доповнюють об’єктну модель, але не є обов’язковими [2; 4; 15; 16]. Перераховані вище елементи – концептуальна основа об’єктно-орієнтованого програмування. Без неї виразна здатність об’єктно-орієнтованої мови буде втрачена та істотно зменшаться шанси впоратися із складністю розв’язуваних задач.

Отже, об’єктна модель принципово відрізняється від моделей, пов’язаних з більш традиційними методами структурного проектування і програмування. Вона не вимагає відмови від всіх раніше знайдених і випробуваних часом методів і прийомів, але вносить до них нові елементи. ООП підхід забезпечує ряд зручностей, які іншими моделями не передбачалися.

Книга Д. Поя «Математичне відкриття» починається із наступного твердження Р. Лейбніца: «Метод розв’язання дієвий, якщо від початку ми можемо передбачати і далі підтвердити що, слідуючи цьому методу, ми досягнемо цілі» [176, с. 3]. У технології програмування об’єктно-орієнтований підхід у поєднанні із засобами візуального проектування програм – єдиний на сьогодні метод, що задовольняє критерій Р. Лейбніца.

Відзначимо, що використання ООП дає змогу вчителю перейти з рівня простого виконавця чиїхось глобальних задумів – «гвинтика великої системи» – на рівень стратега, який створює необхідні умови для максимальної

реалізації здібностей кожного учня. Ще однією перевагою використання вказаного підходу у навчанні алгоритмізації і основ програмування є здійснення практичної орієнтованості всього курсу інформатики, ООП спрямований на розв'язання найпростіших прикладних завдань планування навчальної діяльності і вносить різноманітність у користувацькі можливості використання ІКТ. Під час навчального процесу на базі ООП формується так званий об'єктний стиль алгоритмічного мислення, якому притаманна чітка послідовність дій і разом із образним та логічним мисленням визначає силу інтелекту старшокласників, їх творчий потенціал.

ООП більш властивий людині, яка ще не надто добре володіє методикою алгоритмізації. Моделювання реальних об'єктів за допомогою готових елементів об'єктно-орієнтованої мови програмування часто є значно природнішим, ніж програмування за допомогою окремих чисел та інших даних, опрацювання яких відбувається якби збоку щодо цих даних. Крім того, подібний підхід виключає ряд помилок, пов'язаних з некоректним використанням операцій у разі їх застосування до даних неіснуючого типу.

Великим досягненням на шляху розвитку методів алгоритмізації став метод поділу програми на незалежні, окремо компільовані і згодом зібрані разом модулі. ООП сприяє природній модульності програм, при якій між різними частинами системи існують зрозумілі та природні зв'язки.

Людина, використовуючи об'єкти певного класу, має доступ тільки до відкритої частини їх опису. Зміна ж внутрішніх властивостей об'єкта є неможливою і це гарантує, що при випадковій або невдалій дії вона не здійснить суттєвої помилки.

Поповнення навчального курсу інформатики значним за обсягом матеріалом з використанням ООП дає змогу не тільки трансформувати зміст навчання, а й впливати на технологію навчання алгоритмізації та програмування (докладніше про це сказано у п. 2.5 дисертації), оскільки він передбачає інший, відмінний від традиційного, цикл розробки програмного забезпечення. Корекція існуючих методик навчання інформатики на основі

ООП для старшокласників спричинює зміну етапів розв’язування навчальних задач в процесі навчання алгоритмізації та основ програмування. Нині у шкільному курсі вивчення інформатики на поглибленому рівні на це виокремлюється дуже мало часу, протягом якого учні мають змогу ознайомитися лише з базовими поняттями об’єктно-орієнтованого програмування (повніше із місцем ООП в сучасній шкільній програмі можна ознайомитися в наступному підрозділі).

Традиційний, нині існуючий підхід передбачає побудову курсу інформатики навколо поняття інформаційної моделі, а весь процес побудови та реалізації алгоритму виглядає як процес математичного моделювання явища, що вивчається. При такому підході виникають значні труднощі у процесі створення дуже складних алгоритмів і програм та їх реалізації, а також вирішенні проблеми підвищення надійності створених програм. В зв’язку з цим стає нагальною потреба в іншій парадигмі алгоритмізації та програмування, яка здатна вирішити весь цей комплекс проблем. І такою парадигмою є об’єктно-орієнтований підхід, який змінив погляд на етапи проектування і реалізації програмних систем, замінивши традиційний цикл створення програмного забезпечення на більш гнучкий поступальний процес. У його основі лежить метод поворотного проектування. У цьому методі головна увага приділяється процесу поступального та ітеративного розвитку шляхом покращення безлічі різноманітних, але, тим не менше, сумісних, логічних і фізичних моделей системи.

Етапи розробки програмного забезпечення за допомогою ООП дещо відрізняються від традиційних. Методи, передбачені схемою цього підходу, не обов’язково виконуються послідовно, а можуть бути зациклениими випадковим чином. Не існує прямої відповідності між етапами розробки програм традиційними методами і при ООП. Циклічність в розробці алгоритмів методами ООП більш продуктивна, оскільки в ході розробки програми результати одного з етапів можуть вплинути на рішення, прийняті раніше.

Отже, метод розробки алгоритму чи програми на основі ООП є значно ширшим від традиційних методів, всі етапи яких він охоплює. Використання цього методу значно підвищує ефективність розв'язання задач у процесі навчання алгоритмізації і основ програмування у класах з поглибленим вивченням інформатики.

Загальновідомо, що досконало оволодіти прийомами алгоритмізації можна, тільки розробляючи програми на практиці. Один із розробників мови програмування Сі Брайєн Керніган говорив, що «єдиний спосіб вивчити мову програмування – писати нею програми» [104]. Тобто для впровадження ООП у навчальний процес необхідна програмна підтримка. Експериментально встановлено, що в навчанні доцільніше застосовувати такі мовні середовища програмування, які задовільнили б, з одного боку вимоги об'єктно-орієнтованих систем, а з іншого – систем, які традиційно використовуються при навчанні алгоритмізації та програмування.

Таким чином, правильний вибір системи програмування, а, отже, і алгоритмічної мови, відіграє чималу роль в ефективності майбутнього курсу інформатики. Об'єктно-орієнтована мова є мовою високого рівня для об'єктно-орієнтованих систем програмування. У такій системі поняття підпрограм і даних, які використовуються в звичайних системах програмування, замінені поняттями «об'єкт» і «повідомлення». На відміну від процедури, що описує, як повинно виконуватися опрацювання, повідомлення визначає тільки те, що бажає виконати відправник, а отримувач точно визначає, що повинно відбуватися.

Зважаючи на необхідність розробки власної методики навчання алгоритмізації та програмування на основі ООП, доцільно обґрунтувати важливу роль вибору мови програмування з метою її використання в навчальному процесі. Ця проблема залишається методично складною і все ще дискусійною. Якщо основи алгоритмізації і програмування посідають надзвичайно важливе місце в курсі інформатики, то вирішення цієї проблеми має принципове значення, «оскільки від цього багато в чому залежить

методика вивчення курсу, зміст і послідовність подання навчального матеріалу, система навчально-пізнавальних завдань, а головне, вся подальша робота з оволодіння програмуванням для розв'язання реальних практичних завдань», – вважає Б.С. Гершунський [36, с. 28].

При вирішенні цієї проблеми повинні враховуватися не тільки широта можливостей використання мови програмування, охоплення нею основних прийомів, а й певні психолого-дидактичні характеристики, серед яких варто виокремити доступність мови для засвоєння та її природність. Мова програмування, яку використовують у процесі навчання старшокласників, повинна задовольняти ряд вимог, що насамперед враховують наступну обставину: мета діяльності учня – «розв'язування за допомогою комп'ютера певної задачі, а програмування – лише один з етапів її розв'язання» [151, с. 35]. У зв'язку з цим мова програмування повинна мати дуже розвинені засоби реалізації комунікацій «людина – комп'ютер», вона мусить бути зручною для аналізу й опису умов задачі, планування розв'язування, складання програми з контролю правильності розв'язку в цілому і окремих його етапів. Це означає, що при розв'язуванні завдання учень повинен мати можливість зосередитися на його умовах, а не на подоланні обмежень використовуваної ним мови програмування [150].

Трактування програмування як спілкування мовою, «зрозумілою» комп'ютеру і його користувачеві, передбачає створення і використання такої мови, за якої оволодіння обміном повідомленнями з комп'ютером стало б настільки ж природним процесом, як засвоєння рідної мови. Це означає, по-перше, доступність мови; по-друге, надання учневі можливості розширення мови за допомогою складання власних програм; по-третє, «спілкування» з комп'ютером обраною мовою має створювати умови, що сприяють формуванню в учня здатності до рефлексії: кожного разу він бачить, як комп'ютер «реагує» на його команду, порівнює цю реакцію з очікуваною і, якщо вони відрізняються, намагається знайти помилку в ході власних міркувань.

У навчальному середовищі, в якому відбувається навчання школярів програмування, повинна бути можливість обмінюватися повідомленнями. Варто мати на увазі, що із використанням комп'ютера моделюється не просто спілкування, а педагогічне спілкування, при якому, як підкреслював О. О. Леонтьєв [134, с. 23], повинні створюватися найкращі умови для розвитку мотивації учнів і творчого характеру навчальної діяльності, для формування особистості школяра має забезпечуватися сприятливий емоційний клімат навчання.

Таким чином, до мов програмування, що вивчаються у курсі інформатики, повинні ставитися наступні вимоги:

– наявність розвинених засобів реалізації комунікації «людина – комп'ютер»;

– зручність і природність опису умов задачі та її розв'язування;

– можливість розширення мови самим учнем;

– забезпечення основи для подальшого вивчення інших сучасних мов програмування, розвитку ефективних методів програмування і розв'язання завдань;

– близькість концепції мови до образу мислення учнів.

Більшість сучасних мов і систем програмування розвивається в напрямку все більшого використання об'єктної методології у створенні програм. Розробка, заснована на компонентах (це заздалегідь створені об'єкти, які за необхідності можна вставляти в розроблювану програму), стала наступним еволюційним кроком у розвитку методів програмування.

Останнім часом термін «об'єктно-орієнтований підхід» застосовується не тільки стосовно програмування, а й в сфері ІКТ. Доцільність використання ООП в прикладній інформатиці, тобто при освоєнні роботи з основними засобами ІКТ, розглядає Б.Е. Стариченко [235]. Автор обґрунтував використання основних понять об'єктно-орієнтованого програмування (об'єкт, клас об'єктів, властивість об'єкта, подія тощо) як при роботі з прикладними програмними системами (наприклад, інформаційні об'єкти), так

і при розгляді сучасних систем програмування (створювана програмістом програма будується з програмних об’єктів).

Такий підхід дає змогу розглядати програмування як складову інформаційних комп’ютерних технологій. Цим забезпечується ідейна, логічна та термінологічна єдність підходів до освоєння будь-яких ІКТ на об’єктній основі. «Об’єктний підхід об’єднує і програмістську гілку прикладної інформатики, і призначену для користувача. Єдність підходів до побудови документа і програми надзвичайно важлива з ідейної точки зору, оскільки демонструє спільність і універсальність методів інформатики, з одного боку, і помітно полегшує опанування комп’ютерних технологій, з іншого боку», – стверджує Б.Е. Стариченко [235, с. 85].

Враховуючи сучасні тенденції розвитку інформатики, одним з провідних і найбільш перспективним напрямом, на думку О.Г. Андросова, Д.С. Іванової, Г.Ю. Хачевої та інших науковців, є об’єктно-орієнтоване програмування [6; 87; 5, с. 5]. У роботах цих авторів відзначаються переваги такого програмування: можливість реалізації проектів з візуалізації, моделювання, імітації різних процесів, об’єктів, явищ, а також з продукування інформаційного ресурсу.

ООП сприяє суттєвому підвищенню сприйняття та оволодіння учнями сучасних програмних продуктів, що дає їм змогу користуватися новими можливостями для розв’язування найрізноманітніших задач. Зокрема, сьогодні значного поширення набуло використання СКМ при розв’язуванні прикладних математичних задач. Тому у нашому дослідженні використання ООП в навчанні алгоритмізації та основ програмування старшокласників з поглибленим вивченням інформатики розглядається на прикладі застосування програми СКМ Maple, яка користується великою популярністю завдяки своїй доступності, функціональності, потужній програмній підтримці та постійному вдосконаленню.

Програмні СКМ спочатку ділились на два принципово відмінні класи: системи для чисельних та символьних (аналітичних) обчислень. До перших

відносили системи Eurica, MATLAB, MathCAD, електронні таблиці, зокрема Microsoft Excel. Другий клас утворювали такі системи комп'ютерної алгебри, як Derive, MuPAD, Maple. На сучасному етапі розвитку ІКТ такий поділ вважається застарілим. Усі перераховані вище системи отримали свій розвиток як універсальні математичні системи, які забезпечують автоматизацію і чисельних, і символьних обчислень. Системи комп'ютерної математики дають змогу уникнути процесу громіздких формульних обчислень і отримати кінцевий результат в числовій, формульній і графічній формах, що звільняє користувача від непродуктивних затрат часу. Новітні СКМ мають достатньо потужний арсенал засобів для розв'язування задач різноманітних класів, оснащені великою кількістю вбудованих функцій, засобами символьних перетворень, візуалізації та анімації.

Важливою вимогою для математичного середовища є можливість інтегрувати його з іншими системами комп'ютерної математики, а це спричинено необхідністю:

- раціонального вибору системи комп'ютерної математики з урахуванням особливостей задачі, що розв'язується;
- розв'язування складних задач за допомогою різних засобів для перевірки правильності результатів, не покладаючись на одну СКМ (збільшення вірогідності одержаного результату);
- підготовки математичних документів навчального призначення [223].

На думку Ю.В. Триуса, сьогодні необхідна інтеграція математичних систем між собою та з іншими програмами, що може розглядатись як один із перспективних напрямів розвитку систем комп'ютерної математики [246, с. 364-365].

У світі одне з чільних місць посідає система Maple, яка й була від початку орієнтована на застосування її в сфері освіти. На думку М.І. Жалдака, сучасний розвиток програмного забезпечення комп'ютерів досяг такого рівня, коли в багатьох випадках алгоритм досягнення мети може бути побудований

автоматично. Вказівки комп'ютерові потрібно задавати в термінах очікуваних результатів, а не в описах процесів, що призводять до таких результатів [68].

Нині Maple – це потужна та універсальна програма, призначена для розв'язування широкого спектра задач і має потужні засоби опрацювання графічних даних. Вона є спільною розробкою Університету Ватерлоо (штат Онтаріо, Канада) і Вищої технічної школи (м. Цюріх, Швейцарія), для продажу якої була створена спеціальна компанія – Waterloo Maple, Inc. Порівняно з іншими пакетами Maple виграє своїми графічними та анімаційними послугами. Оскільки для шкільної освіти важливим критерієм є наочність, а найкраще його задовільняє система Maple, то саме цю програму варто застосовувати в школі на уроках математики, інформатики та фізики [291].

Крім того, використання СКМ Maple дає змогу здійснювати імпорт та експорт даних. Так, наприклад, файл можна експортувати в загальновідомі HTML, LaTeX, RTF форматах. Зберігається форматування документа, наявність формул, що є надзвичайно зручним. Також широко розвинена можливість імпорту об'єктів в документ Maple – таких, як графічне зображення, звук, відео, формула. Імпорт та експорт файлів успішно здійснюється між Maple та такими програмами: Corel Draw, Microsoft Word, Microsoft Power Point, Paintbrush. У зв'язку з широким впровадженням інтернет-технологій в Maple існує команда, виконання якої відкриває ресурс інтернету або файл Maple, який знаходиться на іншому комп'ютері в мережі, не виходячи з програми аналітичних обчислень [291].

За допомогою Maple можна розв'язувати велику кількість математичних задач шляхом введення команд без будь-якого попереднього програмування. Maple оперує цілими, раціональними числами та їх десятковими наближеннями. Це дає змогу отримати вираз результатів як в радикалах, так і їх наближення з потрібною точністю [49].

Також важливі ергономічні вимоги, які задовільняє Maple, з точки зору використання їх в системі освіти: відображення даних на екрані вибирається користувачем (колір, шрифт, масштаб, редагування графіки тощо);

забезпечується робота в кількох режимах (текстовий, графічний, символьний); до пакета можна підключати бібліотеки з метою розв'язування додаткового ряду задач.

Комп'ютерний математичний пакет Maple з точки зору педагогіки є дидактичним засобом навчання, який за наявності відповідної розробленої методики навчання дає змогу підвищити ефективність навчального процесу, а з точки зору інформатики є засобом, призначеним для автоматизації розв'язування математичних задач в різноманітних галузях науки, техніки та освіти, інтегруючим в собі сучасний інтерфейс користувача, аналітичні та чисельні методи розв'язування різних математичних задач, засоби візуалізації результатів обчислень. На стадії прийняття рішень такий засіб дає змогу з більшою вірогідністю проаналізувати одержані результати.

Тому, враховуючи вищесказане, будемо вважати СКМ Maple найбільш доцільною для ознайомлення учнів із основами ООП на уроках поглибленого вивчення інформатики. На нашу думку, вона сприятиме ефективному застосуванню цього підходу у подальшому навчанні.

Застосування СКМ в системі освіти сприяє реалізації ряду дидактичних принципів навчання:

– з їх використанням в процесі навчання реалізується творча активність та ініціатива учнів, отже, відбувається зміщення акценту в сторону активного навчання;

– використання СКМ (Maple в тому числі) гармонійно об'єднує групову та індивідуальну форми навчання при виконанні кожним учнем індивідуального завдання як частини загального завдання, отже, реалізується принцип колективного характеру в поєднанні з розвитком індивідуальних особливостей особистості;

– ілюстративність і практичне значення матеріалу, поданого за допомогою математичних пакетів, сприяє активізації навчання і формуванню стійкого пізнавального інтересу;

– із використанням СКМ реалізується принцип професійної спрямованості навчання, який виражається у формуванні в школярів професійно значущих умінь і навичок для своєї майбутньої роботи в сфері прикладної математики;

– застосування Maple дає змогу реалізації принципу наочності навчання, оскільки за допомогою систем комп'ютерної математики стає можливим наочно показувати учням більшу кількість фундаментальних наукових досягнень в галузі природознавства, сформувати знання про загальнонаукові методи пізнання, методи дослідження і комп'ютерне моделювання.

Таким чином, з'ясовано та обґрунтовано навчальну доцільність використання об'єктно-орієнтованого підходу до навчання алгоритмізації та програмування старшокласників в процесі поглибленого вивчення інформатики на прикладі застосування СКМ Maple.

З вищесказаного можна зробити такі висновки:

– об'єктно-орієнтований підхід ґрунтується на побудові об'єктної моделі системи, яка включає в себе об'єктно-орієнтовану декомпозицію, уявлення змін внутрішнього стану об'єктів та відображення взаємодії об'єктів;

– використання ООП зумовлено як в роботі з прикладними програмними системами, так і в застосуванні сучасних систем програмування, що дає змогу розглядати програмування як одну з ІКТ, демонструючи загальність та універсальність методів інформатики і полегшуючи навчання алгоритмізації та програмування на основі ООП;

– ООП на сучасному етапі є першочерговим в галузі розробки та використання програмних засобів;

– вивчення універсальних методів і прийомів роботи з об'єктно-орієнтованим програмним забезпеченням дає змогу школярам в подальшому самостійно засвоювати та використовувати засоби ІКТ для розв'язування поставлених задач;

– застосування Maple є доцільним в процесі вивчення старшокласниками алгоритмізації та програмування, оскільки цей комп'ютерний математичний пакет забезпечує їм зручне інтелектуальне середовище для математичних досліджень.

1.4. Аналіз методичних систем навчання алгоритмізації та програмування у старшій школі та шляхи їх вдосконалення

Аналізуючи стан навчання алгоритмізації та програмування в шкільному курсі інформатики, можна зробити висновок, що його методологічною основою є структурний підхід. У даній роботі пропонується будувати навчання інформатики із застосуванням ООП. Нині об'єктно-орієнтоване проектування широко застосовується при створенні програмних продуктів, питання ООП включені в зміст освіти з інформатики для класів з поглибленим вивченням. Разом з тим методика навчання об'єктно-орієнтованого програмування в загальноосвітній школі потребує вдосконалення.

Для підтвердження висловленого твердження проведемо аналіз найпоширеніших методик навчання інформатики та шкільних програм і підручників, що використовують при навчання інформатики в школі.

У 1985 році в змісті шкільної освіти з'явився новий предмет «Основи інформатики і обчислювальної техніки». Спочатку цей курс був орієнтований на вивчення основ програмування, а згодом – на ознайомлення та використання засобів інформаційних технологій, існуючих тоді. Швидкий розвиток комп'ютерної техніки і програмного забезпечення, який почався буквально після введення цього курсу, призвів до переосмислення цілей та місця цього предмета у системі освіти. В останнє десятиліття інформатика як фундаментальна наука стає ключовою складовою всієї системи наукового пізнання і суттєво визначатиме шлях формування глобального інформаційного суспільства, заснованого на знаннях.

Особливості навчання школярів основам інформатики детально розглядалися у методичних працях та розробках Н.В. Апатової [7; 122], С.А. Бешенкова [63], Я.А. Ваграменко [27], Є.Ф. Вінниченко [68], Ю.В. Горошка [68], В.Б. Дем'яненко [51, 52], А.П. Єршова [63], М.І. Жалдака [64; 65; 67; 68; 69; 70; 72; 73; 74; 75], О.А. Кузнєцова [123], Н.В. Морзе [93; 155; 156; 157], Ю.С. Рамського [192; 193; 199; 200; 201; 194; 195; 196], І.В. Роберт [203; 204], З.С. Сейдаметової [212], Н.Д. Угринович [249; 250], Є.К. Хеннера [132; 214] та інших вчених. Їх дослідження відображали методичну систему вивчення окремих тем інформатики, яка складалася із основних взаємопов'язаних компонентів: цілей, змісту, засобів, організаційних форм і засобів навчання. Ці дослідження відрізняла спрямованість на актуальні в той час вимоги до знань, умінь і навичок учнів, завершеність відповідного навчально-методичного забезпечення.

Говорячи про розвиток інформатики як дисципліни, не можна не відзначити педагогічну діяльність колективу вчених під керівництвом А.П. Єршова. В її ході було вироблено багато теоретичних та методичних положень, які згодом були використані при створенні змісту предметів, пов'язаних з інформатикою на різних рівнях освіти. Вивчення пристроїв ЕОМ та основ програмування в навчальних закладах привели до появи нового етапу: впровадження поглибленого вивчення питань інформатики, в тому числі пов'язаних з обчислювальною технікою і програмуванням. Паралельно з цим виникла низка питань педагогічного характеру. Треба було зробити вибір у змісті та методиці навчання інформатики. Були досліджені деякі загальноосвітні аспекти навчання алгоритмізації та програмування й питання взаємозв'язку навчання програмування зі змістом математичних дисциплін: таких, як алгебра, логіка, геометрія (напрямки досліджень А.П. Єршова [63], М.П. Лапчіка [132], В.М. Монахова [63], І.Г. Семакіна [213] та ін.).

За останні роки методична система навчання інформатики пройшла довгий шлях розвитку, і він тісно пов'язаний з розвитком власне інформатики. Зокрема, застарілість комп'ютерної техніки у школах, її невпорядкована

різноманітність, вкрай обмежені можливості використання, а часто повна відсутність комп'ютерів стали причиною того, що у попередні роки основою навчання інформатики стало вивчення алгоритмізації (акцент робився на вміння будувати порівняно прості алгоритми) та мов програмування. Такий підхід, зазначимо, мав свої переваги, незважаючи на вимушену звуженість предмета вивчення.

Використання комп'ютера в учнів пов'язувалося в основному з програмуванням, а тому вони значно більш масово, ніж зараз, вивчали мови програмування, створювали досить складні для їхнього віку програми. Найбільш розповсюдженою у той час мовою програмування була мова BASIC, що мала широкі графічні функції. Крім того вона була тісно інтегрована в апаратне забезпечення. Водночас намітилася позитивна тенденція переходу від BASIC до краще обумовленої методично мови програмування PASCAL, однак остання поступалася меншими графічними засобами.

Поштовхом для розвитку інтересу учнів до програмування стала поява численних аматорських графічних бібліотек до неї. Програмне забезпечення та наявний комп'ютерний парк визначали, які навчальні програми з інформатики будуть використовуватися в тій чи іншій школі. Широко використовувався і так званий безмашинний варіант навчання інформатики. Саме в цей час сформувалася перша вітчизняна школа навчання інформатики. До неї увійшли М.І. Жалдак, Ю.С. Рамський, Н.В. Морзе та ін., З'явилися теоретичні розробки з методики навчання інформатики: навчальні посібники «Вивчення мов програмування у школі» [71] та «Інформатика» [70].

Порівняно зі станом навчання інформатики в 1985 році – офіційною точкою відліку впровадження інформатики в освіту, коли інформатика вивчалася лише в старших класах загальноосвітніх закладів – сьогодні вивчення інформатики відіграє провідну роль не тільки в змісті шкільної освіти, а й у подальшому навчанні. Сучасний курс інформатики є результатом великого спектра досліджень, відображених в роботах С. А. Бешенкова [63], В. Ю. Бикова [19], А.Ф. Верланя [7], Ю.В. Горошка [39; 41; 68],

А. Г. Гейна [34; 33], А. П. Єршова [62], М. І. Жалдака [64; 65; 73; 74; 70; 69; 75; 68; 67; 72], В.І. Клочка [109; 111], О. А. Кузнєцова [121], А. Г. Кушніренка [128; 129], М. П. Лапчика [131], В.В. Лапінського [72; 205], Ю.І. Машбиця [151], Н. В. Морзе [155; 156; 157], В.М. Монахова [63], Ю. С. Рамського [192; 193; 195; 196; 199; 200; 201; 194], С.А. Ракова [189], В. Д. Руденко [206], З. С. Сейдаметової [212], О. М. Спіріна [234], Ю.В. Триуса [223; 246; 245], Є. К. Хеннера [132; 214], С. М. Яшанова [269] та багатьох інших.

Однією з основних цілей курсу інформатики є навчання учнів розв'язуванню задач, збирання, опрацювання, передавання, збереження та захисту даних, значущих з точки зору подальшої діяльності. В межах цього дослідження подібні завдання узагальнено розглядаються як завдання з опрацювання даних. Однак для розв'язування складних сучасних задач з опрацювання даних не завжди вдається знайти необхідне програмне забезпечення. У зв'язку з цим розділи, присвячені вивченню програмування, є важливою та невід'ємною складовою існуючих курсів інформатики. Як правило, вивчення програмування переслідує дві основні цілі: формування алгоритмічного стилю мислення та навичок розв'язування конкретних задач з опрацювання даних.

Незважаючи на те, що питання навчання алгоритмізації і програмування докладно вивчені в роботах багатьох вчених, постійний розвиток ІКТ вимагає вдосконалення існуючих методичних систем навчання відповідних розділів курсу інформатики. Дійсно, потужні сучасні комп'ютери стають все більш дешевими і масовими засобами опрацювання даних. Змінюються пріоритети в програмуванні. Завдання оптимізації використовуваних ресурсів комп'ютера (часу розрахунків, витрат пам'яті і т. д.) стає менш актуальною, ніж оптимізація трудовитрат фахівця, який створює нові програмні засоби. Від створюваних програм найчастіше потрібно лише коректне розв'язування поставлених завдань. Водночас, програмісти повинні відповідати все більш високим професійним вимогам. У зв'язку з цим необхідна зміна пріоритетів, на які повинна орієнтуватися методична система навчання інформатики.

Якщо раніше в системі підготовки програмістів пріоритетне місце посідали розділи курсу інформатики, присвячені створенню ефективних алгоритмів і оцінки ефективності розроблюваних програм, то зараз на перше місце виходять розділи, пов'язані з вибором технологій, що призводять до мінімізації трудовитрат і забезпечують коректність розв'язку поставлених завдань з опрацювання даних.

Нині наступає новий період розвитку інформатики як міждисциплінарного наукового напряму, що буде виконувати інтеграційні функції для інших напрямів науки – як природничих, так і гуманітарних. Проникнення ідей і засобів інформатики в ці галузі диктується потребами і логікою розвитку фундаментальної науки, а також необхідністю розв'язання ряду важливих прикладних проблем. Варто очікувати, що це не тільки дасть новий імпульс для розвитку наукових досліджень на стику інформатики з іншими науками, а збагатить й інформатику новими перспективними ідеями.

Однак ці дослідження торкаються лише окремих компонентів системи підготовки з основ інформатики в межах загальної середньої освіти, а також підготовки вчителів до використання ІТ. Крім того, у них не повністю віддзеркалюється зміст предметної галузі «Інформатика», який змінився протягом останніх років, і соціальний контекст розвитку освіти в Україні наприкінці XX – на початку XXI ст. Розвиток засобів інформатизації, ІКТ зумовлює суттєві зміни в інформатиці як навчальній дисципліні, що вимагає переосмислення цілей, змісту, засобів і форм підготовки учнів і вчителів з інформатики на сучасному рівні. Ці зміни повинні знайти відображення як у системі загальної освіти, так і у підготовці педагогічних кадрів.

Головною причиною постійного вдосконалення методики навчання інформатики є розвиток науки інформатики, ІТ та засобів зв'язку. Перехід середніх навчальних закладів до профільної освіти та вишів до ступеневої освіти є ще однією умовою, яка повинна суттєво вплинути на методику навчання інформатики, розробку нових технологій і систем навчання та удосконалення наявних, традиційних. Наявність такої умови вимагає

перегляду всіх методологічних і концептуальних основ традиційної педагогіки та переходу до неперервної відкритої освіти, заснованої на особистісно-орієнтованому навчанні [158, с. 5].

Проблема нових підходів до вивчення алгоритмізації та основ програмування відображена в багатьох дослідженнях, однак більша їх частина спрямована на дорослішу аудиторію – студентів, а шкільному процесу навчання приділяється дещо менша увага. Наприклад, ці питання досліджувала Л. В. Гришко, яка відзначила, що при вивченні основ програмування повинні формуватися:

- уявлення про наявні парадигми програмування;
- уміння аналізувати алгоритми і уміння будувати ефективні алгоритми;
- навички практичного програмування однією або двома мовами високого рівня;
- уміння і навички колективного проектування програмного забезпечення;
- навички самостійної роботи з навчальною літературою при розв’язуванні завдань з програмування [45].

Сукупність традиційних і інноваційних напрямів впровадження ІКТ в освітній процес призвела до утвердження інтегративної концепції використання цих технологій в шкільній та університетській освіті, що розглядає ІКТ насамперед як засіб розвитку особистості учня (студента), а також як засіб, що сприяє переведенню його в режим саморозвитку, що перетворює учня з об’єкта педагогічного впливу в повноправного суб’єкта освітнього процесу й сприяє актуалізації його управлінської діяльності як активного учасника інформаційного процесу всередині освітньої системи [193, с. 10-12.].

У шкільних курсах інформатики пропонуються переважно традиційні процедурно-орієнтовані мови програмування. Д. А. Грамаков радикально заявляє, що «не можна викладати те, що є позавчорашнім днем в сфері

інформаційних технологій». Процедурно-орієнтована парадигма повинна бути замінена об’єктно-орієнтованою моделлю, яка на сьогоднішній день є найбільш ефективною моделлю, що використовується в індустрії програмування [43, с. 27].

Поява навчальних закладів нового типу та перехід загальноосвітніх навчальних закладів на новий зміст і структуру спричинила поглиблення змісту основного курсу інформатики та посилення його прикладної спрямованості. Інформатика нині є одним із засобів формування не тільки освітнього, а й розвиваючого та інтелектуального потенціалу особистості.

У процесі поглибленого вивчення інформатики основні завдання курсу суттєво розширюються та доповнюються, що обумовлено необхідністю виявлення та розвитку в учнів логічних здібностей, підготовки їх до участі в олімпіадних змаганнях та наукових дискусіях, формування в них стійкого інтересу до інформатики і пов’язаної з нею професійної діяльності, підготовки до навчання у вищих навчальних закладах.

У 10-11 класах з поглибленим вивченням математики та інформатики курс інформатики може вивчатись за програмами:

− Програми для спеціалізованих шкіл, гімназій, ліцеїв. Інформатика і програмування. 8-11 класи» з розрахунку 4 години на тиждень;

− Програми для загальноосвітніх навчальних закладів, спеціалізованих шкіл, гімназій, ліцеїв. Інформатика (поглиблений курс). 8-11 класи з розрахунку 2 години на тиждень у 9 класах і 5 годин на тиждень у 10-11 класах [181].

Розглянемо детальніше навчальну програму поглибленого вивчення інформатики для учнів 10-11 класів загальноосвітніх навчальних закладів (розроблену на основі Закону України «Про загальну середню освіту» від 27.08.2010 р. № 834) та проаналізуємо обсяг, який надається вивченню основ ООП та його застосуванню в роботі учнів.

Характерною особливістю структури цієї навчальної програми є те, що вона складається з двох паралельних змістовних ліній: сучасних ІКТ і алгоритмізації та програмування. Саме у другій змістовій лінії і І семестрі

11 класу і відведено 48 годин на вивчення основ об'єктно-орієнтованого програмування (базовий курс, основи мови програмування та моделювання мовою програмування).

За цей навчальний період учні повинні ознайомитися із базовими алгоритмічними структурами (поняття про візуальне конструювання програм, можливості створення програм під різні платформи, складові середовища, етапи розробки проекту; етапи проектування форми, поняття про об'єкт, властивості та події, арифметичні операції, стандартні функції, команди присвоювання, введення та виведення даних, оператори присвоювання, компоненти для введення та виведення даних, типи даних – 20 годин), основами ООП (клас**,** об'єкт, метод, принципи ООП – 4 години), графікою у мові об'єктно-орієнтованого програмування (властивості, необхідні для створення графічних зображень, методи, необхідні для креслення графічних примітивів – 8 годин), мультимедійними можливостями об'єктно-орієнтованої мови програмування (компоненти для відтворення анімації, відео файлів, звуку, програми для створення довідкової системи – 4 години), елементами комп'ютерного моделювання (поняття моделі, моделювання, особливості чисельного моделювання фізичних задач – 10 годин), базами даних в об'єктно-орієнтованої мові програмування (поняття баз даних, типи баз даних, структуру баз даних – 14 годин).

Позитивним моментом шкільної програми є стимулювання самостійної роботи учнів шляхом виконання власних проектів протягом вивчення всіх розділів і тем курсу. Зокрема, в 11-му класі передбачено виконання учнями проектних робіт, яким передує знайомство з об'єктно-орієнтованими середовищами програмування. Це сприяє розвитку їх креативного мислення під час опанування курсу поглибленого вивчення інформатики. Так, у програмі ми бачимо наскрізний зв'язок тем курсу з іншими предметами шкільного компонента через виконання практичних, лабораторних робіт, розробки власних проектів. А тому вважаємо доцільним використання

СКМ Maple при вивченні цієї теми як ефективного прикладу застосування об’єктно-орієнтованого підходу до вивчення інформатики.

Як бачимо, на вивчення основ ООП – таких, як клас, об’єкт та головні принципи цього підходу, дається лише 4 години, що, на нашу думку, недостатньо для формування в учнів умінь і навичок активно користуватися ООП у подальшій взаємодії з комп’ютерними програмами.

Реалізація профільного навчання інформатики у 10-11 класах забезпечується системою курсів за вибором (за рахунок варіативного компонента), які враховують інтереси і можливості учнів даного профілю. Курси за вибором, спецкурси та факультативи поглиблюють та розширюють основний курс інформатики відповідно до профілю навчання, надають можливості для організації творчої роботи учнів через систему індивідуальних завдань професійної спрямованості. Одним із таких існуючих курсів є курс за вибором «Основи алгоритмізації та програмування», що вивчається із розрахунку 2 години на тиждень (всього 140 годин) в класах з поглибленим вивченням математики та інформатики. Авторами програми цього курсу за вибором є Т. П. Караванова і В. П. Костюков [180]. Головна мета цього курсу – формування в учнів знань, умінь і навичок об’єктно-орієнтованого візуального програмування, про основні правила і методи складання, редагування та виконання програм в середовищах візуального програмування.

Значним удосконаленням навчального процесу стало вивчення інформатики в середніх класах ЗОШ, що призведе до стрімкого зростання ступеня сформованості інформатичних компетентностей старшокласників не тільки профільного рівня, а й базового. Варто відзначити велику кількість нових видань для шкільного навчання інформатики, в яких глибше розкрита тема алгоритмізації та програмування. До таких можна віднести:

а) рівень стандарту:

− Інформатика: підручник для 5 класу загальноосвітніх навчальних закладів / Н. В. Морзе, О. В. Барна, В. П. Вембер, О. Г. Кузьмінська, Н. А. Саражинська. – К. : Видавничий дім «Освіта», 2013. – 256 с.;

− Інформатика. 5 клас / Ривкінд Й. Я., Лисенко Т. І., Чернікова Л. А., Шакотько В. В. – К. : «Генеза», 2013. – 200 с.;

− Інформатика: підручник для 6 класу загальноосвітніх навчальних закладів / Н. В. Морзе, О. В. Барна, В. П. Вембер, О. Г. Кузьмінська, Н. А. Саражинська. — К. : Видавничий дім «Освіта», 2014. — 240 с.;

− Інформатика. 6 клас / Ривкінд Й. Я., Лисенко Т. І., Чернікова Л. А., Шакотько В. В. – К. : «Генеза», 2014. – 256 с.;

− Інформатика. 9 клас / Ривкінд Й. Я., Лисенко Т. І., Чернікова Л. А., Шакотько В. В.; за заг. ред. М. З. Згуровського. – К. : «Генеза», 2009. – 296 с.;

− Інформатика : 9 клас : підручник для загальноосвітніх навчальних закладів / Завадський І.О., Стеценко І.В., Левченко О.М. – К. : Видавнича група BHV , 2009. – 320 с.

− Інформатика : Підручник для 9 класу загальноосвітніх навчальних закладів / Володін В.В., Володіна І.Л. – Х. : Видавництво «Гімназія», 2009. – 384 с.;

− Інформатика: підручик для 10 класу загальноосвітніх навчальних закладів : рівень стандарт / Н. В. Морзе, В. П. Вембер, О. Г. Кузьмінська, Н. А. Саражинська. – К. : Школяр, 2010. – 304 с.

− Інформатика. 10 клас / Ривкінд Й. Я., Лисенко Т. І., Чернікова Л. А., Шакотько В. В.; за заг. ред. М. З. Згуровського. – К. : Генеза, 2010. – 296 с. : іл.;

− Інформатика : підручник для 11 класу загальноосвітніх навчальних закладів : рівень стандарту/ Н. В. Морзе, О. В. Барна, В. П. Вембер, О. Г. Кузьмінська. – К. : Школяр, 2011. – 304 с.

− Інформатика. 11 клас : рівень стандарту / Ривкінд Й. Я., Лисенко Т. І., Чернікова Л. А., Шакотько В. В. ; за заг. ред. М. З. Згуровського. – К. : Генеза, 2011. – 304 с.

б) академічний та профільний рівень:

− Інформатика. 10 клас : академічний рівень, профільний рівень / Ривкінд Й. Я., Лисенко Т. І., Чернікова Л. А., Шакотько В. В. ; за заг. ред. М. З. Згуровського. – К. : Генеза, 2010. – 304 с.

− Практикум і робочий зошит з інформатики. 10 клас. Академічний рівень / Завадський І.О., Пасічник О.В., Бойчук В.В. – К. : Вид. група BHV. 2010. – 190 с.

− Інформатика. 11 клас : академічний рівень, профільний рівень / Ривкінд Й. Я., Лисенко Т. І., Чернікова Л. А., Шакотько В. В. ; за заг. ред. М. З. Згуровського. – К. : Генеза, 2011. – 304 с.

Як вважає Н.О. Бугаєць, серед сучасних ІТ особливе місце відводиться технології програмування. Візуальне програмування – один з сучасних напрямів програмування. В його основі лежить об’єктно-орієнтований підхід до описання процесів (явищ), який є одним з найбільш ефективних та зручних і використовується сьогодні програмістами для створення великих програмних систем [24, с. 68].

Таким чином, з розвитком нових ІКТ, заснованих на принципах ООП, стає актуальним питання вивчення об’єктно-орієнтованого програмування. Як вказують С.А. Жилін та І.Б. Жиліна, це особливо важливо для класів з поглибленим вивченням інформатики та математики [76, с. 44-50]. На думку М. П. Лапчика [131], І. Г. Семакіна [213], Є. К. Хеннера [132], у ході його вивчення вирішуються такі завдання: засвоєння методології об’єктно-орієнтованого програмування; оволодіння технікою об’єктно-орієнтованого програмування однією з мов; уведення учнів у проблематику, адекватну до даного підходу; розширення загального кругозору (загальноосвітній компонент). Разом з тим вказані науковці вважають, що методика навчання в школі будь-яких видів об’єктно-орієнтованого програмування розроблена «абсолютно недостатньо», того цей процес перебуває на початковій стадії. Авторам невідомо жодного посібника з об’єктно-орієнтованого програмування, який би повністю задовільнив вимоги методики навчання цього виду програмування в школі [132, с. 14].

Одним з перших А.С. Лесневський на початку 1990-х років запропонував вивчати в школі елементи об’єктно-орієнтованого програмування в курсі інформатики VII класу на базі мови Смолток [135, с. 41-44.]. Його точка зору

аргументувалась тим, що інформатика – це фундаментальна наука, отже, її основи повинні вивчатися в школі. Як формальну мову він пропонував використовувати об'єктно-орієнтовану систему програмування Смолток. А. Лесневський звернув увагу на те, що мови (системи) програмування не є основним предметом вивчення інформатики, але підтримують певне коло її понять.

Пізніше ці ідеї знайшли своє віддзеркалення в запропонованій А. Горячевим спільно з А. Лесневським програмі курсу інформатики для I-IX класів [42, с. 12-17.]. У курсі передбачено вивчення наступних розділів: статична картина об'єкта, картина поведінки об'єкта, мова як об'єкт моделювання, інформаційна модель об'єкта впродовж всього курсу концентрично, тобто обсяг понять зростає від класу до класу. Дана програма і результати її апробації, на думку авторів, підтвердили можливість вивчення принципів, засобів і прийомів формалізації, що застосовуються в ООП, не лише в старшій, а й в середній і навіть початковій ланці школи при обов'язковому використанні спеціальної методики. Виокремлення ООП як однієї з цілей навчального курсу інформатики показує необхідність і важливість його вивчення поруч з алгоритмічним і системним підходами до розв'язування завдань. Проте основна проблема полягає в тому, що випускники шкіл повинні мати уявлення про ООП вже сьогодні.

Історично склалося так, що поняття ООП спочатку знайшло своє віддзеркалення при розгляді питань алгоритмізації, коли алгоритм і його властивості вводилися через поняття «виконавця». У навчальних посібниках А.Г. Гейна [34; 33], А.Г. Кушніренка, В.А. Кайміна не реалізується об'єктно-орієнтований підхід, але їх виконавців можна трактувати як об'єкти з властивими їм методами [34; 103].

Н. В. Морзе ознайомлення учнів з алгоритмами та їх виконавцями розпочинає у 6 класі [93]. У її підручнику поняття «виконавець» спирається на поняття, введені раніше. Так, виконавець – це об'єкт (людина, тварина, машина), здатний виконувати задані йому команди (повідомлення, яке

спонукає до виконання певної дії). Система команд виконавця визначає набір команд, які він може виконувати. Уявлення про систему команд виконавця за підручником формується конкретно-індуктивним способом. Приклади виконавців та їх систем команд ілюструються за допомогою малюнків [93, с. 8-10]. Таке унаочнення сприяє усвідомленню того, що кожен виконавець може виконати лише команди із власної системи команд, а команди, що не входять до його системи, будуть незрозумілі. Варто зазначити, що під час вивчення алгоритмізації значна увага приділяється добору виконавця поставленої задачі, враховуючи те, що він також є певним об'єктом із притаманними йому властивостями та набором допустимих операцій, які треба аналізувати з метою правильного та ефективного їх використання. Тобто вже тут можна провести певні паралелі з використанням об'єктно-орієнтованого підходу в навчальному процесі. Таким чином, учні отримують початкові уявлення про виконавця, його середовище, команди, системи команд виконавця, алгоритм, які сприяють формуванню навичок виконувати готові алгоритми, а також складати прості алгоритми для виконавців, що працюють у певному зрозумілому для відповідної вікової категорії комп'ютерному середовищі, використовуючи просту систему їх команд. Сформовані у 6 класі поняття та вміння будуть використані як базові для подальшого вивчення цієї теми у старших класах.

У базовому курсі інформатики І. Г. Семакіна і Є. К. Хеннера [214] виконавець розглядається через такі поняття, як середовище, система команд, режими роботи, дані. Так якнайповніше можна визначити виконавця як об'єкт. Використання різних виконавців дає змогу говорити про пропедевтику понять ООП. Вони мають об'єктно-орієнтовані риси: об'єкт (виконавець) і засоби (команди) [214].

У цьому ракурсі навіть такі відомі виконавці, як, наприклад, машина Тюрінга, автомат Маркова, рекурсивні функції Чорча і Кліні та ряд інших, представники, здавалося б, традиційного підходу до алгоритмізації і програмування – застосовуються і з точки зору об'єктно-орієнтованої

парадигми. Прийнято вважати, що алгоритмом є послідовність дій, що може бути відображена у програмі, яка виконується за допомогою машини Тюрінга. Також інші названі алгоритмічні моделі є еквівалентами. Таким чином, будь-яка ідея в принципі може бути виражена будь-якою мовою (обчислювальною системою). Тобто об'єктно-орієнтована парадигма не дає якихось нових обчислювальних можливостей, що дають змогу розв'язувати задачі, неможливі для інших засобів, але вона робить ці задачі простішими і призводить їх до більш природньої форми. Це дає можливість справитись з проблемою розв'язання таким способом, який сприяє найефективнішому управлінню великими програмними системами.

Питання об'єктно-орієнтованого підходу розглядаються в посібнику Ю.А. Шафріна. У ньому розглядаються такі поняття, як об'єкт, набір об'єктів, частково абстрактні об'єкти. Можна сказати, що в цьому посібнику дається первинне знайомство з ООП [264].

У праці А.Г. Гейна наголошується, що парадигми об'єктно-орієнтованого і логічного підходів в інформатиці закладають методологічні основи для оволодіння іншими дисциплінами, а отже, в шкільному курсі інформатики доцільно мати розділи «Об'єктно-орієнтований підхід до інформаційного моделювання» і «Логічний підхід до інформаційного моделювання» [33].

У навчальному посібнику О. А. Кузнєцова та Н. В. Апатової «Основи інформатики» для 8-9 класів загальноосвітньої школи велике місце приділене вивченню мови програмування Паскаль. Як продовження курсу Н.В. Апатова в програмі для 10-11 класів розглядає питання ООП на Паскалі, логічного програмування на Пролозі і Ліспі (дерева, мережі, фрейми та інші), проте навчально-методичне забезпечення запропонованого курсу відсутнє [122; 7].

Н.Н. Леонова, Н.В. Коробков включили об'єктно-орієнтоване програмування в курс програмування системи «ліцей-виш». Вони запропонували навчати писати програми з використанням принципів інкапсуляції, поліморфізму, наслідування. За результатами педагогічного

експерименту авторів більшість учнів успішно опанували основні прийоми об’єктно-орієнтованого програмування. Це ще раз доводить доступність питань ООП для учнів [133, с. 41].

Розвиток технологій програмування йде від операційного, через структурний і модульний підхід до об’єктно-орієнтованого та візуального програмування. Кожен новий етап характеризується новим рівнем абстрагування, декомпозиції та ієрархії. У процесі «протягування» школяра цими етапами відбувається розвиток розумових здібностей, інтелекту, отже, вивчення програмування (мови та стилю) розвиває здатність до мислення. На думку С.М. Окулова, програмування багато в чому аналогічне процесу людського мислення [163, с. 34].

Технологія об’єктно-орієнтованого програмування розглядається в методичній праці А. Семенова та Н.Д. Угринович. У темі «Алгоритмізація і програмування» передбачається вивчення об’єктно-орієнтованої системи програмування. Автори реалізують ООП на основі однієї конкретної мови програмування – VisualBasic [249, с. 104].

До речі, вибір відповідної мови програмування для цього навчання є проблемою протягом багатьох років і є предметом обговорення педагогічного співтовариства в усьому світі. Дві найбільш популярні мови (приблизно рівні за популярністю) – Java і C – поглинають зараз приблизно третину світового користувацького попиту; ця частка має слабку тенденцію до зниження. Наступні чотири (PHP, C++, Visual Basic, C#), менш популярні – ще одну третину; ця частка має слабку тенденцію до підвищення. Досить мати уявлення про ці шість мов (і навички їх застосування), щоб ефективно спілкуватися із двома з кожних трьох програмістів світу. Кожна мова програмування є штучною, спроектованою для виконання обчислень на машинах, зокрема, на комп’ютері; в мові програмування визначається набір лексичних, синтаксичних і семантичних правил і вона може бути використана для створення програм, виконуваних за допомогою машин.

З.С. Сейдаметова розглядає класифікацію мов програмування за парадигмами (імперативна, процедурна, об'єктно-орієнтована, узагальнена, подійно-орієнтована, функціональна, аспектно-орієнтована, компонентно-орієнтована). Вчений відзначає, що найбільш часто використовуваною є об'єктно-орієнтована парадигма програмування. Привернення уваги педагогічного співтовариства України до цих тверджень може стати стимулом для широкого обговорення національних педагогічних тенденцій, які склалися в цій сфері [212, с. 35-41].

Проте без опанування фундаментальних понять ООП, знання якоїсь мови програмування можуть «застаріти» через декілька років у зв'язку зі швидким розвитком засобів програмування. Отже, при розробці навчального курсу потрібно пам'ятати, що об'єктно-орієнтоване програмування – це новий стиль мислення, а не розширення мови програмування.

Однак вивчення лише принципів і засобів об'єктно-орієнтованого програмування в загальноосвітній школі недостатньо, оскільки головне завдання даного курсу – навчити учнів застосовувати наявні знання при створенні ефективних програм. Отже, необхідно поєднувати вивчення основних принципів, засобів об'єктно-орієнтованого програмування і вивчення об'єктно-орієнтованої мови програмування. Частково зазначений підхід реалізований Н.Д. Угриновичем в навчальному посібнику для 10-11 класів [250]. Тут разом з вивченням об'єктно-орієнтованого програмування мовою VisualBasic даються основні поняття ООП: об'єкт, властивості об'єкта, класи об'єктів, повідомлення і засіб. Проте, не зважаючи на те, що VisualBasic працює з об'єктами, він не є об'єктно-орієнтованою системою програмування, оскільки його об'єкти не мають таких властивостей, як наслідування і поліморфізм, що обов'язково мають бути наявними в об'єктів справжнього об'єктно-орієнтованого середовища.

Дослідження Н.Я. Салангіної присвячене розробці підходу, що дає змогу реалізувати лінію алгоритмізації в курсі «Мови і методи програмування» [209, с. 53-58]. Запропоновані методичні підходи спрямовані

на вивчення ООП у школі і для їх використання в загальноосвітніх установах потрібні спеціально розроблені методики.

Навчальний посібник для школярів В.Б. Попова [177] присвячений вивченню основ програмування мовою Паскаль і використання інтегрованого середовища програмування ТурбоПаскаль 7.0 при розробці програм. В одній з частин посібника описано основи об'єктно-орієнтованого програмування в середовищі ТурбоПаскаль для Windows. Перші чотири частини посібника насичено великою кількістю детально розібраних прикладів програм, питань для перевірки засвоєння теоретичного матеріалу і завдань для формування міцних навиків програмування. Матеріал п'ятої частини – об'єктно-орієнтоване програмування – суто теоретичний, вивчення якої накопичує в учнів теоретичні знання, а не вміння і навички створення об'єктно-орієнтованих програм. Розглянутий вище посібник є хорошою підмогою для вчителів інформатики і учнів у вивченні мови Паскаль як мови структурного програмування, а не об'єктно-орієнтованого програмування.

Р.І. Баженовим обґрунтована і розроблена методика використання ООП для розвитку розумових дій школярів під час навчання базового курсу інформатики [15]. Дослідження показали доступність теми для вивчення учнями, підвищення якості засвоєння знань з інформатики, довели підвищення як теоретичної, так і прикладної спрямованості базового курсу інформатики. Розглянутий об'єктний підхід реалізований з використанням мови декларативної парадигми – мови логічного програмування.

Доцільність використання ООП в прикладній інформатиці, тобто при освоєнні роботи з основними засобами ІТ, розглядається в навчальному посібнику Б.Є. Стариченка [235 с. 85]. Автором обґрунтоване використання основних понять об'єктно-орієнтованого програмування (об'єкт, клас об'єктів, властивість об'єкта, подія та ін.) як під час роботи з прикладними програмними системами (в цьому випадку йдеться про інформатичні об'єкти), так і при розгляді сучасних систем програмування (створювана програмістом програма будується із програмних об'єктів).

Такий підхід дає змогу розглядати програмування як одну з ІКТ. Цим забезпечується ідейна, логічна і термінологічна єдність підходів до освоєння будь-якої ІТ на об'єктній основі. «Об'єктний підхід об'єднує нішу і програміста прикладної інформатики, і користувача. Єдність підходів до структури алгоритму і програми дуже важлива з ідейної точки зору, оскільки демонструє спільність і універсальність засобів інформатики, з одного боку, та помітно полегшує освоєння програмування, з іншого боку», – стверджує Б.Є. Стариченко [236, с. 12].

Проблема використання ООП в навчанні інформатики досліджена у працях Л.В. Гришко. На її думку, при «об'єктному підході» до навчання програмування здійснюється раннє ознайомлення з об'єктно-орієнтованим програмуванням, яке останніми роками стало надзвичайно важливим. Однак об'єктно-орієнтована модель має недоліки, зокрема багато мов, які використовуються для об'єктно-орієнтованого програмування (C++, Java, C#), є набагато складнішими за класичні процедурні мови. Вибираючи цю модель, необхідно ретельно продумати методику вивчення матеріалу, інакше деталі вибраної мови можуть затінити суть об'єктно-орієнтованого програмування [45, с. 134-148].

І. Н. Лукаш зазначає, що цілеспрямоване використання засобів технологій об'єктно-орієнтованого програмування у процесі навчання інформатики позитивно впливає на формування інтелектуальних умінь учнів. Якщо раніше навчання в основному було спрямоване на формування знань, умінь, навичок, то зараз пріоритетним критерієм вважається рівень сформованості відповідних якостей мислення (глибини, гнучкості, критичності, стійкості, усвідомленості розумової діяльності тощо). Застосування такої методичної системи сприяло переходу учнів від репродуктивного рівня засвоєння навчального матеріалу до активного його здобуття та логічного осмислення, підвищенню рівня їх інтелектуальної активності та загальної успішності в навчанні [139, с. 21].

Заслуговує на увагу науковий доробок Ю.С. Рамського та І.М. Лукаш, який відображений у циклі статей з методики навчання основ об'єктно-орієнтованого програмування [196]. Вчені зазначають, що при об'єктно-орієнтованому підході існує можливість за допомогою об'єктів об'єднати всі майже незалежні подання системи з різних точок зору в єдине семантичне ціле. Об'єктно-орієнтоване програмування забезпечує механізм відокремлення операцій занесення і отримання даних від реалізації конкретного формату об'єкта. Використовуючи сучасні об'єктно-орієнтовані програмні засоби, можна створювати великі програмні компоненти, придатні для багаторазового використання, що дає змогу значно економити час учнів під час створення програмних проектів і концентрувати їх зусилля на творчому мисленні. У статтях автори охарактеризували основні положення об'єктно-орієнтованого програмування, становлення, удосконалення об'єктної моделі від її реалізації в Pascal до реалізації в Delphi; розглянули можливості застосування конкретного інформаційного середовища для вивчення різних тем шкільного курсу інформатики: бази даних, Web-програмування, опрацювання і графічне подання табличних даних; розглянули засоби автоматизації розробки програмних проектів на основі об'єктно-орієнтованого програмування [195].

Нині серед українських видань переважають навчальні посібники, орієнтовані на оволодіння конкретною мовою об'єктно-орієнтованого програмування. Проте суспільство потребує наукових розробок, які були би присвячені власне ідеології і методології ООП, а також містили глибокий аналіз основних ідей, принципів і понять, що лежать в основі об'єктно-орієнтованої парадигми програмування.

Таким чином, зі сказаного вище бачимо, що в поглибленому курсі інформатики існує невідповідність між теоретичним та технологічним рівнями методики навчання об'єктно-орієнтованого програмування. Це виражається в наступному:

– повна відсутність матеріалу з ООП в навчальних посібниках, рекомендованих для вивчення базового курсу інформатики;

– лише пропедевтика об’єктно-орієнтованого програмування;

– вивчення деяких понять ООП і технології візуального проектування на основі VisualBasic;

– присутнє лише теоретичне об’єктно-орієнтоване програмування;

– вивчення лише технології візуального проектування (середовище Delphi).

Перехід до об’єктно-орієнтованої парадигми можна вважати кардинальною зміною, що спричинила собою один з основних напрямів – розвиток технології об’єктно-орієнтованого програмування.

На основі аналізу структури і змісту курсів, орієнтованих на навчання об’єктно-орієнтованого програмування, були зроблені висновки, що, незважаючи на ряд навчальних курсів, розроблених провідними фахівцями для навчання вказаного програмування в межах шкільного курсу інформатики, методичні підходи до навчання висвітлені недостатньо. Розгляд нормативних документів на профільному рівні середньої (повної) загальної освіти з інформатики та ІКТ щодо розгляду понять алгоритмізації та програмування дає змогу зробити такі висновки:

1. Вивчення понять алгоритмізації та програмування спрямоване на досягнення таких цілей, як оволодіння вміннями побудови логічних програм формальною мовою, що задовільняють заданій умові; створення програми мовою програмування; розвиток алгоритмічного мислення, здібностей до формалізації; набуття досвіду проектної діяльності.

2. Обов’язковий мінімум змісту основних освітніх програм містить такі базові поняття інформатики та ІКТ: елементи теорії алгоритмів; формалізація поняття алгоритму; еквівалентність алгоритмічних моделей; побудова алгоритмів і практичні обчислення; мова програмування; типи даних; основні конструкції мови програмування; система програмування; основні етапи розробки програм; розбиття задачі на підзадачі.

3. У вимогах до рівня підготовки випускників визначені знання основних конструкцій мови програмування, а також властивостей алгоритмів і основних алгоритмічних конструкцій.

4. В основному переважає структурна парадигма вивчення алгоритмізації та основ програмування у шкільному курсі інформатики.

5. Запропоновано засоби побудови і дослідження фізичних, математичних та інших моделей засобами об’єктно-орієнтованого програмування (наприклад, побудова графіків функцій; біологічні моделі розвитку популяцій: моделі необмеженого зростання, обмеженого зростання, оптимізаційне моделювання в економіці), що демонструє можливості використання даної парадигми виключно в межах візуалізації процесів.

6. Відзначена необхідність детальніших занять з навчання об’єктно-орієнтованого програмування школярів для підвищення мотивації до вивчення програмування.

7. Відсутні методичні підходи до навчання об’єктно-орієнтованого програмування без попереднього розгляду структурної парадигми в умовах шкільної освіти.

Отже, можна говорити про необхідність вдосконалення існуючих методичних підходів до вивчення об’єктно-орієнтованого програмування в системі базової освіти дітей з поглибленим вивченням інформатики, оскільки аналіз сучасного рівня розвитку інформатики як науки дає змогу стверджувати про масовий перехід до нової об’єктно-орієнтованої парадигми програмування.

Висновки до першого розділу

Перший розділ дослідження присвячений детальному розгляду теоретичних та методологічних аспектів навчання алгоритмізації та програмування учнів старших класів з поглибленим вивченням інформатики.

Проведено аналіз специфіки формування системно-логічного мислення учнів старших класів та місця алгоритмізації і програмування в цьому процесі.

Можемо підсумувати, що системно-логічне мислення – це нова категорія свідомості, підґрунтям якої є вміння швидко і якісно, враховуючи швидкий розвиток ІТ, розв'язувати складні завдання. Водночас це новий щабель психологічних і психофізіологічних можливостей людини в галузі мислення, який робить можливим опрацювання та засвоєння великих обсягів даних, визначення відповідних певній ситуації відомостей, здобуття нових знань.

Таким чином, формування системно-логічного мислення є процесом розвитку здібностей до роздумів, який відбувається в процесі створення алгоритмів та програм шляхом багаторазових дій. Отже, системно-логічне мислення – це діяльнісне управління пошуком розв'язування задач, результатом якого є алгоритми та програми як специфічні продукти.

Проведений аналіз специфіки формування системно-логічного мислення учнів старших класів та місця алгоритмізації і програмування в цьому процесі дає змогу зробити висновок про те, що ґрунтовне вивчення цих розділів інформатики є інструментом розвитку розумових здібностей школярів, точніше – інструментом пізнання в процесі навчання.

Можна говорити про розвиток системно-логічного мислення як передумову формування інформатичних компетентностей випускника школи, спираючись на які студент вузу зможе розвивати їх у подальшому як складову системи професійних компетентностей фахівця, а це відповідно, сприятиме ефективному оволодінню майбутньою професією.

Отже, формування та розвиток інформатичних компетентностей є важливим завданням сучасної освітньої підготовки учнів з поглибленого

курсу не лише інформатики, а й інших дисциплін. Ці компетентності (інтелектуальні, особистісні, соціально-значущі та професійні) за своєю сутністю є одними з головних ключових компетентностей майбутнього фахівця. Вони передбачають формування початкових навиків роботи з комп'ютерною технікою і програмним забезпеченням, засвоєння основ алгоритмізації та програмування. Система інформатичних компетентностей особистості характеризує її знання, вміння, навички, прагнення, мотиви, інтереси, здатність і готовність до використання ІКТ у навчальній і подальшій професійній діяльності.

Розвиток умінь та навичок роботи з ІКТ є складним процесом, тому в структурі системи інформатичних компетентностей виокремлюємо такі компоненти: мотиваційний, когнітивний (інформаційний, змістовий), технологічний (поведінковий, діяльнісний) та особистісний (емоційно-вольовий). Ці компетентності ефективно формуються в учнів старших класів під час вивчення алгоритмізації та програмування.

Розуміючи психологічні особливості учнів, учитель повинен уміти дієво сприяти формуванню і розвитку цих компетентностей, допомогти школярам розкрити власний творчий потенціал, вибрати індивідуальний освітній шлях.

Також з'ясовано та обґрунтовано дидактичну доцільність використання ООП до навчання алгоритмізації та програмування старшокласників в процесі поглибленого вивчення інформатики на прикладі застосування СКМ Maple.

Проведений аналіз дає змогу зробити певні висновки.

1. Вивчення понять алгоритмізації та програмування спрямоване на досягнення таких цілей, як оволодіння вміннями побудови логічних програм формальною мовою, що задовільняють заданій умові; створення програми мовою програмування; розвиток алгоритмічного мислення, здібностей до формалізації; набуття досвіду проектної діяльності.

2. Обов'язковий мінімум змісту основних освітніх програм містить такі базові поняття інформатики та ІКТ: елементи теорії алгоритмів; формалізація

поняття алгоритму; еквівалентність алгоритмічних моделей; побудова алгоритмів і практичні обчислення; мова програмування; типи даних; основні конструкції мови програмування; система програмування; основні етапи розробки програм; розбиття задачі на підзадачі.

3. У вимогах до рівня підготовки випускників визначені знання основних конструкцій мови програмування, а також властивостей алгоритмів і основних алгоритмічних конструкцій.

4. В основному переважає структурна парадигма вивчення алгоритмізації та основ програмування у шкільному курсі інформатики.

5. Запропоновано засоби побудови і дослідження фізичних, математичних та інших моделей засобами об'єктно-орієнтованого програмування (наприклад, побудова графіків функцій; біологічні моделі розвитку популяцій: моделі необмеженого зростання, обмеженого зростання, оптимізаційне моделювання в економіці), що демонструє можливості використання даної парадигми виключно в межах візуалізації процесів.

6. Існує необхідність змістовного навчання об'єктно-орієнтованого програмування школярів для підвищення мотивації до вивчення програмування, оскільки:

– використання ООП обумовлене як в роботі з прикладними програмними системами, так і в застосуванні сучасних систем програмування, що дає змогу розглядати програмування однією з ІКТ, демонструючи загальність та універсальність методів інформатики і полегшуючи навчання алгоритмізації та програмування на основі об'єктно-орієнтованого підходу;

– Сьогодні ООП є першочерговим в галузі розробки та використання програмних засобів;

– вивчення універсальних методів і прийомів роботи з об'єктно-орієнтованим програмним забезпеченням дає змогу школярам в подальшому самостійно засвоювати та використовувати засоби ІКТ для розв'язування поставлених задач.

7. Відсутні методичні підходи до навчання об’єктно-орієнтованого програмування без попереднього розгляду структурної парадигми в умовах шкільної освіти, тому застосування Maple є доцільним в процесі вивчення старшокласниками алгоритмізації та програмування, оскільки цей комп’ютерний математичний пакет забезпечує їм зручне інтелектуальне середовище для математичних досліджень.

8. Існує нагальна потреба вдосконалення існуючих методичних підходів до вивчення об’єктно-орієнтованого програмування в системі базової освіти дітей з поглибленим вивченням інформатики.

Основні результати розділу представлено в опублікованих працях [292; 270; 265; 273; 274].

РОЗДІЛ 2.
МЕТОДИЧНА СИСТЕМА НАВЧАННЯ АЛГОРИТМІЗАЦІ ТА ПРОГРАМУВАННЯ СТАРШОКЛАСНИКІВ

2.1. Особливості методичної системи навчання алгоритмізації та програмування

2.1.1. Компоненти методичної системи навчання алгоритмізації та програмування

На основі аналізу існуючих програм та навчально-методичних посібників з інформатики, враховуючи роль розділу алгоритмізації та програмування в навчанні учнів у класах з поглибленим вивченням інформатики, розглянемо модель методичної системи навчання алгоритмізації та основ програмування старшокласників з використанням об'єктно-орієнтованого підходу і визначимо її місце у шкільному курсі інформатики.

Пропонована методична система є інваріантною щодо вибору мови програмування. Йдеться не стільки про конкретні засоби об'єктно-орієнтованої алгоритмізації, скільки про операційні можливості використання популярних мовних засобів. Тому такий підхід повинен посісти важливе місце в процесі навчання курсу інформатики щодо різноманітних мовних середовищ структурного, процедурного підходу та ООП у програмуванні. Крім того, курс алгоритмізації на основі ООП займе власну нішу, пов'язану з новими методами розв'язування сучасних практичних задач в галузі програмного забезпечення.

Курс алгоритмізації та програмування гармонійно включається в існуючу сьогодні систему інформатичної підготовки школярів, не пошкоджуючи взаємозв'язків усередині системи.

Перед тим, як зупинитися на змістових деталях методичної системи, визначимо завдання і мету навчання алгоритмізації і програмування.

Завдання:

− з’ясувати ключові поняття алгоритмізації і програмування;

− виявити загальні закономірності і принципи алгоритмізації;

− навчити основних способів організації операцій і даних;

− розглянути базові алгоритмічні конструкції при складанні описів алгоритмів та визначити ті, які доцільно використати для розв’язування конкретної задачі;

− охарактеризувати принципи роботи основних виконавців алгоритмів, розглянути сучасні засоби тестування складених алгоритмів;

− сформувати елементи операційного стилю мислення, який полягає в умінні: формалізувати задачу; виокремити в ній логічно самостійні частини; визначити їх взаємозв’язки; спроектувати алгоритм розв’язування; визначити ефективні шляхи отримання результату; інтерпретувати та аналізувати результати;

− з’ясувати сутність поняття «об’єкт» і його застосування в сучасному програмуванні, розглянути використання об’єктів та їх класів;

− використовувати прості алгоритми і структури даних, які не вимагають поглиблених знань особливостей конкретної мови;

− дослідити та пояснити основні принципи об’єктно-орієнтованого програмування як логічного продовження і розвитку процедурного програмування;

− розглянути основні етапи процесу розробки програми;

− сформувати знання та вміння використання на практиці основних конструкцій ООП, сприяти оволодінню учнями навичками програмування в об’єктах;

− проаналізувати моделювання і проектування як фундаментальні компоненти процесу програмування із використанням конкретного комп’ютерного середовища;

− з’ясувати можливості використання програмування в житті суспільства.

З узагальненням цих завданнь сформульовано мету [180]:

– розвинути системно-логічне, аналітичне мислення та основні види розумової діяльності: уміння використовувати індукцію, дедукцію, аналіз, синтез, робити висновки, узагальнення;

– сформувати теоретичну базу знань учнів щодо процесів перетворення, передавання та використання даних;

– розвинути уміння розв’язувати змістовні задачі різного рівня складності, олімпіадні задачі, користуючись відомими теоретичними положеннями, математичним апаратом, літературою та комп’ютерною технікою;

– підготовити старшокласників до участі в олімпіадах, конкурсах, турнірах, науково-практичних конференціях, конкурсах-захистах науково-дослідницьких робіт різного рівня та інших інтелектуальних змаганнях;

– довести вивчення інформатики до творчого рівня;

– сформувати бачення учнями можливостей використання набутих знань у їх майбутній професійній діяльності;

– інтегрувати навчання інформатики з іншими предметами, що викладаються в навчальному закладі.

Мета курсу досягається через практичне оволодіння учнями навичками роботи з основними складовими сучасного програмного забезпечення комп’ютерів, ознайомлення з функціональним призначенням основних пристроїв комп’ютера, з основами технології розв’язування задач за допомогою комп’ютера, починаючи від їх постановки й побудови відповідних інформаційних моделей і завершуючи інтерпретацією результатів, отриманих за допомогою комп’ютера.

Для створення методичної системи навчання алгоритмізації та програмування учнів старших класах з поглибленим вивченням інформатики на основі ООП було дібрано зміст навчання. Добір здійснювався відповідно до критеріїв, широко відомих в педагогіці:

1) наукова строгість та послідовність: кожна тема повинна бути логічно-послідовною, реалізація кожної теми має відповідати оцінці наукового рівня і характеристикам логічної строгості;

2) системність наукових знань: кожне поняття повинно мати строго визначене місце в системі понять курсу; використовувані методи мають забезпечувати раціональне розв'язування практичних завдань;

3) принцип доступності: поступовий перехід від простого до складного; доцільність нової термінології та символіки, їх відповідність базі знань, вмінь та навичок учнів;

4) принцип практичної направленості теоретичного матеріалу: зв'язок теорії з практикою як видом людської діяльності; здобуті теоретичні знання повинні мати практичне застосування;

5) відповідність меті навчання: здобуті знання, вміння та навички повинні сприяти досягненню мети;

6) єдність теорії, технології та техніки: використання взаємозв'язків між різноманітними аспектами інформатики (теоретичними, технологічними і технічними).

Окрім цього, методична система має відповідати наступним вимогам:

1. Вона повинна бути інваріантною щодо мови програмування. Добір мови має бути зумовлений лише класом задач, напрямом педагогічного процесу, рівнем підготовки учнів тощо.

2. Оскільки об'єктно-орієнтований підхід до навчання алгоритмізації та програмування служить логічним завершенням понять структурного програмування, яке стало традиційним об'єктом навчання, то пропонована система повинна містити всі поняття, покликані навчити учнів методів структурного складання алгоритмів.

3. У зміст навчання методичної системи мають бути включені основні процеси і особливості об'єктно-орієнтованого конструювання програм.

Як показує досвід, доцільно розпочинати навчання алгоритмізації з обговорення схем програм проектування, характеристики її методів і засобів,

включаючи знайомство учнів із загальними характеристиками найпоширеніших мов програмування та їх порівняння.

Згідно з положеннями Державного стандарту базової і повної загальної середньої освіти у старшій школі широкого поширення набуло профільне навчання [54]. Це дає змогу за рахунок змін у структурі, змісті та організації освітнього процесу навчання інформатики повніше враховувати інтереси, нахили та здібності учнів, створювати умови для навчання старшокласників відповідно до їх професійних інтересів та намірів щодо продовження освіти. Реалізація профільного навчання інформатики у 10-11 класах забезпечується також системою курсів за вибором (за рахунок варіативного компоненту), які певною мірою враховують інтереси і можливості учнів цього профілю. Курси за вибором, спецкурси та факультативи поглиблюють та розширюють основний курс інформатики відповідно до профілю навчання, надають можливості для організації творчої роботи учнів через систему індивідуальних завдань професійної спрямованості. Профільний рівень вивчення інформатики поданий в межах інформаційно-технологічного та фізико-математичного профілів навчання школярів. Особливістю поглибленого навчання інформатики на такому рівні є виокремлення в її змісті освітніх ліній, пов’язаних з алгоритмізацією і програмуванням.

Поглиблене вивчення інформатики в 10–11 класах ЗОШ передбачає вивчення двох паралельних змістових ліній – інформаційно-комунікаційні технології і основи алгоритмізації та програмування, які є взаємопов’язаними і послідовно узгодженими. При цьому в процесі навчання передбачається формування предметних ІКТ-компетентностей та ключових компетентностей при виконанні репродуктивних, проблемних і евристичних (частково-пошукових) завдань, зокрема індивідуальних і групових проектів, компетентнісних задач, виконання яких передбачає використання кількох різних ІТ або програмних середовищ. Це сприяє розвитку алгоритмічного мислення, елементів системного мислення, здібностей до формалізації, вихованню, формуванню установки на позитивну соціальну діяльність в

інформаційному суспільстві. Учні набувають досвіду проектної діяльності, створення, редагування, оформлення, збереження, передавання інформаційних об’єктів різного типу з використанням сучасних програмних засобів, колективної реалізації інформаційних проектів, інформаційної діяльності в різних сферах, затребуваних на ринку праці.

Старшокласникам, що вивчають поглиблено інформатику, важливо не просто оволодіти вмінням написання програмних кодів певною мовою програмування, а й отримати практичні навички алгоритмізації і програмування, використовуючи при цьому сучасні засоби та інформаційні технології. У зв’язку з цим більш ґрунтовне вивчення інформатики набуває особливої значущості. У цьому разі основний акцент діяльності учнів переноситься на створення алгоритмів і програм у середовищі сучасних пакетів автоматизованих обчислень. Водночас сучасні засоби навчання повинні задовільнити певні специфічні вимоги, такі, наприклад, як зручний інтерфейс, можливість імпорту та експорту даних, доступ в інтернет і багато інших.

При навчанні зазначених розділів курсу інформатики використовуються як універсальні системи програмування, так і стандартні прикладні програми, в тому числі системи програмування мовами Basic, Pascal, C, системи управління базами даних тощо. Наразі розроблено велику кількість ефективних методів навчання алгоритмізації та програмування з використанням таких засобів. Підготовка школярів в межах поглибленого навчання вимагає освоєння старшокласниками прийомів автоматизації складних математичних розрахунків, графічного подання інформатичних та математичних об’єктів, інформаційного, зокрема математичного моделювання, основаного на побудові алгоритмів і програм. На жаль, перераховані системи програмування та прикладні програми не тільки не завжди задовільняють вищеназвані вимоги, а й не цілком достатні для повноцінного вирішення завдань профільної підготовки школярів. Тому для

підготовки школярів в межах поглибленого вивчення інформатики необхідно використовувати додаткові засоби і методи навчання.

При цьому треба враховувати, що у процесі такого навчання доцільно використовувати комп'ютерні математичні пакети. Більшість таких програмних засобів разом з потенційними можливостями навчання математики та фізики мають і надзвичайно істотний потенціал щодо навчання алгоритмізації та програмування.

Однак, питання стосовно методики використання комп'ютерних математичних пакетів при навчанні шкільного курсу інформатики на поглибленому рівні ще недостатньо вивчені.

Етап програмування за умови якісно розробленого алгоритму виконується автоматичною заміною блоків схеми алгоритму операторами мови програмування, але вимагає знання методів і правил формування програм. Для непрофесійних користувачів, які вміють структурувати і алгоритмізувати задачу, але не мають навичок складання програм і роботи з системами програмування, цей етап викликає значну трудність. На рис. 2.1 показано, що розв'язування може бути отримане двома шляхами: перший – за допомогою створення користувальницької програми конкретною мовою програмування, другий – із використанням прикладної системи без безпосереднього програмування.

Стрімкий розвиток прикладних систем із зручним графічним інтерфейсом користувача дозволив значно збільшити кількість розв'язаних із їх допомогою задач. З урахуванням аналізу процесу розв'язування прикладної задачі на комп'ютері розроблено та теоретично обґрунтовано методичну систему навчання алгоритмізації та програмування, яка відображає (охоплює) послідовність етапів розв'язування прикладної задачі на комп'ютері (аналізу, побудови алгоритму, виокремлення об'єктів, реалізації алгоритму, візуалізації, виведення результату та оформлення звіту) та взаємозв'язки між ними (рис. 2.1).

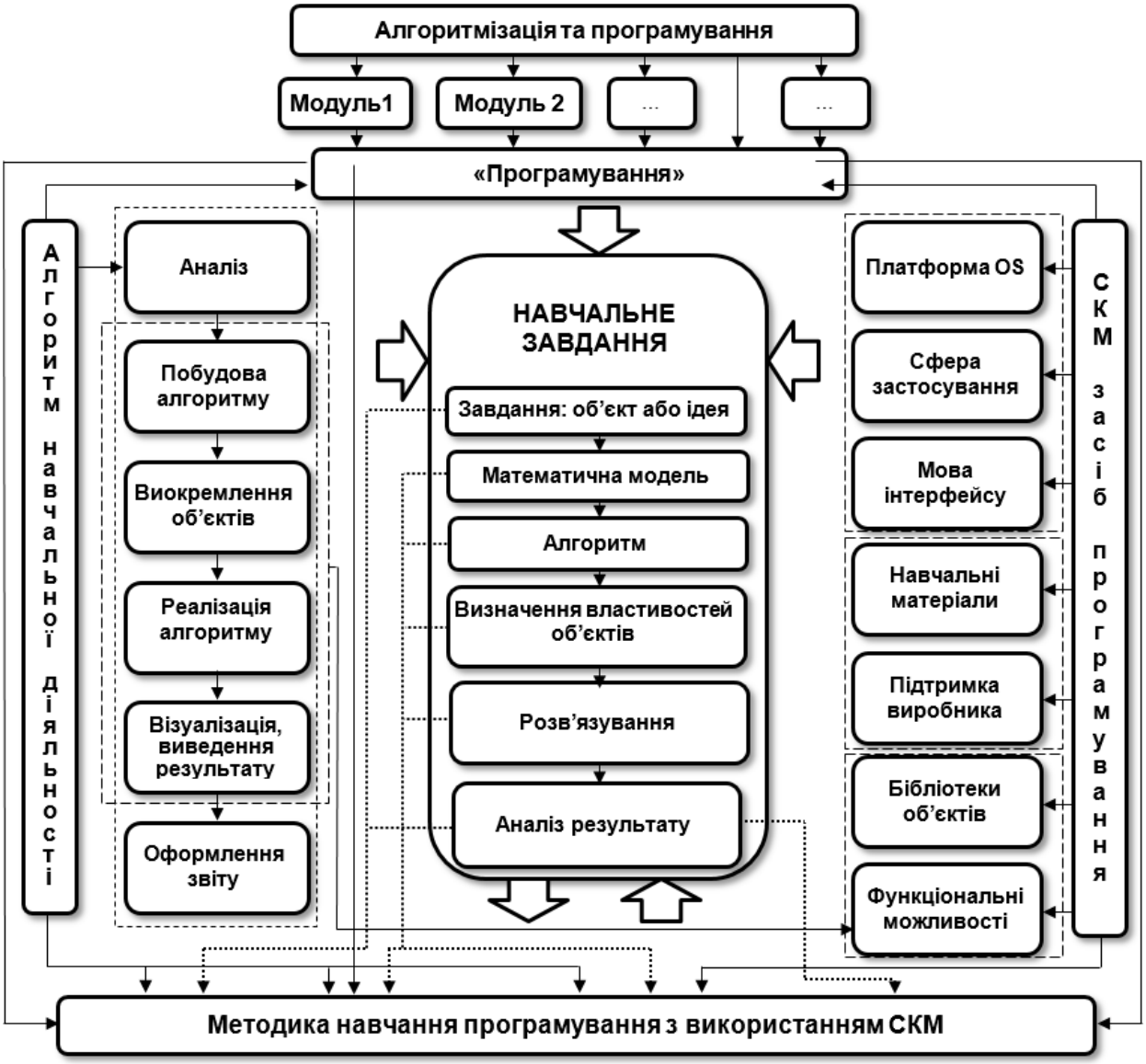


Рис. 2.1. Модель навчання учнів старших класів

Розглянемо основні змістові лінії пропонованої методичної системи.

Перша змістова лінія, присвячена простим алгоритмам, поняттям програми, графічним-схемам найпростіших алгоритмів, програмуванню в інтегрованому середовищі. У цій частині розглядаються особливості операторів введення-виведення, способів маніпулювання комірками пам'яті і типізації даних, методів розробки розгалужених та циклічних алгоритмів. Більшу увагу в цьому розділі треба звернути на логіку умов логічних та циклічних конструкцій, що базується на використанні набору логічних зв'язків. Крім того, важливою є проблема вивчення видів умов і циклів та їх

відмінностей один від одного з точки зору раціонального застосування в програмуванні.

Друга змістова лінія розширює перелік типів даних за рахунок можливості конструювання нових типів даних, використовуючи вже вивчені, а також комбінуючи їх. Разом з тим, збільшується коло задач, поставлених перед учнями: від роботи з окремими даними вони переходять до опрацювання таблиць, матриць, різнорідних інформаційних структур.

Підпрограми, які подані процедурами та функціями, утворюють третю змістову лінію методичної системи. У цій частині навчального процесу основний акцент спрямований на методику структурного програмування і технологію покрокової деталізації при поділі програми на підпрограми. На цьому етапі частина навчального часу відводиться на вивчення використання рекурсії. В подальшому всі перераховані поняття служитимуть базою при навчанні об’єктної орієнтованості сучасних мов програмування.

Четверта змістова лінія навчання алгоритмізації та програмування присвячена основним способам опрацювання та збереження даних.

Своєрідним мостом, який пов’язує традиційне структурне програмування та програмування на основі ООП, є модульне програмування, в якому обговорюються питання поділу програм на замкнені модулі.

Усі зазначені компоненти методичної системи навчання алгоритмізації та програмування не є кардинально новими і відповідають традиційним правилам навчання структурного підходу в програмуванні. Однак всі вони побудовані так, щоб весь курс можна було доповнити темами, присвяченими безпосередньо методам об’єктно-орієнтованого програмування.

Всі наступні частини курсу алгоритмізації та програмування включають вже поняття програмування на основі ООП. Після закінчення цих розділів, крім розширеної понятійної бази, учні знайомляться з набором прийомів і методів, які дають змогу зробити процес створення програм більш інтенсивнішим.

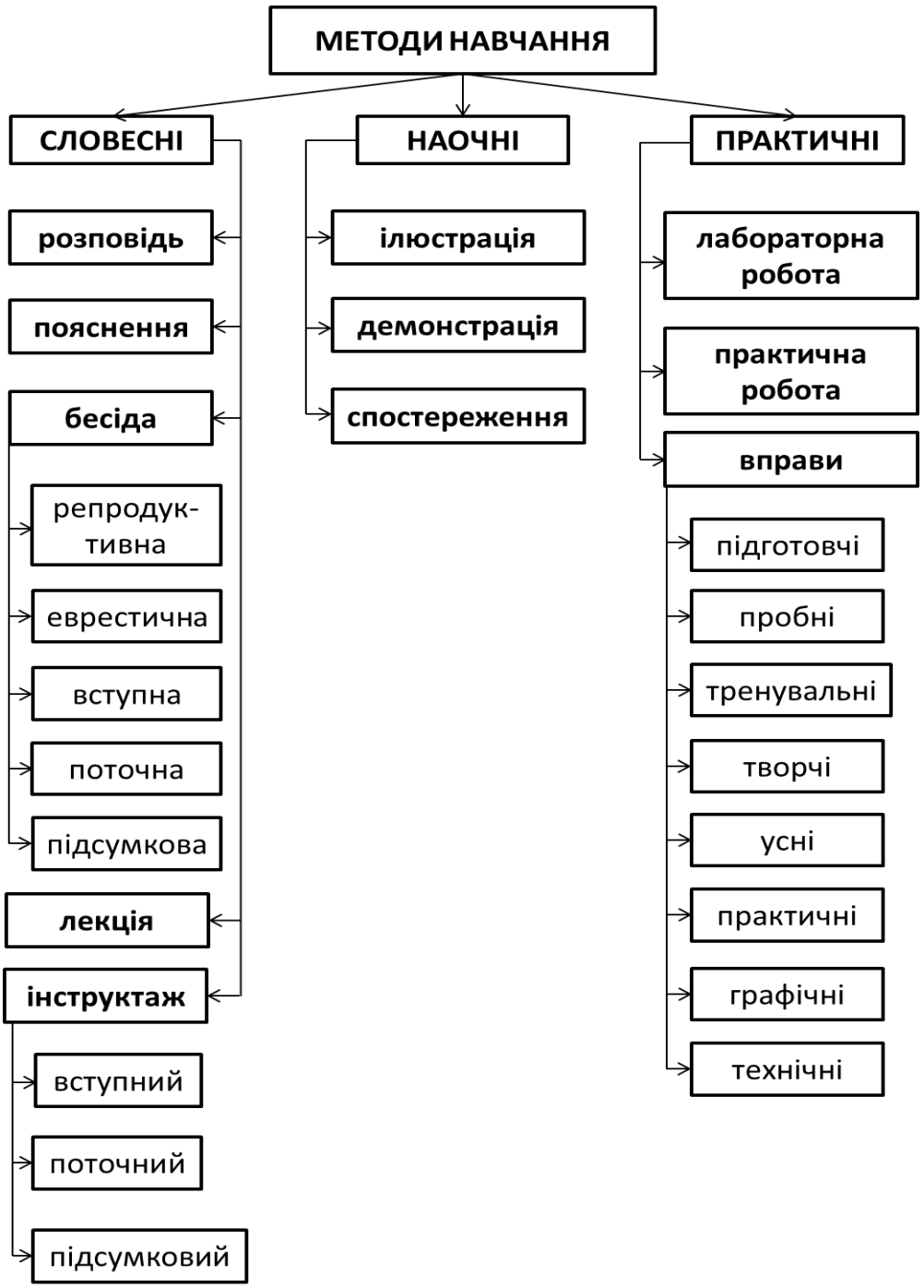


Рис. 2.2. Класифікація методів навчання (за С. Петровським, Є. Голантом)

Для завершення побудови методичної системи навчання алгоритмізації та програмування на основі ООП розглянемо методи, форми та засоби, які можуть застосовуватися при навчанні старшокласників в процесі поглибленого вивчення інформатики.

Подамо класифікації методів навчання в межах курсу алгоритмізації та програмування на основі ООП:

- за джерелом передавання та сприймання навчальних повідомлень – словесні, наочні, практичні (за С. Петровським, Є. Голантом) (див. рис. 2.2);

- за характером пізнавальної діяльності учнів – пояснювально-ілюстративний, репродуктивний, проблемне викладання, частково-пошуковий, дослідницький [107];

- залежно від основної дидактичної мети і завдань – методи оволодіння новими знаннями, формування вмінь і навичок, перевірки та оцінювання знань, умінь і навичок [49]; методи усного подання навчального матеріалу, закріплення навчального матеріалу, самостійної роботи учнів з осмислення й засвоєння нового матеріалу роботи із застосування знань на практиці та вироблення вмінь і навичок [259];

- класифікація з точки зору цілісного підходу до діяльності у процесі навчання – методи організації та здійснення навчально-пізнавальної діяльності; стимулювання й мотивація учіння, контролю, самоконтролю, взаємоконтролю і корекції, самокорекції, взаємокорекції в навчанні [12].

Всі перелічені методи використовуються при навчанні програмування на основі ООП. Детальніше на характеристиці цих методів ми зупинимося у розділі 2.4.

Форми організації навчання алгоритмізації та програмування на основі ООП є традиційними у педагогіці [149]:

- урок засвоєння нових знань – пояснення нового матеріалу, найбільш усталена форма заняття, методичною ціллю якої є розвиток пізнавальної активності учнів, інтересів до навчального предмета;

- урок застосування знань, умінь і навичок (практична робота) – використання дає змогу інтегрувати теоретико-методологічні знання і практичні вміння та навички учнів в єдиному процесі їх діяльності навчально-дослідницького характеру, оскільки специфіка занять з інформатики проявляється насамперед у переважанні практичних робіт з використанням комп’ютера.

Це відіграє важливу роль в курсі інформатики загалом, оскільки його основною особливістю є узгоджене поєднання теорії та практики з попереджуючим поданням теорії і наступною апробацією учнями теоретичних положень на практиці з використанням комп'ютера. На практичних заняттях використовуємо фронтальну форму роботи, що включає в себе одночасне виконання загального завдання всіма учнями класу, індивідуальну роботу, коли кожному школяреві даються різні за характером завдання, та спільну групову роботу, при якій групи об'єднуються для спільного дослідження визначеної моделі і раціональності застосування тих чи інших структур даних, методів організації пам'яті, виконання певних алгоритмів тощо.

Засобами навчання алгоритмізації та програмування у старших класах є: підручники та навчальні посібники, в яких описані традиційні теми; різноманітне програмне забезпечення на основі ООП; методичні розробки учителя, а також ТЗН, які використовуються на уроках вивчення алгоритмізації та програмування в старших класах: комп'ютери, ноутбуки, проектор, екран, планшети тощо.

Рис. 2.3. Критерії ефективності методичної системи

На рис 2.3 продемонстровано основні критерії ефективності пропонованої методичної системи: операційно-діяльнісний, навчально-компетентнісний і мотиваційно-особистісний та обґрунтовано показники і засоби діагностики до кожного з них.

Таким чином, запропонована методична система навчання алгоритмізації та основ програмування старшокласників в процесі поглибленого вивчення інформатики в ЗОШ озброює учнів необхідними знаннями, які будуть базою для розуміння можливостей та обмежень використання персональних комп'ютерів та програмного забезпечення у житті суспільства; опанування курсу алгоритмізації забезпечує здобуття фундаментальних знань в галузі інформатики; введення ООП дає змогу адаптувати здобуті знання до швидкозмінних обставин у сфері нових ІКТ, що допомагає на якісно новому рівні використовувати такі технології у навчальному процесі та для організаційних цілей.

2.1.2 Машина Тюрінга як універсальний виконавець алгоритмів та її застосування в процесі поглибленого вивчення алгоритмізації і основ програмування

Інформатика наразі є одним із засобів формування не тільки освітнього, а й розвивального та інтелектуального потенціалу особистості учнів в одній з найперспективніших сфер сучасної діяльності – алгоритмізації та програмуванні. Здобуті компетентності у цій сфері мають відкривати для учнів широкі можливості в навчанні та майбутній професійній діяльності. Тому у процесі поглибленого вивчення алгоритмізації та основ програмування значну увагу варто приділяти розвитку системно-логічного мислення учнів та формуванню складових системи інформатичних компетентностей [69, с. 11; 234; 191, с. 16-25].

Тема «Алгоритмізація та програмування» вивчається в школі протягом кількох етапів. У початковій школі відбувається знайомство з поняттям

алгоритму. Навчальні задачі розглядають як алгоритми на побутову, ігрову, казкову тематику. У середніх класах школи в межах даної теми поняття алгоритму розглядається детальніше, елементи логіки вивчаються на більш формальному рівні. У процесі розв’язування навчальних задач учні знайомляться з різними способами запису алгоритмів, вивчають їх властивості, вчаться будувати, використовуючи LEGO середовище для виконання алгоритму Scratch [93]. У старших класах і особливо в класах фізико-математичного, інформатично-технологічного профілів ця тема вивчається значно ширше. Успіхи учнів в освоєнні навчального матеріалу багато в чому залежать від набутих ними загально-навчальних навичок, що становлять основу системно-логічного мислення і повинні формуватися, починаючи з введення поняття алгоритму в навчальний процес.

Створюючи алгоритм чи програму, учень повинен передбачати їх виконання. Проте часто, через певні помилки і неточності результат не відповідає очікуванням учня. Протиріччя, яке виникло, змушує його аналізувати свою роботу та знаходити помилки. Для розвитку мислення поява «протиріччя» – це сигнал появи проблеми, нерозв’язної за допомогою вже відомих інтелектуальних дій, сигнал для включення мислення як самостійного осмислення предмета [37, с. 21–30.]. Це сприяє стимулюванню мислення учнів та систематизації уже наявних знань з метою досягнення чогось нового. Крім того, різноманітність виконавців алгоритмів і мов програмування спонукає учня чітко формулювати свою думку рідною мовою, а тоді вже записувати її з допомогою мови конкретного виконавця.

Розглядаючи поняття алгоритму потрібно звернути увагу учнів, що ми даємо не строге математичне поняття, а роз’яснення. Проте інколи при розгляді алгоритму як об’єкта математичної теорії потрібно мати саме точне поняття алгоритму, яке відповідало б фактично змістові суті цього поняття. Один із способів введення такого поняття пов'язаний з алгоритмічною системою – машина Тюрінга, в якій поняття алгоритму ґрунтується на командно-адресному принципі.

З попередніх класів учням уже відомо, що під алгоритмом розуміється точне розпорядження, яке задає обчислювальний процес, що починається з довільних початкових даних і спрямоване на отримання повністю визначеного цими вхідними даними результату. Це так зване інтуїтивне поняття алгоритму. Такого поняття алгоритму цілком достатньо, якщо стоїть завдання побудови конкретного алгоритму або доведення алгоритмічного розв'язання конкретної задачі.

Інша справа, якщо треба довести, що алгоритму розв'язування певного класу задач не існує. Тоді і потрібне строге математичне уточнення поняття алгоритму. Поняття алгоритму в його загальному вигляді належить до основних, невизначених понять, тому всілякі уточнення цього поняття призводять, загалом кажучи, до деякого його звуження. Отож, на роль уточнення може претендувати таке уточнення, коли у нас буде переконання в тому, що для кожного алгоритму в інтуїтивному сенсі може бути вказаний еквівалентний йому уточнений алгоритм [172, с. 12].

Описуючи різноманітні алгоритми для машин Тюрінга і стверджуючи реалізованість усіляких композицій алгоритмів, А. Тюрінг переконливо показав розмаїтість можливостей використання запропонованої ним конструкції, що дало змогу йому висунути тезу: «Буть-який алгоритм може бути реалізований відповідною машиною Тюрінга» [191, с. 12–15]. Це основна гіпотеза теорії алгоритмів у формі Тюрінга. Одночасно ця теза є формальним визначенням алгоритму, за умови, що є формальне означення «Машини Тюрінга». Завдяки їй можна доводити існування або неіснування алгоритмів, створюючи відповідні машини Тюрінга, що є загальним підходом до пошуку алгоритмічних розв'язків.

Досвід показує, що для формування складових компетентностей з алгоритмізації та програмування при поглибленому вивченні інформатики в старших класах доцільно розглянути на уроках машину Тюрінга. У багатьох підручниках з інформатики при вивченні поняття і властивостей алгоритму використовують фрази такого змісту: «... Існує багато різних способів для

запису одного і того ж алгоритму, наприклад, запис у вигляді тексту, запис у вигляді блок-схеми, запис якою-небудь алгоритмічною мовою, подання алгоритму у вигляді машини Тюрінга або машини Поста...» [69]. На жаль, фрази, де згадується машина Тюрінга, не є поширеними в ЗОШ. Без сумніву, обсяг годин, що відводяться на вивчення алгоритмів, не завжди дає змогу включати в цю тему вивчення способів запису алгоритму у вигляді машини Тюрінга. Однак ця тема вкрай цікава, важлива і корисна для школярів, які особливо захоплюються інформатикою. Тема «Машина Тюрінга» може вивчатися в 10–11 класах у межах теми «Алгоритм. Виконавці алгоритму», також на гуртковій роботі з обдарованими дітьми, у профільному навчанні інформатики та бути безпосереднім продовженням вивчення формальних виконавців. Складання програм для машини Тюрінга для розв’язування доцільно дібраних задач сприяє розвитку системно-логічного мислення, дає змогу реалізовувати міжпредметні зв’язки (особливо інформатики і математики), формувати складові алгоритмічної культури учнів.

Введення в курс інформатики теми вивчення і застосування машини Тюрінга як універсального виконавця алгоритмів підвищує ефективність засвоєння учнями поняття алгоритму, сприяє формуванню компонентів інформатичних компетентностей, інформаційної культури та розвитку логічного мислення старшокласників.

У процесі навчання учнями здобуваються і закріпляються такі знання, вміння і навички:

− знання про склад машини Тюрінга, принципи її роботи;

− реалізація базових алгоритмічних структур на машині Тюрінга;

− навички побудови машин Тюрінга для розв’язування навчальних задач;

− закріплення і вдосконалення знань та вмінь з алгоритмізації;

− ознайомлення з поняттям алгоритмічно нерозв’язної проблеми.

Усі команди, які може виконувати певний виконавець алгоритму, складають систему команд виконавця (СКВ). Алгоритм будується з команд

СКВ. Об’єкти, над якими виконавець може виконувати дії, складають так зване середовище виконавця.

Виконання алгоритму здійснюється цілком формально. Звідси випливає, що виконавцем алгоритмів може бути автоматичний пристрій. Клас задач, на розв’язання яких орієнтований виконавець, визначається його системою команд. У методиці навчання алгоритмізації прийнято виокремлювати дві категорії виконавців: виконавці, що працюють в певній ситуації, і виконавці, що працюють з величинами. Для першої категорії середовищем виконавця може бути лист (екран), на якому виконавець формує зображення (малюнки, креслення та ін.); лабіринт, який виконавець повинен подолати; предмети, які виконавець повинен розставити в певному порядку і т. д. Виконавці роботи з величинами призначені для опрацювання слів, поданих у певному алфавіті, числових або символьних даних. Виконавець, в систему команд якого входять арифметичні і логічні операції, може розв’язувати обчислювальні задачі. Вхідними даними і результатами для нього є числа. Універсальним виконавцем алгоритмів для роботи з величинами є комп’ютер.

Машина Тюрінга – це абстрактний виконавець (абстрактна обчислювальна машина), яку запропонував Алан Тюрінг в 1936 для формалізації поняття алгоритму. Машина Тюрінга є розширенням скінченого автомата і, згідно з тезою Черча-Тюрінга, здатна імітувати всіх інших виконавців (за допомогою задання правил переходу), що будь-яким чином реалізують процес покрокового обчислення, де кожен крок обчислення достатньо елементарний [11].

Доцільно звернути увагу учнів, що метою створення абстрактної, уявної машини було отримання можливості доведення існування або неіснування алгоритмів розв’язування певних класів задач. Керуючись цією метою, А. Тюрінг шукав якомога простішу, «бідну» алгоритмічну схему, лише б вона була універсальною.

Машина Тюрінга є уявним математичним апаратом, створеним для розв’язування певних класів задач. Цей апарат був названий «машиною» тому,

що опис його складових і функціонування схожий на опис складових і функціонування обчислювальної машини. Принципова відмінність машини Тюрінга від обчислювальних машин полягає в тому, що це суто уявна, абстрактна конструкція, зокрема її блок зовнішньої пам'яті є нескінченною стрічкою, тоді як у реальних обчислювальних машин запам'ятовуючий пристрій може мати дуже великий об'єм, але обов'язково скінченний. Машину Тюрінга не можна реалізувати через нескінченність її стрічки.

До складу машини Тюрінга входить нескінченна в обидві сторони стрічка (можливі машини Тюрінга, які мають кілька нескінченних стрічок), поділена на комірки, і управляючий пристрій, здатний перебувати в одному з безлічі станів – автомат (зчитувальний елемент для зчитування/запису, керований програмою). Число можливих станів управляючого пристрою та зміст цих станів точно задане. Управляючий пристрій може за певними командами переміщатися вліво і вправо по стрічці, читати, аналізувати і записувати в комірки стрічки символи деякого скінченого алфавіту. Серед символів алфавіту є так званий порожній символ, що заповнює всі комірки стрічки, крім тих з них (скінченого числа), в яких записані вхідні дані. Управляючий пристрій працює згідно з правилами переходу, які складають алгоритм, реалізований даною машиною Тюрінга. Кожне правило переходу скеровує машину, залежно від поточного стану і спостережуваного в поточній комірці символу: записати в цю комірку новий символ, перейти в новий стан та переміститися на одну комірку вліво або вправо, або залишитися на місці. Деякі стани машини Тюрінга можуть бути помічені як термінальні і перехід в будь-який з них означає кінець роботи, зупинку алгоритму.

З кожною машиною Тюрінга пов'язані два скінчені алфавіти: алфавіт вхідних сигналів $A = \{a_0, a_1, ..., a_m\}$ і алфавіт станів $Q = \{q_0, q_1, ..., q_p\}$, (з різними машинами Тюрінга можуть бути пов'язані різні алфавіти A і Q). Стан q_0 називається пасивним. Вважається, що коли машина потрапила в цей стан, то вона закінчила свою роботу. Стан q_1 називається початковим. Перебуваючи в цьому стані, машина починає свою роботу. Вхідне слово розміщується на

стрічці по одній букві в розташованих підряд комірках. Ліворуч і праворуч від вхідного слова розташовані тільки порожні комірки (в алфавіт А завжди входить «порожня» буква a_0 – ознака того, що комірка порожня).

Конкретна машина Тюрінга задається описанням елементів множини букв алфавіту А, станів Q, початковою конфігурацією букв з алфавіту А і набором правил, за якими працює машина. Вони мають вигляд: $q_i a_j \rightarrow q_{i1} a_{j1} d_k$ (якщо читаючий елемент перебуває в стані q_i, а в поточній комірці записана буква a_j, то читаючий елемент переходить у стан q_{i1}, в поточну комірку замість a_j записується буква a_{j1}, читаючий елемент робить рух d_k, який має три варіанти: на клітинку вліво (L), на клітинку вправо (R), залишитися на місці (N)). Для кожної можливої конфігурації $<q_i, a_j>$ є тільки одне правило. Випадок $q_{i=}q_1$ визначає початковий стан – розташування зчитувального елемента *i*, тобто визначається яка буква початкової конфігурації зчитується першою. Випадок $q_{i=}q_0$ означає пасивний стан машини Тюрінга, потрапивши в який, машина зупиняється.

Машина Тюрінга є найпростішою обчислювальною машиною з лінійною пам’яттю, яка згідно з заданими формальними правилами перетворює вхідні дані за допомогою послідовності елементарних дій. Елементарність дій полягає в тому, що дія змінює лише невеликий шматочок даних у пам’яті (у разі машини Тюрінга – лише зміст однієї комірки). Незважаючи на простоту машини Тюрінга, на ній уявно можна обчислити все, що можна обчислити на будь-який іншій машині («Машина Поста», «Алгоритми Маркова»), яка здійснює обчислення за допомогою послідовності елементарних дій. Ця властивість називається повнотою.

Машина Тюрінга може виконувати всі можливі перетворення слів, реалізуючи цим всі можливі алгоритми. На ній можна імітувати обчислення «Машини Поста», «алгоритмів Маркова» і будь-яку програму для звичайних комп’ютерів, перетворюючи вхідні дані у вихідні з якого-небудь алгоритму. Відповідно, на різних абстрактних виконавцях можна імітувати машину Тюрінга. Виконавці, для яких це можливо, називаються повними за Тюрінгом.

Багатство можливостей машини Тюрінга проявляється в тому, що коли якісь алгоритми А і В реалізуються машинами Тюрінга, то можна побудувати машини Тюрінга, що реалізують різні композиції алгоритмів А і В. Наприклад: «Виконати А, потім виконати В» або «Виконати А. Якщо в результаті вийшло слово «так», виконати В. В іншому випадку не виконувати В або виконувати почергово А, В, поки виконується певна умова».

Очевидно, що такі композиції також є алгоритмами, тому їх реалізація за допомогою машини Тюрінга підтверджує, що вона є універсальним виконавцем. Варто звернути увагу учнів, що всі алгоритми, придумані людством протягом століть, можуть бути реалізовані машиною Тюрінга. Цей фундаментальний результат був отриманий тоді, коли універсальних обчислювальних машин ще не існувало. Крім того, факт побудови уявного універсального виконавця дозволив висловити припущення про доцільність побудови універсальної обчислювальної машини, яка б могла розв'язувати будь-які задачі за умови відповідного кодування вихідних даних і розробки відповідної програми дій виконавця.

Тема «Машина Тюрінга» може вивчатися в 10–11 класах на уроках інформатики, на факультативних заняттях, гуртках, в системі додаткової освіти, наприклад, в школах юних програмістів. Вивчення цієї теми варто супроводжувати комп'ютерною підтримкою, використовуючи програмний тренажер-імітатор «Машина Тюрінга». Після закінчення вивчення теми учні повинні знати поняття і принципи дії машин Тюрінга, операції над ними, а також в процесі навчання мають навчитися будувати машини Тюрінга для розв'язування навчальних задач.

Під керівництвом вчителя учні дізнаються, як можна реалізувати основні алгоритмічні структури на машині Тюрінга. Зокрема, для реалізації повного розгалуження достатньо ввести два стани q1 і q2, перебуваючи в яких читаючий елемент розглядає комірку з деяким символом. Команда S в стані q1 повинна забезпечити перехід на виконання <серії 1>, а S в стані q2 – на виконання <серії 2>. Аналогічно розглядається команда вибору, циклу.

Команда розгалуження – оператор вибору

	q_1	q_2
S	<серія 1>	<серія 2>

Команда вибору – «case»

	q_1	q_2	…	q_n
S	<серія 1>	<серія 2>	…	<серія n>

Деякі питання щодо алгоритмічно нерозв'язних проблем, універсальної машини Тюрінга доцільно розглянути у процесі роботи гуртка з інформатики. І хоча побудова універсальної машини є нелегкою справою, все ж варто запропонувати учням розробити власні проекти такої машини.

Застосування поняття машини Тюрінга як універсального виконавця алгоритмів на уроках поглибленого вивчення інформатики у старших класах допоможе учням розширити розуміння поняття алгоритму, навчитися описувати принципи роботи цієї машини та будувати машину Тюрінга для розв'язання нескладних задач; формувати ключові інформатичні компетентності, світогляд; сприятиме розвитку системно-логічного мислення і ефективному закріпленню вмінь старшокласників аналізувати, систематизувати та доводити [див. Додаток Г].

2.2. Засоби організації навчально-пізнавальної діяльності

2.2.1. Навчальне середовище «ІнфоНІС»

Із розвитком ІКТ для вдосконалення навчального процесу набули значного поширення новітні засоби його організації. Розглянемо серед них «ІнфоНІС», MOODLE, ATutor, що були створені для розроблення, управління та поширення навчальних матеріалів он-лайн із забезпеченням спільного доступу багатьох користувачів.

Зупинимося детальніше на аналізі навчально-інформаційного середовища (НІС) «ІнфоНІС». Його розробник – С. О. Лещук, науковий керівник – Ю. С. Рамський [138, с. 116-152]. Застосування цього засобу у навчальному процесі ЗОШ сприяє розкриттю потенціалу кожного учня відповідно до його нахилів, запитів і здібностей, а також педагогів, зокрема їх професійних навичок як у школі, так і у віртуальних учительських центрах.

Використання «ІнфоНІС» у навчанні учнів дає змогу уникнути деякі недоліки «традиційного» навчання [192, с. 120–125]:

− недостатньо якісно проводиться контроль навчальної діяльності учнів, тобто відсутній надійний зворотний зв’язок;

− всі учні навчаються в одному темпі, що сповільнює розвиток багатьох учнів, «усереднюючи» талановитих і «втрачаючи» тих, хто не може вчитися без допомоги;

− важко забезпечити об’єктивність в оцінюванні знань учня, здійснювати постійне спостереження за його роботою;

− великі обсяги неструктурованого навчального матеріалу;

− учні часто не вміють вести пошукову чи творчу роботу наодинці.

«ІнфоНІС» може використовуватись на уроках різного типу для:

− подання навчального матеріалу;

− диференціації завдань стосовно навчальних успіхів учня;

− організації індивідуальної пізнавальної діяльності;

− проведення поточного та підсумкового контролю;

− доповнення навчального матеріалу широким спектром додаткових та довідкових даних;

− спрощення організаційної роботи, яка супроводжує навчальний процес.

Рис. 2.4. Початкова сторінка навчально-інформаційного середовища

Розробники «ІнфоНІС» реалізували ідею створення умов управління індивідуальною роботою кожного учня в умовах класно-урочної системи навчання, активізації пізнавальної діяльності, використання якої сприяє розвитку системно-логічного типу мислення та підвищення ефективності навчального процесу.

У розробці інтерфейсу та засобів «ІнфоНІС» врахованно особливості усіх типів дітей (візуальні, кінетичні, аудіальні). Використовуються також такі особливості ППЗ, як гіпертекстовість, мультимедійність, інтегрованість (поєднання теоретичних матеріалів, практичних завдань, запитань, тестів, програмних середовищ), реалізована можливість інтерактивного навчання.

З елементів «ІнфоНІС» – мультимедіа, що є формою організації навчального матеріалу. Принцип наочності забезпечується завдяки використанню звукового супроводу, анімації [136, с. 122].

Рис. 2.5. Схема «ІнфоНІС»

Пропонуємо використовувати «ІнфоНІС» в старших класах при поглибленому вивченні алгоритмізації та програмування, коли учні вже мають значний загальноосвітній рівень підготовки, в них сформовані певні інтереси, сталі уявлення про роль комп'ютерів у сучасному суспільстві. Кожна навчальна програма має свої особливості, спрямовані на розвиток тих чи інших якостей школяра. Інтегруючи різні властивості (багатогранна подача матеріалу, контроль, індивідуальний вибір учня, тощо) в «ІнфоНІС», можна забезпечити навчання широкого кола учасників навчального процесу, зацікавити учнів, нестандартно поєднувати індивідуальне і групове навчання, що сприяє формуванню інформатичних компетентностей. Використання «ІнфоНІС» в навчальному процесі дає можливість реалізувати принципи індивідуального підходу у навчанні учнів старших класів. Щоб забезпечити індивідуальне навчання, розроблено відповідні технології (рис. 2.6).

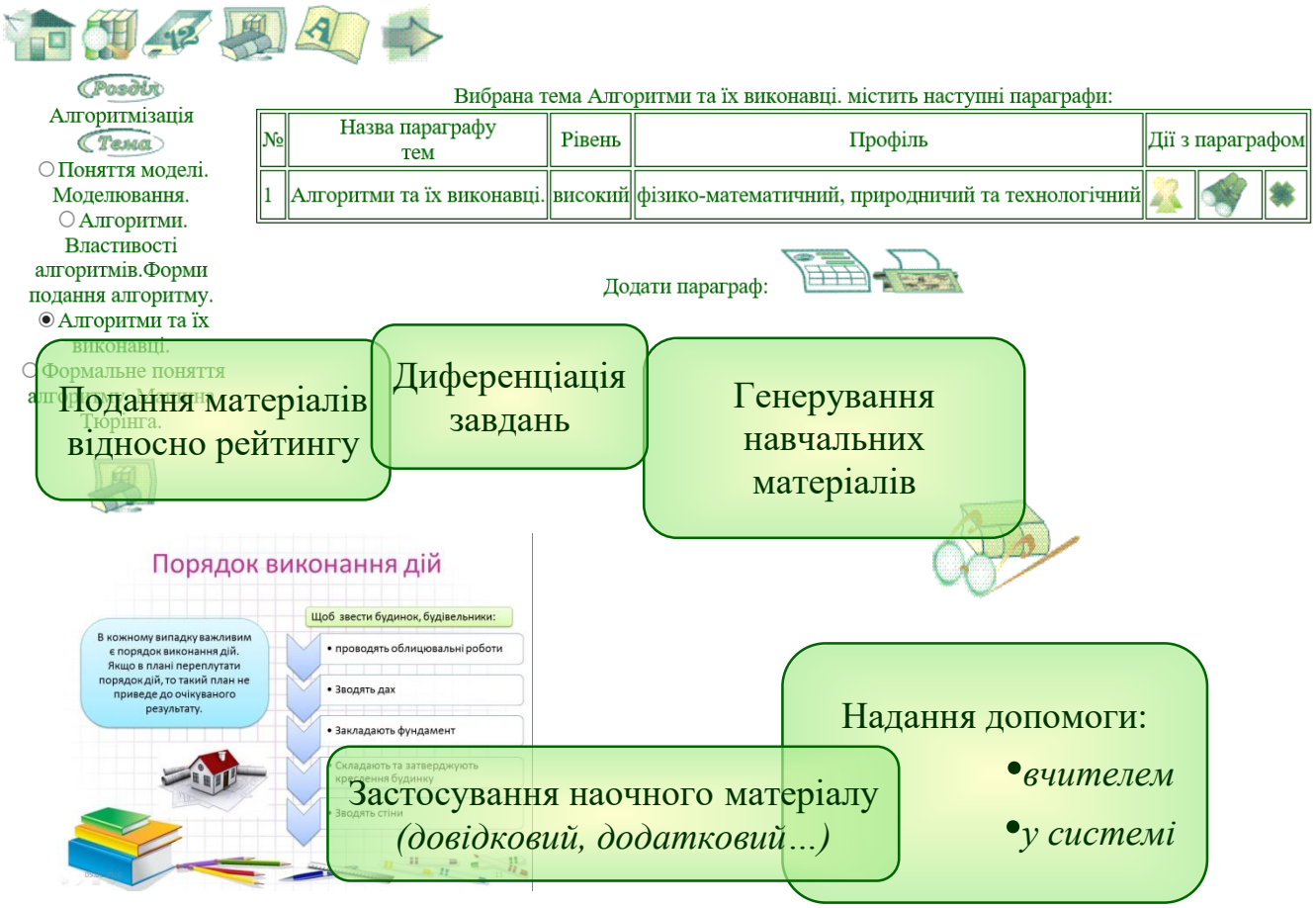


Рис. 2.6. Організація індивідуального навчання

Для створення умов об’єктивного оцінювання навчальних досягнень учнів при вивченні алгоритмізації та програмування, в «ІнфоНІС» динамічно наповнюється база навчальними матеріалами: вправами, тестами, запитаннями (рис. 2.7), які генеруються на комп’ютер учня випадково з урахуванням теми, профілю, рівня навчальних досягнень (початковий, середній, достатній, високий).

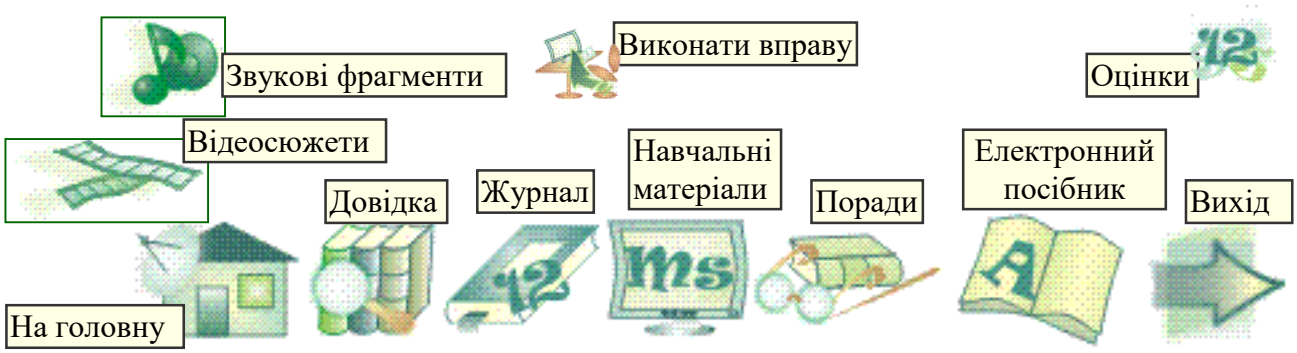


Рис. 2.7. Елементи інтерфейсу

Кожен учень працює на рівні своїх можливостей і успішно виконує навчальні завдання, що налаштовує його на ефективне навчання.

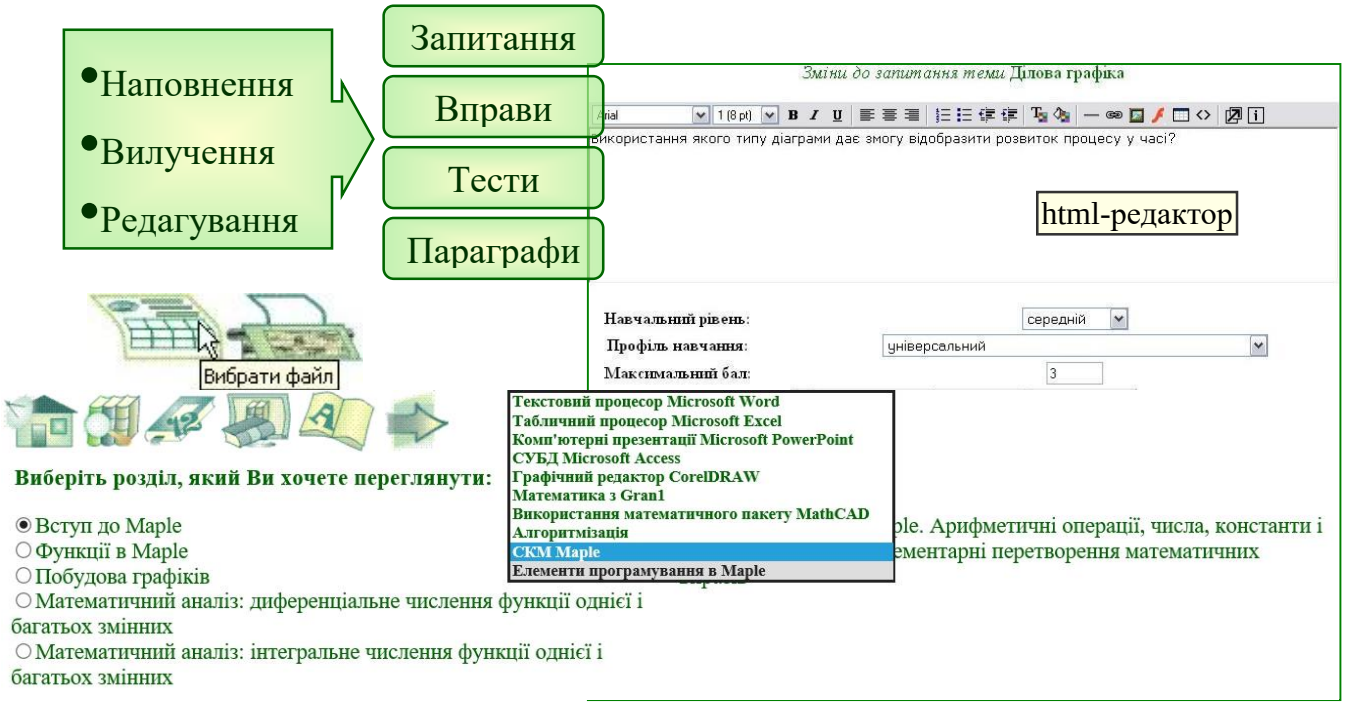


Рис. 2.8. Наповнення навчальними матеріалами

Наповнюється середовище навчальним матеріалом двома способами: прикріпленням розробленої html-сторінки або її створенням у середовищі вбудованого html-редактора (рис. 2.8).

Використовуючи «ІнфоНІС», є можливість залучати учнів до розробки навчальних матеріалів, що сприяє розвитку їх активності у навчанні, відповідальності за доручену справу, вмінь самостійно працювати тощо.

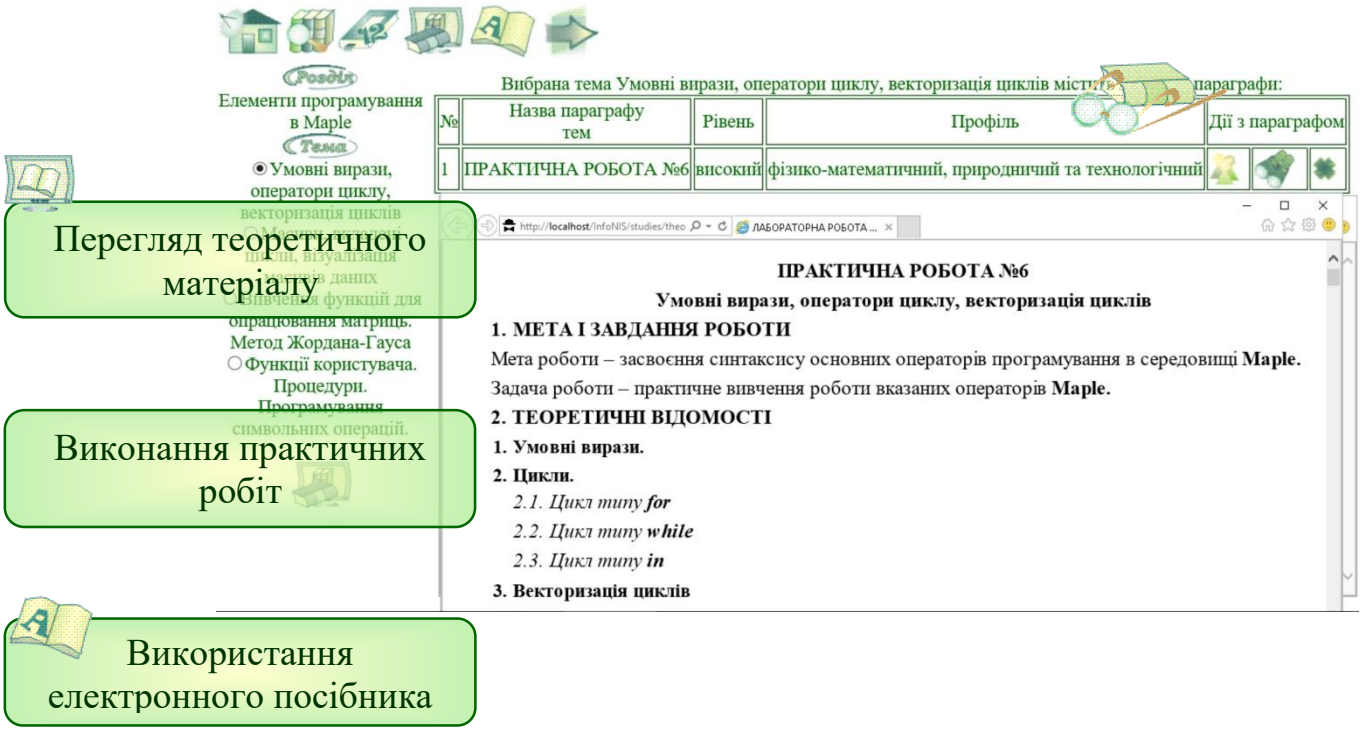


Рис.2.9. Подання навчального матеріалу

На рис. 2.9 відображенні основні аспекти подання навчального матеріалу з алгоритмізації та програмування, яке здійснюється відповідно до планування. Для кожного рівня характерна завершеність та цілісність.

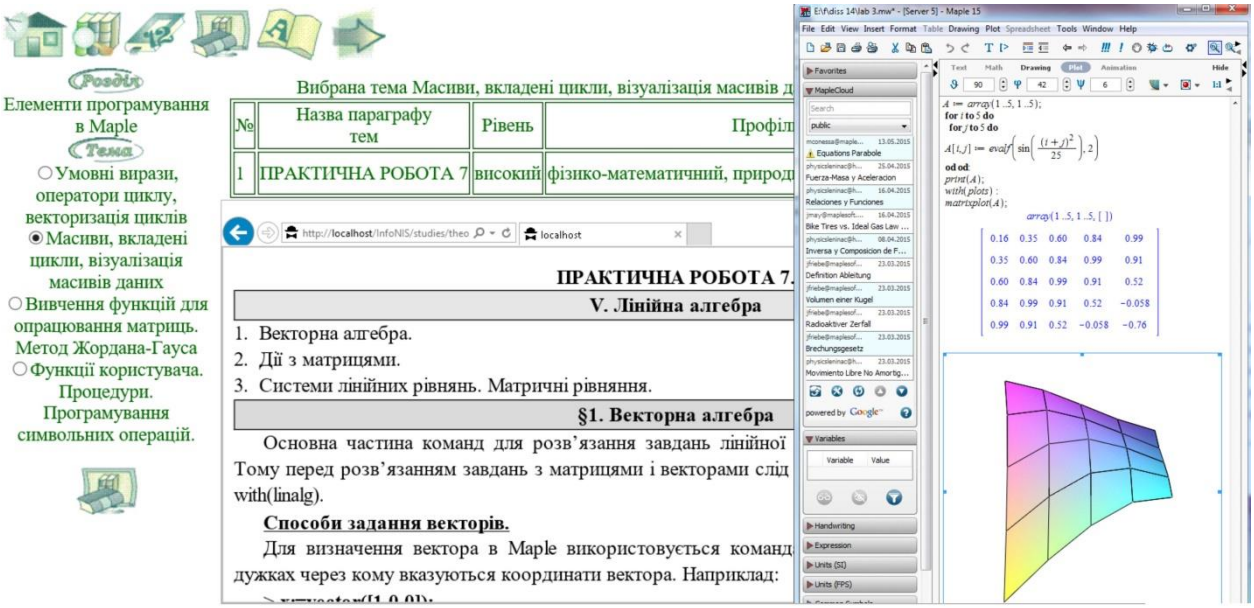


Рис. 2.10. Виконання практичної роботи

На рис. 2.10 наведено приклад виконання практичної роботи. Запуск програми Maple відбувається над НІС. Робота кожного учня над індивідуальним завданням захищена індивідуальним профілем. Для перевірки вчителем файли-відповіді учень відсилає у базу даних. За результатами перевірки вчитель виставляє відповідну оцінку в електронний журнал.

Таким чином, власний досвід застосування «ІнфоНІС» дозволяє зробити такі висновки. Використання цього НІС у старшій школі при навчанні алгоритмізації та програмування дає змогу забезпечити диференціацію завдань, різнотипне подання навчальних матеріалів, індивідуальну роботу старшокласника, проведення поточного та підсумкового контролю, зручний доступ до додаткового, довідкового матеріалу.

Організація навчального процесу з використанням «ІнфоНІС» сприяє розвитку системно-логічного мислення старшокласників, формує в них ставлення до засобів ІКТ як до інструментів пізнання, сприяє формуванню інформатичних компетентностей учнів і тим самим підвищенню ефективності навчального процесу.

Як показує експериментальне дослідження, використання «ІнфоНІС» в навчальному процесі ЗОШ при вивченні інформатики на поглибленому рівні навчання учнів, що мають вже достатньо сформований пізнавальний та профорієнтаційний інтерес до дисциплін інформатичного циклу, надзвичайно ефективне. Крім того, НІС просте у користуванні.

2.2.2. Системи управління навчальними ресурсами

Наразі, відповідно до загальносвітової тенденції, в системі освіти все більшу роль починають відігравати системи управління навчальними ресурсами. Активізації цього процесу насамперед сприяє розвиток Інтернету і WEB-технологій, які створили зовсім нові можливості для організації навчання.

На сучасному етапі розвитку ІКТ стали невід'ємною складовою процесу навчання. У навчальних закладах усіх рівнів розроблено велику кількість курсів, орієнтованих на їх використання в навчанні. Розглянемо основні вимоги до використання систем управління навчальними ресурсами та проаналізуємо найбільш використовувані з них.

Успішне впровадження ІКТ ґрунтується на правильному виборі програмного забезпечення, що відповідає конкретним вимогам. Ці вимоги визначаються потребами учня, вчителя та адміністратора; останній повинен контролювати встановлення, налаштування програмного забезпечення і результати навчання.

До засобів організації навчання з використанням ІКТ можна висунути певні вимоги.

− ***Функціональність:*** наявність у системі набору функцій різного рівня: форуми, чати, аналіз активності учнів, управління курсами і діяльність учнів тощо.

− ***Надійність:*** характеризує зручність і простоту адміністрування, оновлення контенту на базі існуючих шаблонів. Зручність управління і захист від зовнішніх впливів суттєво впливають на ставлення користувачів до системи і ефективності її використання.

− ***Стабільність*** вказує на ступінь стійкості роботи системи щодо різних режимів роботи і ступеня активності користувачів.

− ***Вартість*** складається з вартості системи, а також з витрат на її впровадження, розробку курсів і супровід, наявність або відсутність обмежень з кількості ліцензій на учнів (слухачів).

− ***Наявність засобів розробки контенту.*** Вбудований редактор навчального контенту не тільки полегшує розробку курсів, а й дає змогу інтегрувати в єдиному поданні освітні матеріали різного призначення.

− ***Підтримка SCORM.*** Стандарт SCORM *(Sharable Content Object Reference Model)* є міжнародною основою обміну електронними курсами і

відсутність в системі його підтримки знижує мобільність, не дає змоги створювати портативні курси.

− ***Система перевірки знань.*** Дає змогу в режимі *on-line* оцінити знання учнів. Зазвичай, така система включає в себе тести, завдання і контроль активності учнів на форумах.

− ***Зручність використання.*** При виборі нової системи необхідно забезпечити зручність її використання. Це важливий параметр, оскільки потенційні учні не будуть використовувати технологію, яка здається надто складною або створює труднощі при навігації. Технологія навчання повинна бути інтуїтивно зрозумілою. У навчальному курсі має бути просто знайти меню допомоги, не повинно бути важко переходити від одного розділу до іншого і спілкуватися з інструктором.

− ***Модульність.*** В сучасних системах ІКТ курс навчання може бути набором мікромодулів або блоків навчального матеріалу, які можна використати в інших курсах.

− ***Забезпечення доступу.*** Учні не повинні мати перешкод у доступі до навчальної програми, пов’язаних з їх розташуванням, а також з факторами, що обмежують можливості школярів (обмежені функції організму, ослаблений зір). Також використання технологій «завтрашнього дня», які підтримуються обмеженим колом програмного забезпечення, істотно знижує круг потенційних користувачів.

− ***100% мультимедійність.*** Можливість використання як навчального матеріалу не лише текстових, гіпертекстових і графічних файлів, але і аудіо, відео, gif і flash-анімації, 3D-графіки різних файлових форматів.

− ***Масштабованість і розширюваність.*** Можливість розширення кола слухачів та додавання нових програм та вдосконалення навчальних курсів.

− ***Перспективи розвитку платформи.*** Системою управління навчальними курсами доцільно використовувати середовище, яке постійно оновлюється, створюються більш ефективні версії системи з підтримкою нових технологій, стандартів і засобів.

− ***Крос-платформеність***. В ідеалі такі системи не повинні бути прив'язані до будь-якої операційної системи або середовища, як на серверному рівні, так і на рівні клієнтських машин. Користувачі можуть використовувати стандартні засоби без завантаження додаткових модулів, програм і т.д.

− ***Якість технічної підтримки.*** Можливість функціональної підтримки, стабільності систем, усунення помилок і недоліків із залученням фахівців компаній розробників та якісної підтримки на рівні адміністрування.

− ***Наявність (відсутність) вітчизняної локалізації продукту.*** Локалізована версія продукту більш зручна як для адміністрування, розробки курсів, так і для кінцевих користувачів освітніх послуг.

Варто зазначити, що, крім цих вимог, ефективність використання систем управління навчальними ресурсами істотно залежить від використовуваних в них технологій. Можливості і характеристики технології навчання з використанням ІКТ забезпечують максимально можливу ефективність взаємодії учня і вчителя в межах системи. Складне у використанні програмне забезпечення не тільки ускладнює сприйняття навчального матеріалу, але і викликає певне неприйняття використання інформаційних технологій у навчанні.

Програмне забезпечення для навчання з використанням ІКТ подане як простими статичними HTML-сторінками, так і складними системами управління навчанням і навчальним контентом (Learning Content Management Systems – LCMS), що використовуються в корпоративних комп'ютерних мережах. Розглянемо системи управління навчанням (Learning Management Systems – LMS).

Використання ІКТ, як і будь-який навчальний процес, крім змістової частини, обов'язково включає організаційний компонент. Елементи управління процесом проходження курсів наявні в розвинених електронних бібліотеках, але для реалізації великої системи e-Learning (навчання з використанням ІКТ) цієї функціональності буде недостатньо. Знадобиться

автоматизація таких завдань, як надання навчального контенту учням в конкретний час, контроль використання навчальних ресурсів, адміністрування окремих слухачів і груп, організація взаємодії з вчителем, звітність тощо. Ці функції реалізують системи управління навчанням LMS, які є платформою для розгортання e-Learning, але в багатьох випадках можуть використовуватися і для адміністрування традиційного навчального процесу.

Система LMS в ідеалі повинна надавати кожному учневі персональні можливості для найефективнішого вивчення матеріалу, а менеджеру навчального процесу (яким у школі є вчитель) – необхідні інструменти для формування навчальних програм, контролю їх проходження, складання звітів про результативність навчання, організації комунікацій між учнями і педагогами. Школяр отримує від LMS можливості доступу до навчального порталу, який є відправною точкою для доставки навчального контенту, вибору відповідних навчальних треків на основі попереднього і проміжних тестувань, використання додаткових матеріалів за допомогою спеціальних посилань.

Адміністративні функції LMS охоплюють кілька базових галузей. Управління учнями включає в себе завдання реєстрації та контролю доступу користувачів до системи і до навчального матеріалу, організацію учнів в групи для надання їм загальних курсів і складання звітності, управління навчальними ресурсами. В LMS також можлива інтеграція додаткових елементів навчального процесу (практичні заняття, лабораторні роботи, тести, засоби спільної роботи, посилання на зовнішні матеріали та ін.).

Крім того, в LMS зручно розподіляти і використовувати навчальний контент. Серед таких завдань – організація зручних для пошуку каталогів курсів, виокремлення груп курсів для обов’язкового вивчення і вивчення «за бажанням», розробка індивідуальних навчальних треків (наприклад, на базі заданих функціональних ролей слухачів), інші механізми цільового надання навчального контенту, підтримка синхронних і асинхронних режимів зв’язку з учителем. Найважливішим елементом LMS є звітність навчального процесу,

використання якої дає змогу робити висновки про ефективність застосування системи управління навчальним курсом. В LMS повинні бути механізми контролю і складання звітів про те, наскільки успішно просувається учень (чи клас – модель «колективний учень») у вивченні певних тем, чи відповідає підвищення рівня професійної кваліфікації в результаті навчання заданій на початку навчання меті, наскільки отримані знання застосовуються в практичній роботі і як впливають на її результативність.

Серед найважливіших функцій LMS в процесі навчання можна виокремити такі:

– Підтримка змішаного навчання. LMS надає можливості просто та легко поєднувати традиційне (аудиторне) навчання в навчальних класах та віртуальне навчання на основі мережевих навчальних курсів. У комбінації ці можливості активізують і звичайне, і персоналізоване (персоніфіковане) навчання.

– Інструменти адміністрування. LMS надає поле для діяльності адміністраторів з управління реєстрацією користувачів і профілів, призначення тьюторів, авторів курсів, управління навчальним матеріалом. Адміністратори мають повний доступ до бази даних навчання, можливість створювати стандартні звіти за індивідуальним і груповими показниками. За необхідності звіти можна змінювати аж до можливості включення всього учнівського та педагогічного колективів. Використання системи дає змогу складати розклад для учнів, інструкторів та навчальних класів. За можливості управління доцільно здійснювати через автоматизований «доброзичливий» інтерфейс.

– Інтеграція контенту. Дуже важливо для LMS забезпечувати підтримку широкого кола курсів від сторонніх виробників. Деякі LMS сумісні з інструментом розробки тільки власного виробництва, а інші дуже обмежено сумісні зі стандартами навчального контенту. Контент LMS потрібно сертифікувати, захищаючи, права інших авторів. Доступ до курсів має бути таким же простим, як використання контекстного меню.

– **Дотримання стандартів.** LMS повинна підтримувати стандарти (SCORM та ін.). Підтримка стандартів означає, що LMS може імпортувати і управляти контентом і курсами, які склали відповідно до стандартів, незалежно від засобів розробки, які були використані. Якщо постачальник не сертифікує контент, то неминучі додаткові витрати на сертифікацію останнього.

– Можливості тестування. Обов'язкова наявність модулів оцінювання та тестування. При цьому найбільш ефективний підхід, коли: а) надається можливість включення тесту (модуля оцінювання) як частини кожного розділу курсу (мережевого уроку); б) самостійний модуль тестування (і модуль оцінювання), наприклад, за результатами вивчення окремого розділу та/або курсу загалом.

– Визначення рівня знань. Модуль контролю знань дає змогу навчальному закладу визначити потребу в навчанні та ідентифікувати сферу докладання зусиль, які базуються на компетентності педагогічного колективу в конкретній галузі. Оцінка знань може бути отримана з різних джерел, включаючи співбесіди і метод 360 градусів. Адміністратори визначають: врівноважувати, усереднювати чи порівнювати результати для визначення рівня знань. Також LMS забезпечує ще й механізми захисту, необхідні для мережевого середовища навчання з використанням ІКТ.

Щоби використання LMS-платформи дало можливість «програвати» різні готові курси, створені стандарти *інтероперабельності*. Так, Airline Industry CBT Committee описує взаємодію комп'ютерних тренінгів з системами управління і є основою для розвитку аналогічних стандартів інтероперабельності для WEB-курсів. Широко відомі стандарти ISM для платформ навчання, а також Sharable Content Object Reference Model (SCORM) – сукупність технічних специфікацій для створення навчального WEB-контенту, розроблених у межах програми Advanced Distributed Learning міністерства оборони США. Традиційними лідерами західного ринку LMS є рішення компаній Saba Software, Docent, WBT Systems, Click2Learn, IBM та ін.

Інтернет-навчання розрізняється за формою проходження курсів слухачами: *online* та *ofline* навчання. І однією з ключових проблем інтернет-навчання обох форм є проблема аутентифікації користувача при перевірці знань. Оскільки досі не запропоновано придатних технологічних рішень, більшість програм використання системи управління навчальними курсами, як і раніше, передбачають очну екзаменаційну сесію.

У системах *online* навчання традиційно використовуються дві домінуючі моделі навчання: *презентації* і *програмовані курси*.

Презентаційна модель використовує діапазон технологій від презентацій в PowerPoint до потокового аудіо та відео, які можуть доставлятися користувачеві на базі різних платформ. До недавнього часу широко використовувалися тільки односторонні презентаційні моделі, зараз же спостерігається розширення технічних можливостей: нові системи дають змогу підтримувати і діалогові відео-конференції.

Програмовані навчальні курси є найбільш популярним методом асинхронного навчання. Технічно розробник «розрізає» утримання курсу на керовані уривки тексту (можливо, доповнені аудіо/відеокліпами і графікою), вводиться послідовність інструкцій, супроводжувана відповідями на найбільш часті питання (FAQ) і забезпечується зворотній зв’язок з навчальним центром. З недавнього часу за рахунок використання IP-платформ нового покоління забезпечується можливість взаємодії з іншими учнями або педагогами.

Крім того, в процесі використання систем управління навчальним курсом використовуються так звані змішані моделі навчання, які дають змогу комбінувати власне *online* навчання з практичними і аудиторними заняттями.

Найбільш поширений спосіб створення системи управління навчальним курсом тривалий час полягав у тому, щоби перекласти навчальні матеріали у HTML-форму та розмістити їх на веб-сайтах навчальних закладів. Зараз всі учасники ринку згодні з тим, що одного лише доступу до навчального матеріалу через інтернет недостатньо, щоб говорити про повноцінну навчальну систему. Очевидно, що навчання передбачає не лише читання

навчального матеріалу, а й активне осмислення та застосування отриманих знань на практиці.

Нині великого розповсюдження набувають хмарні технології, які визначаються як динамічно масштабований вільний спосіб доступу до зовнішніх обчислювальних інформаційних ресурсів у вигляді сервісів, що надаються за допомогою інтернету. Пояснюють це передусім новими можливостями для подання динамічних і актуальних, що базуються на інтернет-технологіях, електронних додатків для освіти. Основні компанії, а саме Google, Microsoft, IBM, що займаються розробкою такої продукції, намагаються удосконалити хмарні технології для їх впровадження у навчальний процес ЗНЗ.

Як відомо, «активність» осмислення передбачає можливість задати додаткові і уточнюючі запитання вчителю, отже, таку можливість має забезпечувати і система управління навчальними ресурсами, у тому числі за рахунок форми побудови матеріалу, який повинен як би «провокувати» запитання. При цьому синхронний навчальний курс має бути розрахований на надання відповідей у режимі реального часу, а асинхронний – на максимальну оперативність педагога.

Практичне застосування знань може бути реалізоване у вигляді проходження тестів або виконання більш складних завдань. В обох випадках результати виконання тесту чи завдання повинні бути перевірені або автоматично, або безпосередньо вчителем.

Online навчальний курс, на відміну від презентації або сайту, не тільки забезпечує доступ до інформаційних джерел, а й передбачає інтерактивну взаємодію слухача з учителем, контроль отриманих знань і накопичення даних про процес навчання. Статистика за результатами процесу навчання є важливою складовою системи управління навчальним курсом, оскільки дає змогу вчителям контролювати активність учнів і сам навчальний процес.

Команда з розробки навчального курсу, як правило, включає три групи фахівців:

– фахівці в предметній галузі – носії знань з навчального курсу, який переводиться в *online* форму;

– фахівці з перекладу матеріалів навчального курсу в *online* форму;

– фахівці з підтримки системи управління навчальним курсом.

На основі аналізу існуючих систем LMS виокремимо такі: ATutor, MOODLE, Claroline, Dokeos, LAMS, OLAT, OpenACS, Sakai. Основними критеріями відбору обрані ступінь підтримки системи і багатомовний супровід.

Claroline (http://www.claroline.net/) (Classroom Online) – платформа побудови сайтів навчання з використанням ІКТ, створена з урахуванням побажань викладачів. Додаток створено в інституті педагогіки і мультимедіа Католицького університету в Левені (Бельгія). Продукт безкоштовний і доступний. Він може обслуговувати до 20000 учнів. Claroline дає змогу створювати уроки, редагувати їх зміст, керувати процесом їхнього використання. Додаток включає генератор вікторин, форуми, календар, функцію розмежування доступу до документів, каталог посилань, систему контролю за успіхами учня, модуль авторизації.

Dokeos (http://www.dokeos.com/) – платформа побудови сайтів елктронного навчання, заснована на гілці (fork) Claroline (версії 1.4.2). Гілка є клоном вільно розповсюджуваного програмного продукту, створеного з метою змінити додаток-оригінал в тому чи іншому напрямку. Dokeos – результат роботи деяких членів первісної команди розробників Claroline, які задумали змінити орієнтацію програми. Однак тепер він підійде скоріше організаціям, ніж навчальним закладам. Справа в тому, що Claroline чудово адаптований для НІС, що виражається в підтримці великої кількості учнів і курсів. Dokeos, на нашу думку, більше орієнтований на професійну клієнтуру, наприклад, персонал підприємства. Dokeos безкоштовний і залишиться таким, оскільки ліцензія Claroline (GNU/GPL) передбачає, що гілки підпадають під ту ж ліцензію. Оскільки гілка була виокремлена нещодавно, обидва додатки

схожі один на одного, хоча деякі відмінності в ергономіці, побудові інтерфейсу, функціональності вже починають проявлятися.

LAMS (http://www.lamscommunity.org). Специфікація IMS Learning Design була підготовлена в 2003 році. В її основу покладені результати роботи Відкритого університету Нідерландів (Open University of the Netherlands) з мови освітнього моделювання «Educational Modelling Language» (EML), за допомогою якої описується «метамодель» розробки навчального процесу. На основі цієї специфікації створена «Система управління послідовністю навчальних дій» Learning Activity Management System (LAMS). LAMS надає викладачам візуальні засоби для розробки структури навчального процесу, що дає змогу задавати послідовність видів навчальної діяльності. Вона надає вчителю інтуїтивно зрозумілий інтерфейс для створення освітнього контенту, який може включати різні індивідуальні завдання, завдання для групової роботи та фронтальної роботи з групою учнів.

OLAT (http://www.olat.org). Розробка системи почалася в 1999 році в Цюріхському університеті в Швецарії (University of Zurich, Switzerland), де вона стала основною освітньою платформою навчання з використанням ІКТ .

OpenACS (http://openacs.org) (Open Architecture Community System) – це система для розробки масштабованих освітніх ресурсів. Вона є основою в роботі багатьох компаній і навчальних закладів, що займаються використанням технологій навчання з використанням ІКТ .

Sakai (http://sakaiproject.org/) є системою *online*-організації навчального освітнього простору. Це система з повністю відкритим вихідним кодом, яка підтримується спільнотою розробників. У Sakai інтегрована підтримка стандартів та специфікацій IMS Common Cartridge, SCORM.

Проаналізуємо особливості застосування в навчальному процесі таких систем, як MOODLE і ATutor, що користуються високою популярністю у багатьох вітчизняних ЗНЗ і ВНЗ, зокрема в ЗОШ.

MOODLE (Modular Object-Oriented Dynamic Learning Environment) (http://MOODLE.org/) – це середовище навчання з використанням ІКТ,

призначене для створення якісних навчальних курсів (рис. 2.11). Цей програмний продукт використовується у понад 100 країнах світу університетами, школами, компаніями і незалежними викладачами. За своїми функціями MOODLE витримує порівняння з відомими комерційними системами управління навчальним процесом, і водночас вигідно вирізняється з-посеред них тим, що поширюється у відкритих вихідних кодах: це дає можливість підлаштувати його під особливості кожного освітнього проекту, доповнити новими сервісами.

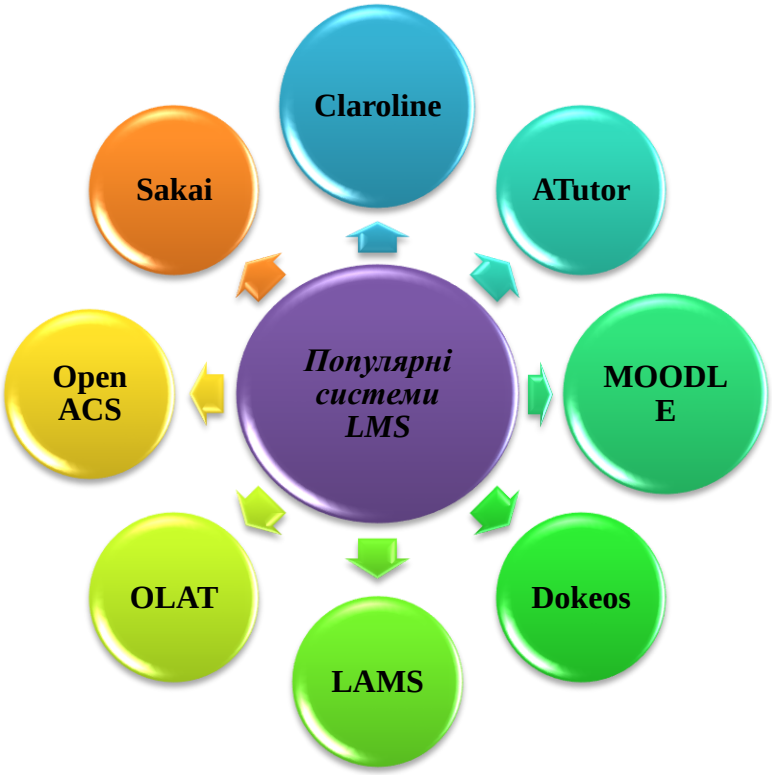


Рис. 2.11. Поширені системи LMS

Переваги MOODLE:

– поширюється у відкритому вихідному коді: можливість «налаштування» під особливості конкретного освітнього проекту, розробки додаткових модулів, інтеграції з іншими системами;

– дає змогу організувати навчання в активній формі, в процесі спільного вирішення навчальних завдань, взаємообміну знаннями;

− широкі можливості для комунікації: обмін файлами будь-яких форматів, розсилка, форум, чат, можливість рецензувати роботи учнів, внутрішня пошта та ін.;

− можливість використовувати будь-яку систему оцінювання (бальну, словесну);

− повні відомості про роботу учнів (активність, час і зміст навчальної роботи, портфоліо);

− відповідає розробленим стандартам і надає можливість вносити зміни без тотального перепрограмування;

− програмні інтерфейси забезпечують можливість роботи людям різного освітнього рівня, різних фізичних можливостей (включаючи з особливими потребами), різних культур.

У системі MOODLE існує три типи форматів курсів: форум, структура (навчальні модулі без прив'язки до календаря), календар (навчальні модулі з прив'язкою до календаря). Курс може містити довільну кількість ресурсів (веб-сторінки, книги, посилання на файли, каталоги) та довільну кількість інтерактивних елементів курсу, до таких елементів належать:

– Wiki, який дає змогу створювати документ кільком людям відразу з допомогою простої мови розмітки прямо у вікні браузера, тобто за його допомогою учні можуть працювати разом, додаючи, розширюючи і змінюючи зміст. Попередні версії документа не видаляються і можуть бути в будь-який момент поновлені.

– Анкети. Цей елемент надає кілька способів обстеження, які можуть бути корисними при оцінюванні і стимулюванні навчання в дистанційних курсах.

– Глосарій. За його допомогою створюється основний словник понять, що використовуються програмою, а також словник основних термінів кожної теми.

– Завдання дають змогу вчителеві ставити задачу, яка вимагає від учнів підготувати відповідь в електронному вигляді (у будь-якому форматі) і завантажити її на сервер.

– Опитування: одне з його застосувань – проводити голосування серед учнів. Це може бути корисним для швидкого опитування, стимулювання мислення чи пошуку спільної думки в процесі дослідження проблеми.

– Пояснення: дає змогу розміщувати текст і графіку на головній сторінці курсу. За допомогою такого напису можна пояснити призначення будь-якої теми чи використовуваного інструмента.

– Тести: цей елемент дає змогу вчителеві створити набір тестових питань.

Використовуючи MOODLE, вчитель подає навчальний матеріал в цікавій та гнучкій формі, що складається з набору сторінок. Кожна сторінка, зазвичай, закінчується запитанням, на яке учень повинен відповісти. Залежно від правильності відповіді учень переходить на наступну сторінку або повертається на попередню.

Варіюючи поєднання різних елементів курсу, викладач організує вивчення матеріалу так, щоб форми навчання відповідали цілям і задачам конкретних занять.

Як уже зазначалося, можна зустріти велику кількість різноманітних продуктів, призначених для організації навчання, в процесі якого немає необхідності витрачати час на переміщення до місця занять, а урок можна почитати в будь-який зручний для себе час, вивчивши весь матеріал відразу або відклавши прочитання частини на потім.

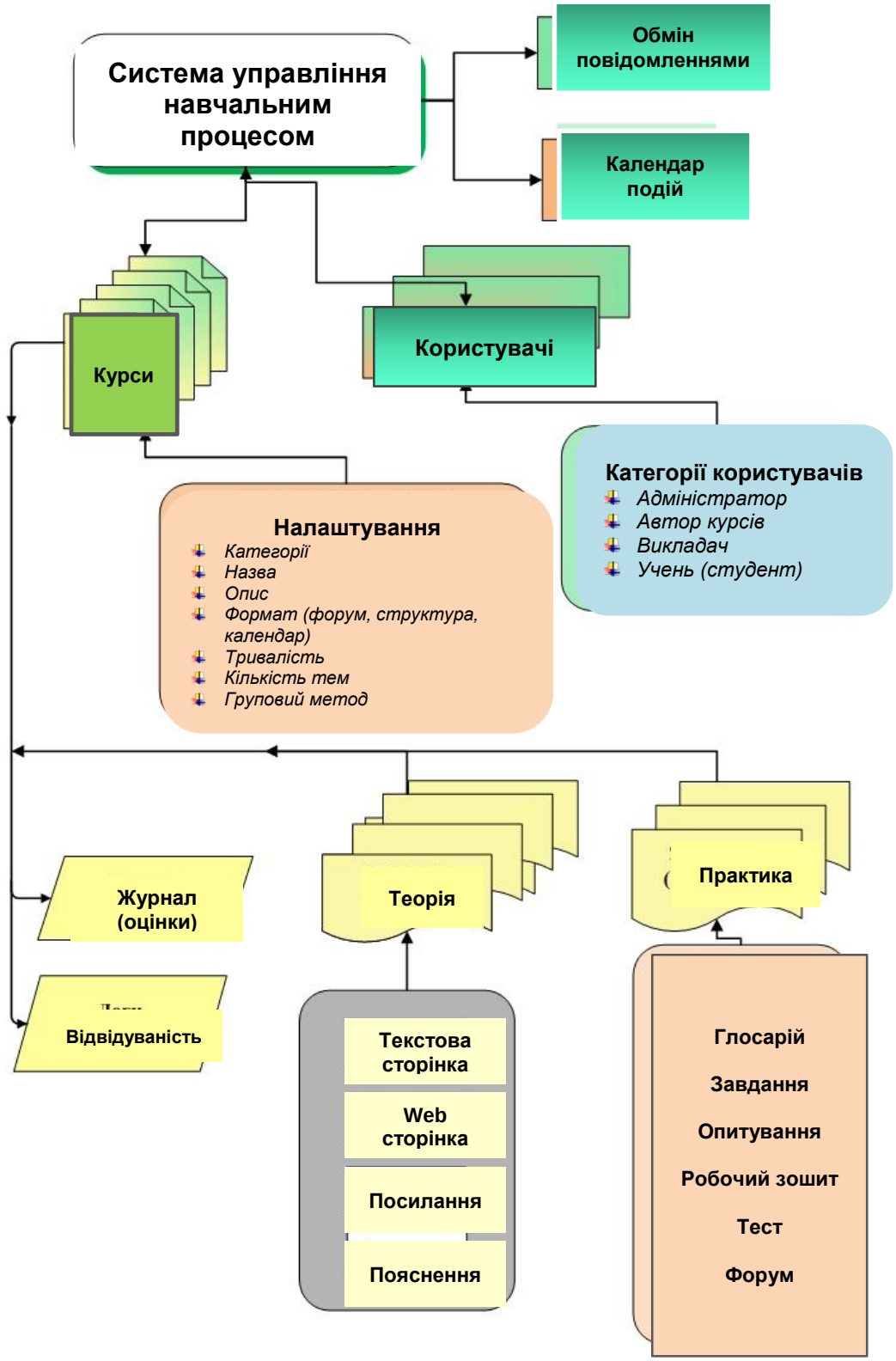


Рис. 2.12. Модель системи управління навчальним процесом MOODLE

Далі піде мова про ще один з них – ***ATutor***. Це система управління навчанням – WEB-based Learning Content Management System (LCMS). Використання ATutor дає змогу вчителям легко організовувати різні курси

навчання. Школярі знайомляться з адаптивним та простим середовищем навчання. Адміністрування нової системи зручне і доступне. Зовнішній вигляд можна легко змінити, доступність вихідного коду і відкриті інструменти, що застосовуються для побудови сервера курсів, дають змогу за крайньої необхідності внести більш серйозні зміни. Крім всього необхідного для створення та управління курсами і процесом навчання, в його складі є засоби обміну повідомленнями. Особлива увага приділяється безпеці. За допомогою додаткових модулів можна наростити функціональність. Вибір останніх достатньо широкий: від забезпечення оплати до роботи з фото, обміну повідомленнями з іншими навчальними системами, конференції тощо (рис. 2.13).

Крім того, розробниками був узятий курс на підтримку продуктом стандартів W3C XHTML 1.0, що дало змогу б у майбутньому легко інтегрувати різноманітний контент. Різноманітні додатки доступні на сайті проекту. Так, ATutor – це перша LCMS, в якій повністю реалізована специфікація доступності WCAG (WEB Content Accessibility Guidelines. Згідно з цими рекомендаціями ресурс доступний і для користувачів з різними вадами здоров'я. Щоб мати можливість використовувати курси, створені для інших e-learning навчальних систем, система підтримує специфікації IMS (Instructional Management Standards; www.imsproject.org) і SCORM (Sharable Content Object Reference Model; www.adlnet.org). Зараз основним джерелом сторонніх курсів для користувачів ATutor є TILE learning objects respository (tile.atutor.ca/tile/servlet/advsearch). Поширюється продукт за ліцензією GPL, домашня сторінка проекту: www.atutor.ca/atutor.

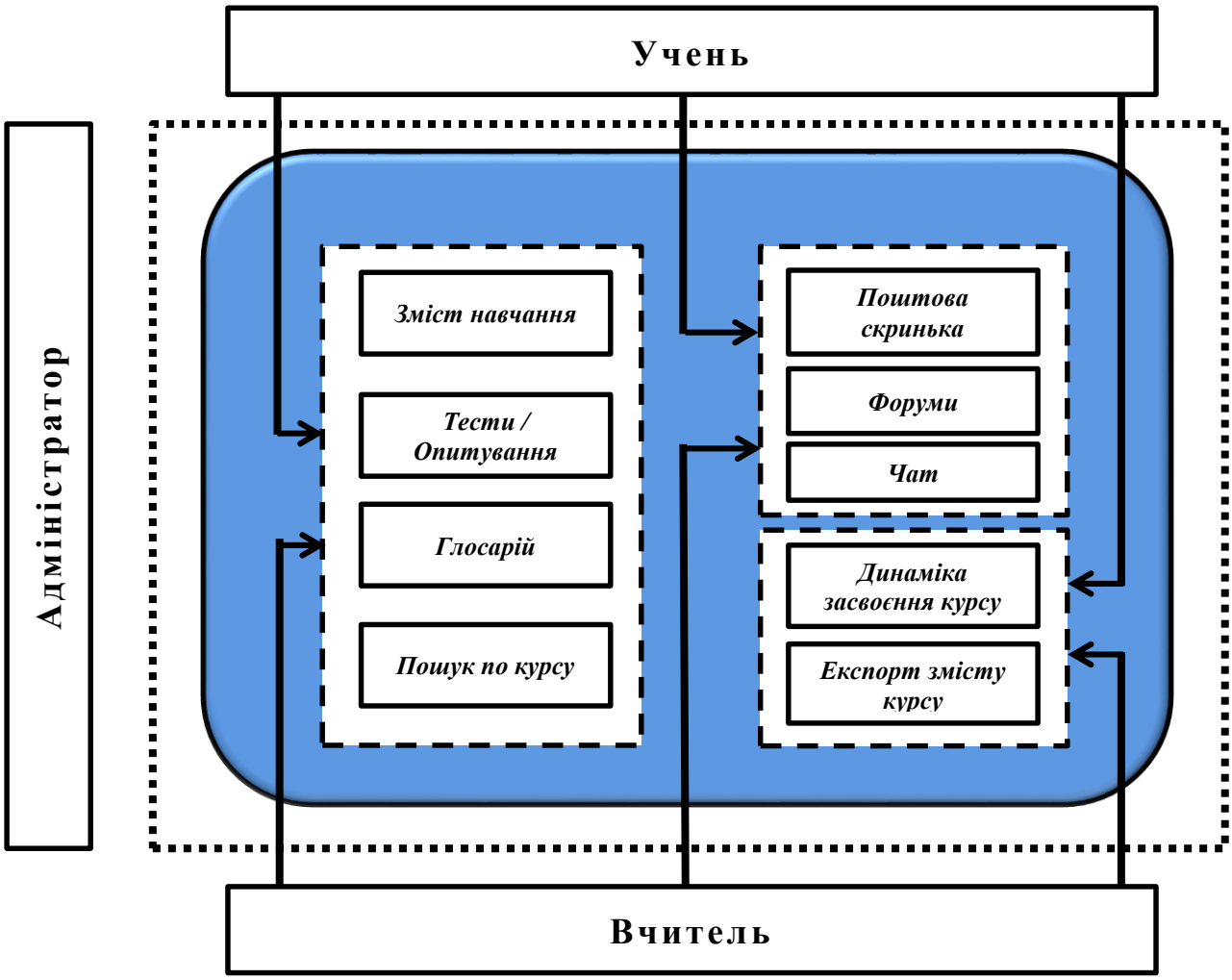


Рис. 2.13. Модель системи управління навчальним процесом ATutor

Встановлення, як і процес оновлення системи, не є складним. Для можливості подальшої роботи необхідно пройти шість кроків, кожен з яких повинен закінчитися успішно. Спочатку необхідно мати комп'ютер з встановленими Apache 1.3/2.x, MySQL 4.0.2+ і PHP 4.3.0+ (з підтримкою MySQL і zlib). Перераховані вище компоненти можуть працювати і на різних варіантах Unix систем, і під управлінням MS Windows. Хоча розробники рекомендують Apache, ATutor може нормально працювати з системати на зразок Zeus, lighttpd, Microsoft IIS та ін.

ATutor підтримує три типи користувачів: адміністратор, вчитель і учень. Залежно від того, в якій ролі прописаний зареєструвався, система сама вибере інтерфейс. Адміністратор володіє найбільшими правами в системі. Після налаштування сервера курсів його основне завдання полягає в оновленні,

локалізації, корекції персональних облікових записів, зміни привілеїв доступу, встановленні нових тем оформлення і додаткових модулів, зборі статистики, захисті контенту. Також адміністратор може створювати нові категорії курсів, при цьому за кожною категорією може бути закріплена своя тема. Всі налаштування зрозумілі і проводяться через WEB-інтерфейс, розібратися з ними нескладно. Локалізувати ATutor надзвичайно легко. Можна заздалегідь завантажити потрібний файл (є модулі українською та російською мовами), а можна завантажити його, використовуючи меню імпорту. Єдине, на що треба звертати увагу, – це збіг версій, для яких доступний модуль локалізації. Останні дещо запізнюються за часом, тому, щоб не доповнювати його вручну, краще використовувати ATutor, для якого вже є переклад інтерфейсу.

ATutor включає ряд технологій, щоби вміст був доступний всім користувачам, включаючи тих, хто застосовує для доступу в інтернет повільні канали, які використовують ранні версії WEB-браузери і старі монітори, та людям з обмеженою працездатністю. Для цього використовується приховування зайвих елементів управління, динамічне меню, мітки полів форми і альтернативна навігація, фіксація останньої позиції, набір гарячих клавіш. Все це дає змогу користувачеві відразу переходити до змісту. Також корисним для поганих каналів є можливість відключення зображень з виведенням альтернативного тексту. Користувачам з поганим зором підійдуть теми з великими шрифтами і можливість збільшення зображення в браузерах, які підтримують цю функцію.

Існують й інші розробки, інтеграція яких в ATutor планується в майбутньому. Однією з них є ATalker (www.atutor.ca/atalker) – це text-to-speech сервер, основою якого послужив festival (www.cstr.ed.ac.uk/projects/festival). В ATutor він буде використовуватися для озвучення уроків, що може бути корисним, наприклад, для учнів зі слабким зором.

Таблиця 1

Порівняльна характеристика
систем управління навчальним процесом MOODLE і ATutor

	MOODLE	ATutor
Розроблення та подання навчального матеріалу	− Модуль «Урок» для подання навчального матеріалу. − Модуль «Глосарій» додає коментарі визначенням та автоматично зв'язує слова в лекціях із визначенням глосарію.	− Створення курсів (вказується опис, доступ, дата публікації). − Модуль відновлення курсів. − Редагування вмісту (ключові слова, схожі теми, попередній перегляд і перевірка відтворення в браузерах). − Словник. − Посилання на інші джерела. − Список літератури.
Розроблення тестів	Модуль «Тест» складається з двох частин: тесту і бази запитань. Тест складається з різноманітних запитань, вибраних із бази запитань. База запитань складається із запитань різних типів: з одним варіантом відповіді, багатьма варіантами чи можливістю вписати свій варіант.	Тести і анкети (запитання, встановлення категорії, оцінка і статистика тестів).
Контроль лекційного матеріалу	Журнал реєстрації активності користувачів (учнів) в блоці «Управління». Можливими параметрами фільтрації журналу є день, назва курсу, клас, учасник, виконане завдання.	− Опитування учасників курсу (при цьому оцінки не виставляються). − Статистика.
Самостійна робота	− Модуль «Завдання». Вчитель створює опис завдання, установку на його виконання та вказує місце, куди учень зобов'язаний завантажити результати. Учень може завантажувати результати у вигляді рефератів, відеоматеріалів, презентацій, таблиць тощо. − Модуль «Робочий зошит» відрізняється від модуля «Завдання» тим, що завдання складаються із відповідей у вигляді тексту, які може редагувати учень.	− Завдання (інструктор задає назву, суть і виконавця). − Пошук в інтернеті на початковій сторінці або вкладці меню.
Інтерактивна взаємодія	*Взаємодія «учень – учень»:* − Форум − Чат − Обмін повідомленнями *Взаємодія «викладач – учні»:* − Форум − Чат − Обмін повідомленнями *Взаємодія «учні – викладач»:* − Чат − Обмін повідомленнями	*Взаємодія «учень – учень»:* − Персональні повідомлення − Модуль обміну файлами − Форум *Взаємодія «викладач – учні»:* − Оголошення − FAQ − Чат − Розміщення новин на баннері − Персональні повідомлення − Рядок новин RSS *Взаємодія «учні – викладач»:* − Форум − Чат − Персональні повідомлення − Опитування

ATutor – повноцінний, легко розширюваний, адаптований, вільнодоступний, оскільки побудований на відкритих технологіях, продукт який може з успіхом застосовуватися як в навчальних закладах, так і в різних установах. Більш повні відомості про продукт можна знайти на сайті проекту.

Проаналізувавши основні принципи створення та роботи систем управління навчальним процесом MOODLE і ATutor, можна узагальнити їхні фундаментальні особливості у порівняльній таблиці і з'ясувати, що дуже значних відмінностей та розбіжностей, які б впливали на їх загальну роботу між ними, не виявлено.

Перспективність застосування LMS. З розвитком нових цифрових технологій системи управління навчальними курсами стають дедалі перспективнішим засобом навчання при вивченні багатьох дисциплін в освітньому просторі України. *Online* – форма навчання сприяє масовому поширенню освіти, роблячи навчальні курси доступними для тих категорій слухачів, які раніше не були охоплені традиційною освітою. Разом з тим, доводиться констатувати низьку якість навчання, що закономірно в контексті існуючих пріоритетів: мінімізація витрат, відповідність стандартам і кількості модулів навчальних програм.

Варто враховувати той факт, що вартість розвитку якісних навчальних систем є (і найближчим часом залишиться) високою. При цьому, необхідно сформувати чіткі критерії виміру ефективності *online* курсів – тільки тоді *online* навчання зможе реалізувати весь свій потенціал.

За твердженнями розробників, широкого застосування системи управління навчальними курсами набудуть тільки тоді, коли в Україні з'являться відповідні технічні можливості та хороші телекомунікаційні канали у всіх навчальних закладах.

Таким чином, можна зробити певні висновки.

− Системи з відкритим кодом дають можливість вирішувати ті ж задачі, що й комерційні системи, але при цьому у користувачів є можливість

доопрацювання та адаптації конкретної системи до своїх потреб та поточної освітньої ситуації.

− Більшість систем з відкритим кодом є крос-платформними рішеннями і не прив'язані до конкретних операційних систем і конкретних WEB-браузерів.

− Використання комерційних систем управління навчальними курсами не доступне більшості наших навчальних закладів через їхню високу вартість, необхідність продовження ліцензії на кожен навчальний рік, прив'язку вартості ліцензій та їх продовження до кількості користувачів системи.

− Сучасні тенденції розвитку LMS спрямовані в бік універсалізації та збільшення функціональності систем. За своїми функціями найбільш передові системи не поступаються комерційним аналогам, а деякі навіть перевершують.

− Системи управління навчальними курсами з відкритим вихідним кодом дають змогу реалізувати той набір функціональних можливостей, що й комерційні рішення з істотно меншими економічними витратами.

Аналіз інформаційних ресурсів інтернету і відгуків на форумах з проблем навчання з використанням ІКТ показав, що найбільший інтерес серед OpenSource систем викликають MOODLE та ATutor. Характерна особливість проекту MOODLE полягає в тому, що навколо нього сформувалося найбільш активне міжнародне співтовариство розробників і користувачів, які діляться досвідом роботи на платформі, обговорюють проблеми, що виникли в процесі роботи, обмінюються планами і результатами подальшого розвитку середовища. Використання ATutor дає змогу педагогам легко організовувати різні курси навчання, під час яких учні отримують зручне середовище навчання, вчитель може максимально легко змінювати зовнішній вигляд системи, а доступність вихідного коду і відкриті інструменти, що застосовуються для побудови сервера курсів, дають змогу за необхідності оперативно внести зміни [285; 207; 237; 240].

2.3. Системи комп'ютерної математики як засіб формування компетентностей в галузі алгоритмізації та програмування

Аналіз стану вивчення інформатики в школі дає підстави стверджувати, що практичному значенню такої важливої теми, як алгоритмізація та програмування, в загальному обсязі курсу інформатики ЗОШ приділяється недостатньо уваги. Основний наголос зроблений на вивчення систем програмного забезпечення комп'ютера, зокрема: операційних систем, програм, що входять до складу MS Office, систем мережевих технологій і т. д. Вагоме місце в деяких навчальних програмах посідає тема безпосереднього програмування, яка, начебто апріорі, передбачає і вивчення алгоритмізації: без алгоритму програми не напишеш. Але як будується цей алгоритм, які правила і рекомендації треба враховувати при його розробці, як сформувати в учнів алгоритмічний підхід до розв'язування прикладної задачі – докладні відповіді на ці запитання не так часто зустрічаються в навчальних програмах курсу інформатики. У підсумку, після навчання інформатики в школі, у старшокласників можуть не сформуватися навички системного алгоритмічного мислення, які так їм необхідні в подальшій діяльності поряд зі знанням прийомів роботи з програмним забезпеченням комп'ютера.

Останніми роками в Україні та за її межами інтенсивно ведуться дослідження з питань упровадження ІКТ у процес навчання шкільних дисциплін. Наукові пошуки започаткували М.І. Жалдак [65; 74; 68; 67; 72; 71], Ю.С. Рамський [192; 193; 195; 199; 200], А.П. Єршов [63]. Проблеми використання комп'ютера як засобу навчання у різних предметних галузях розглядають у своїх працях А.М. Гуржій [47; 32], М.І. Жалдак [65; 74; 70; 68], М.С. Львов [142; 233], Ю.С. Рамський [192; 200; 194], С.А. Раков [189], О.В. Співаковський [227; 233] та ін. Широкого використання у навчальному процесі набули розробки Gran, DG, СЛА тощо вітчизняних дослідників. Застосування комп'ютерної прикладної математики розглядається переважно у довідниковій літературі (В.П. Дьяконов, О. Матросов, О. Лобанова,

Д. Поттер, Г. Прохоров), в якій описуються, як правило, інтерфейси систем (обміну повідомленнями між користувачем та комп'ютером), наводяться приклади розв'язування задач для ілюстрації застосування базових інструментальних засобів.

Питання застосування новітніх винаходів інформатичної науки, зокрема різногалузевих комп'ютерних середовищ, у навчальному процесі сучасної школи у науково-методичній літературі висвітлені недостатньо. Можна було сподіватися, що результати стрімкого розвитку ІКТ будуть узгоджені з подібними змінами як у змісті, так і в методах навчання інформатики. Окремі кроки зроблені, але не в тому обсязі, якого вимагає сучасна практика навчання.

Основними завданнями, що ставляться при вивченні інформатики з метою її подальшого застосування в процесі вирішення проблем у різних напрямках діяльності, є:

– алгоритмізація різноманітних прикладних задач;

– програмування розроблених і типових алгоритмів;

– побудова та дослідження інформаційних, зокрема математичних моделей різних об'єктів з використанням прикладних комп'ютерних систем.

Так, одним із ефективних засобів підвищення результативності навчання алгоритмізації та основ програмування є ООП, що виявляється у педагогічно цілеспрямованому використанні освітніх інноваційних комп'ютерних технологій, у нашому випадку – СКМ, за таких умов: комплексного та систематичного їх застосування; добору навчальних задач, розв'язування яких без них є складним; досягнення вищої мотивації навчання, ніж під час традиційного навчання; формування позитивного ставлення до навчання; забезпечення індивідуалізації процесу навчання. Потрібна подальша робота зі створення підручників, у яких розглядався б процес застосування наведених засобів навчання.

У дисертаційному дослідженні пропонується використання ООП у навчанні алгоритмізації та програмування старшокласників з використанням системи комп'ютерної математики. На думку В. П. Дьяконова [59; 61] та

Ю. С. Триуса [245; 244], комп'ютерна математика може бути визначена як сукупність теоретичних, методичних, алгоритмічних, апаратних і програмних засобів, які призначені для ефективного розв'язування з використанням комп'ютерів широкого кола математичних задач з високим ступенем візуалізації всіх етапів обчислень. На думку Ю. С. Рамського, у системах комп'ютерної математики нагромаджено багатовіковий досвід розвитку математики та інформатики. Нині вони стають потужними засобами діяльності як професійних математиків, так і тих, хто використовує математику для побудови й дослідження математичних моделей в різних предметних галузях, зокрема системі освіти [200, с. 12-16].

Таким чином, перед викладачем ЗОШ стоїть завдання пошуку компромісів, що відповідають вищесказаним суперечливим вимогам при побудові курсу інформатики, в межах якого, як правило, вивчаються основи алгоритмізації та програмування. Автором дисертації апробується використання СКМ не тільки як засобу розв'язування прикладних задач, а і як середовища для вивчення алгоритмізації і програмування.

Виконання алгоритмічного аналізу задачі є дуже складним інтелектуальним процесом. Реальні прикладні задачі різних напрямів, як правило, не завжди добре піддаються формалізації та алгоритмізації, тому алгоритм може бути створений в розрахунку на режим обміну повідомленнями з користувачем, який буде керувати ходом обчислювального процесу.

Етап програмування при якісно розробленому алгоритмі виконується автоматичною заміною блоків схеми алгоритму операторами мови програмування, однак вимагає знання методів і правил формування програм. Для непрофесійних користувачів, які вміють структурувати і алгоритмізувати задачу, але не мають навичок складання програм і роботи з системами програмування, цей етап викликає значні труднощі**.**

Процес розв'язування прикладних задач з використанням сучасних комп'ютерних технологій поданий у вигляді схеми (рис. 2.14):

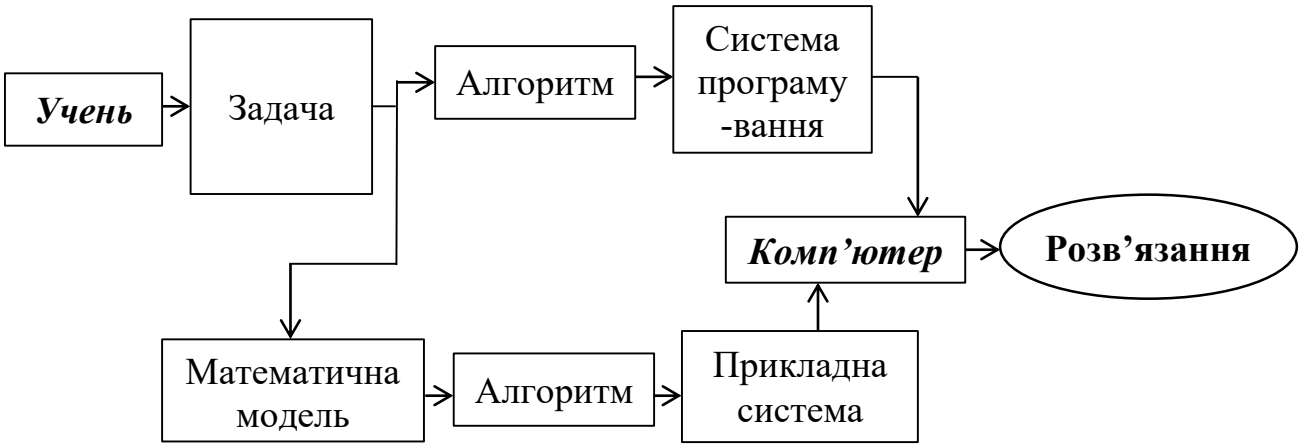


Рис. 2.14. Процес розв'язування прикладної задачі на комп'ютері

З рисунка 2.14 видно, що розв'язування може бути отримане двома шляхами: перший – шляхом створення користувацької програми певною мовою програмування, другий – з використанням прикладної системи без безпосереднього програмування. Стрімкий розвиток прикладних систем із доступним графічним інтерфейсом користувача дає змогу значно збільшити кількість розв'язаних за їх допомогою задач.

Базовими системами для реалізації ООП є, зокрема, системи Matlab, Maple, Mathematica і MathCAD, що включають сотні математичних функцій як для чисельних, так і для аналітичних обчислень, мають доступний інтерфейс, засоби опрацювання текстових і графічних даних, мультимедійні додатки і т. д. СКМ мають розвинену вбудовану мову програмування і можуть бути самодостатньою основою для оволодіння прийомами і принципами програмування і, відповідно, базовим інструментом як для вивчення інформаційних технологій, так і для математично орієнтованих дисциплін.

Сучасна концепція структурного програмування, загальновизнана найбільш універсальною, приділяє найбільшу увагу логіці програми і включає три основні компоненти: низхідне проектування, модульне програмування і структурне кодування.

Наявність перших двох компонентів не викликає сумніву, оскільки вони є очевидним і безумовним атрибутом будь-якої СКМ. Тому стисло зупинимося на третьому компоненті. Структурне кодування передбачає отримання

коректної програми або модуля на основі простих управляючих структур. Як базові вибираються структури слідування, розгалуження, організації циклів і виклику підпрограм. При цьому всі вони допускають тільки один вхід і один вихід. Вказані управляючі структури складають мінімальний базис, що дає змогу створювати коректну програму будь-якої складності з одним входом, одним виходом, без зациклень і недосяжних команд. Структура таких програм чітко простежується від початку до кінця за відсутності передач управління на верхні рівні. Мова СКМ є хорошим прикладом лінгвістичного забезпечення при розробці ефективних навіть складних програм, який поєднує кращі традиції об'єктно-орієнтованої парадигми з орієнтацією на математичні додатки і непрофесійних користувачів.

На рисунку 2.15 продемонстровано критерії оцінювання програмного забезпечення. Таке унаочнення сприяє усвідомленню того, що кожен виконавець може виконати лише команди із власної системи команд, а команди, що не входять до його системи, будуть незрозумілими. Варто зазначити, що під час вивчення алгоритмізації значна увага приділяється добору виконавця поставленої задачі, враховуючи те, що він також є певним об'єктом із притаманними йому властивостями і набором допустимих операцій, які потрібно аналізувати з метою правильного та ефективного їх використання. Отже, вже тут можна провести певні паралелі із використанням ООП в навчальному процесі.

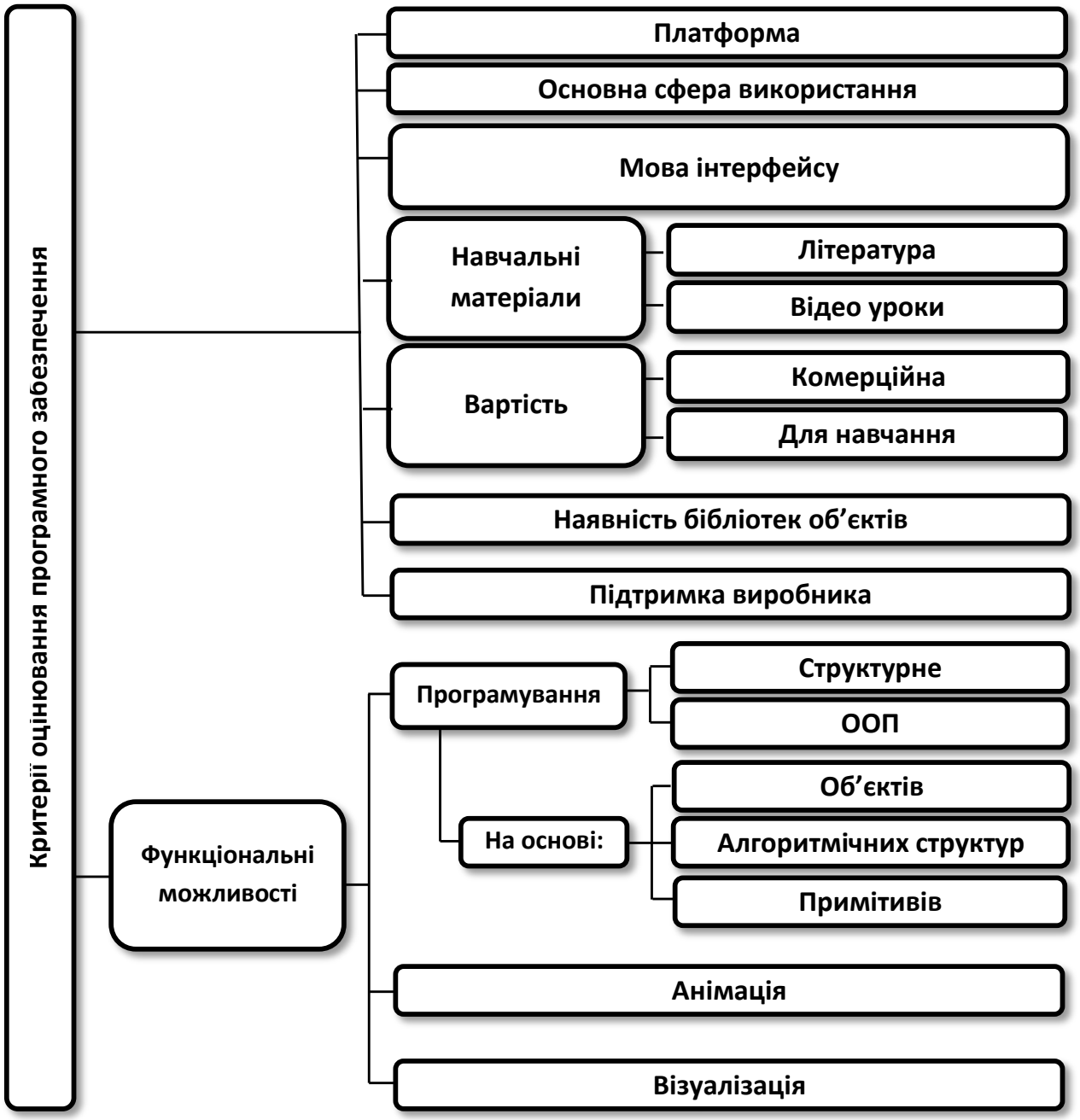


Рис. 2.15. Критерії оцінювання програмного забезпечення

Відзначимо, що використання комп’ютерних математичних пакетів у процесі навчання алгоритмізації та програмування в курсі інформатики сприяє:

– розвитку логічного та алгоритмічного мислення учнів і, як наслідок, підвищенню ефективності навчання інформатики;

– активізації творчої та пізнавальної діяльності учнів, підвищенню їхнього інтересу до навчальної діяльності та зацікавленості в її кінцевому

результаті як в навчанні інформатики, так і в межах міжпредметної інтеграції інформатики та інших дисциплін;

– підвищенню професійної орієнтації учнів в природничо-науковій і технічній діяльності, розвитку практичних вмінь у галузі застосування ІКТ в подальшій професійній роботі.

У ході розробки і виконання програм в СКМ користувач одержує структурований, зручний в користуванні та вивченні документ, який може бути оформлений в будь-якому вигляді: звіт, технічна документація, HTML-документ тощо.

Як коментар до наведених вище тез розглянемо навчальний приклад, що демонструє можливості використання різних СКМ у програмуванні базових алгоритмічних конструкцій. Розв'язування даної задачі супроводжується побудовою базових конструкцій, а саме: слідування, розгалуження, цикл.

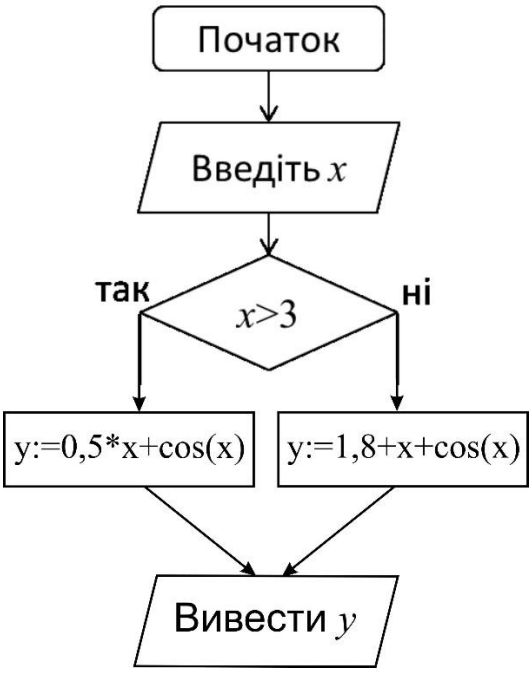
Приклад.

Знайти множину значень функції Y(x) = A(x) + cos(x),,

де A(x) = $\begin{cases} 0{,}5x, \text{якщо } x > 3 \\ 1{,}8 + x, \text{якщо } x \leq 3 \end{cases}$

x змінюється від 0 до 6,28 з кроком 0,1.

Побудуємо алгоритм



Наведемо складену програму на мові Паскаль:

```
Program FUNKCIYA;
Var
        x, y, a: real; {опис змінних}
Begin x: =0;
while x<=6.28 do
        begin
                if x>3 then a:=0.5*x
                else a:=1.8+x;
y=a+cos(x);
writeln (‘При x=‘, x:5:2,
                ‘Значення функції y=‘, y:7:4);
        x=x+0.1;
 end
End.
```

В системі MatLab ця задача розв’язується у вигляді *M*-файла:

```
i=0;
for x=0 : 0.1 : 6.28,
        i=i+1; X(i)=x;
        if x>3, A=0.5*x,
                else A=1.8+x; end;
        Y(i)= A+cos(x);
end;
plot(X,Y).
```

Із розглянутого прикладу бачимо, що набір операторів мовою Паскаль практично не відрізняється від операторів *M*-файла, але сприйняття учнем розв’язуваної задачі значно підвищується завдяки включенню в програму команди побудови графіка функції. Немає необхідності визначати і описувати тип використовуваних даних.

У системі Maple послідовність команд, за допомогою яких розв’язується це завдання, така:

```
> for x from 0 by 0.1 to 6.28 do
> if x>3 then a:=0.5*x
        else a:=1.8+x;
  fi;
> y(x):=a+cos(x);
  end do;
> plot (y(x), x).
```

Результати виводяться автоматично після опрацювання набору команд:

y(0) : =2,8

y(0.1) : =2,895004165

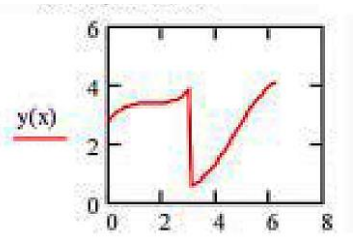
y(0.2) : =2,980066578

y(0.3) : =3,055336489

y(0.4) : =3,121060994

y(0.5) : =3,177582562

Включення функції **plot(y(x), x)** дає змогу отримати графік:



СКМ нині все ширше застосовують в багатьох галузях науки: математиці, фізиці, хімії, інформатиці тощо, стають популярнішими для розв’язування завдань математично орієнтованих дисциплін, в наукових дослідженнях і промисловості. У цьому сенсі СКМ є базою для створення міждисциплінарного зв’язку між математикою і інформатикою, в якій дослідження зосереджуються як на розробці алгоритмів для символьних обчислень (алгебри) та опрацювань на комп’ютерах, так і на створенні мов програмування і програмного середовища для реалізації подібних алгоритмів. Важливо відзначити, що вивчення СКМ дає змогу школяреві (не програмісту)

на достатньо високому рівні оволодіти не тільки навичками програмування, а й сучасними ІКТ (створення WEB-сторінок або призначеного для користувача графічного інтерфейсу, підготовка вихідних мультимедійних документів у вигляді графіків і анімаційних кліпів тощо) і мати в руках потужний інструмент для розв’язання навчальних, а надалі і прикладних інженерних завдань.

Розглянемо роль і місце систем комп’ютерної математики в ієрархії засобів розв’язування математичних задач.

Програмні засоби, орієнтовані на розв’язування математичних задач, умовно можуть бути поділені на п’ять рівнів [4]:

1) вбудовані засоби систем програмування;
2) спеціальні мови програмування;
3) вузькоспеціальні;
4) спеціальні;
5) загальні пакети.

До першого рівня можуть бути віднесені такі системи програмування, як Basic, C, PL/1, Pascal-XSC та ін., до другого – Fortran, ISETL, Prolog та ін. Третій рівень може бути поданий як бібліотеками математичних підпрограм (SSP, NAG та ін.), так і вузькоспеціальними пакетами MacMath, Eureka, VossPlot, Phaser та ін. До четвертого рівня можна віднести такі статистичні пакети, як S-Plus, SAS, XploRe, StatGraf, Systat та ін. Серед пакетів цього рівня в окрему групу можна віднести клас найбільш відомих і, на жаль, дорогих: SAS, Statistica, SPSS. Здійснювати статистичне опрацювання даних, виконувати практично всі необхідні в навчанні і практичній роботі функції, дають змогу такі пакети, як IDAMS і Instat+, але розповсюджуються вони безкоштовно лише для некомерційного використання, а також VisualStat Professional (пакет виготовляється компанією VisualStat Computing (США) і доступний за адресою: www.visualstat.com), ціна якого набагато нижча за більшість комерційних продуктів, при цьому його можна законно використовувати протягом 30 днів.

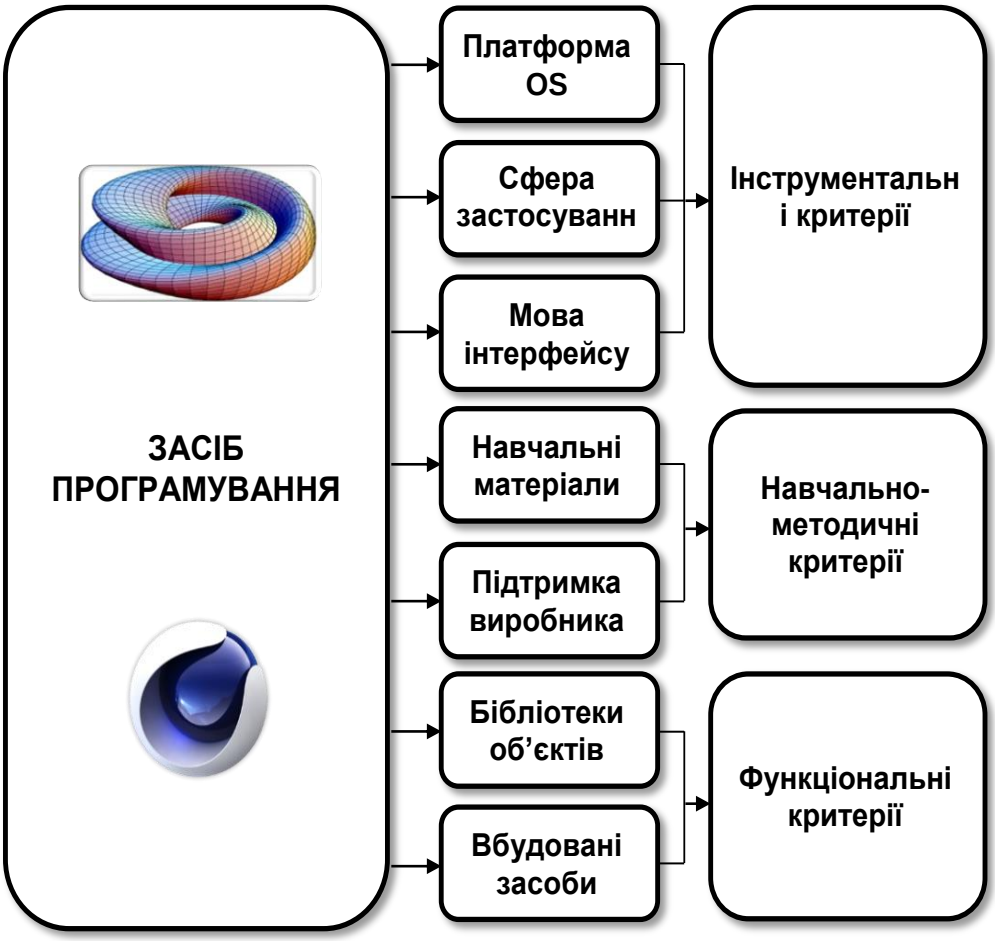


Рис. 2.16. Структурно-функціональна модель засобів програмування

На основі аналізу процесу навчання алгоритмізації та програмування було розроблено структурно-функціональну модель засобів програмування, яка визначає головні критерії для їх порівняння (рис. 2.16).

Серед сучасних пакетів п'ятого рівня, орієнтованих на розв'язування завдань математичного характеру і в числовому, і в символьному видах, можна виокремити групу найбільш потужних пакетів, що користуються значною підтримкою популярних платформ: MathCAD, MatLab, Maple, Mathematica, Reduce, Maxima, MuPAD, SciLab, Derive, Axiom. Перші два пакети, спочатку орієнтовані на чисельні розрахунки, достатньо добре знайомі вітчизняним користувачам, що відображено в літературі [20; 30; 55; 46, 77; 84]. Пакети Mathematica і Maple мають найбільш розвинені засоби аналітичних

(символьних) перетворень, при цьому, володіючи могутніми засобами чисельних обчислень, за останні роки набувають все більшої популярності. В ієрархії СКМ, з погляду оцінки їх обчислювальної потужності, мовних засобів, зручності роботи та інтерпретації результатів, Maple, MatLab, Mathematica і MathCAD – безперечні лідери в своєму класі програмних засобів; при цьому автори розташовують їх саме в наведеній послідовності.

СКМ MatLab (матрична лабораторія) спочатку передбачалася розробником – компанією The Math Works (www.matlab.ru) – як система, орієнтована тільки на чисельні розрахунки, візуалізацію і багаточисельні технічні додатки. Аналітичні обчислення підтримуються в MatLab, але лише за рахунок використання дуже обмеженої частини функцій ядра Maple. Зазначимо при цьому, що аналітичні перетворення – далеко не найсильніша сторона цієї СКМ. Однак в своєму сегменті СКМ MatLab практично не має собі рівних.

MatLab складається з базового додатку: ядра системи, що забезпечує основні функції, і набору додаткових пакетів. Ядро містить велику кількість загальноматематичних і графічних процедур, достатніх для проведення базових розрахунків. Але головна цінність СКМ MatLab, через яку вона завойовує передові позиції серед чисельних систем, – це все-таки додаткові пакети, число яких налічує кілька десятків. Використання цих пакетів дозволяє вирішувати проблеми, мабуть, більшості завдань від моделювання і оптимізації систем управління до створення складних інтерактивних програм візуалізації чисельних експериментів.

У наборі інструментів MatLab центральне місце посідає пакет Simulink, за допомогою якого виконується моделювання і проектування динамічних систем. Simulink – це графічне середовище і налаштовуваний набір бібліотек блоків, що дає змогу моделювати і проектувати технічні системи, тобто в деякому розумінні центральний пакет. Через нього користувач може організувати роботу з іншими спеціалізованими пакетами.

Мова MatLab – високорівнева мова програмування, що ввібрала в себе конструкції Pascal, Basic, Fortran і C, орієнтована на структурний підхід, але підтримуюча й ООП. Компілятор MatLab дає змогу одержувати виконувані модулі, а також об'єктні модулі C++.

На жаль, цей унікальний, але недешевий продукт вимагає чималих обчислювальних потужностей – тільки дискового простору більш 1Гб. Проте наявність у розробника освітніх програм, підтримка фірми-дистриб'ютера (www.mathworks.com) дає змогу планувати його використання в нашій країні.

Якщо MatLab є унікальною системою чисельних обчислень, то Mathematica (розробник – фірма Wolfram Research: www.wolfram.com) – це універсальне високоінтелектуальне середовище, яке орієнтоване на широке коло користувачів різного рівня від – середньої освіти до наукових розробок, але істотно перевершує MatLab щодо символьно-аналітичних обчислень і має багатофункціональну мову програмування, орієнтовану передусім на математичні додатки. Mathematica не має такого багатства інженерних додатків, як MatLab, але пакети розширень системи постійно розвиваються, а її інтерпретована проблемно-орієнтована мова надвисокого рівня за своїм функціональним набором перевершує універсальні мови загального призначення і підтримує практично всі відомі концепції і прийоми програмування. При цьому символьні операції – найбільш сильний аргумент системи. Mathematica задумувалася і створювалася як амбітний науковий проект, тому невипадково її видавничі можливості не мають аналогів серед інших СКМ.

Система MathCAD популярна в науковому середовищі, зокрема в технічній галузі. Характерною особливістю цієї СКМ є використання звичних стандартних математичних позначень, тобто документ на екрані виглядає так само, як і звичайний математичний розрахунок. Система орієнтована, насамперед, на проведення чисельних розрахунків, але має вбудований символьний процесор Maple, що дає змогу виконувати аналітичні перетворення. Mathcad є середовищем візуального програмування, тобто не

вимагає знання специфічного набору команд, має надзвичайно зручний математико-орієнтований інтерфейс і прекрасні засоби наукової графіки. До складу системи входять зручний текстовий редактор, що дає змогу вводити, редагувати і форматувати як текст, так і математичні вирази, обчислювальний процесор, що виконує розрахунки за заданими формулами з використанням вбудованих функцій, які реалізовують чисельні методи, символьний процесор і чимала довідкова система, подана електронним підручником. СКМ Mathcad успішно застосовується в навчальному процесі для реалізації математичних моделей технічних пристроїв і систем в різному проектуванні, в практичних курсах. Існує безліч цікавих розробок, виконаних в цій СКМ. Докладні відомості про нові версії системи можна знайти на сайті виробника MathCAD http://www.mathcad.com.

Дещо менш відомими є решта математичних пакетів даного рівня. Вони також заслуговують певної уваги і не в останню чергу тому, що їх легальні версії можуть бути безкоштовно отриманими з Web-сайтів фірм-виробників. Опис основних можливостей використання ряду СКМ приведений на сайті http://itc.ua/software.

Пакет Scilab вільно розповсюджується центром Scilab Consortium з Web-вузла www.scilab.org, з якого можна завантажити останню версію програми і комплект документації. Scilab не вимагає великих системних ресурсів: інсталяційний модуль має розмір майже 20 MB, дискового простору потрібно 41 MB. Пакет має багато спільного з СКМ Matlab, його можна розглядати як полегшений варіант свого могутнішого «побратима», що деякою мірою ідеально підходить для вивчення в середній школі.

Scilab є командним інтерпретатором і складається з інтерпретуючої системи, яка приймає команди користувача і видає результати, і двох бібліотек: власних функцій і додаткових – мовами C і Fortran, має більше тисячі функцій. Як і MatLab, він призначений майже виключно для реалізації чисельних методів, а підтримувані символьні операції мають радше демонстраційне навантаження. Scilab дає змогу також обмінюватися даними з

іншими додатками: підтримуються формати документів Matlab і Maple, структурований текст і TeX. Аналогія пакету з MatLab ще й в тому, що функції системи, які відносяться до деяких прикладних сфер математики і техніки, зібрані в додаткові пакети розширень (так звані toolboxes). Для досягнення цілей навчання дуже важливо, що з метою створення призначених користувачеві алгоритмів і програм Scilab має в своєму розпорядженні вбудовану мову, яка володіє широким набором конструкцій для організацій циклів, умовних переходів, операцій введення/виведення, за допомогою якої можна дістати доступ до всіх внутрішніх команд додатку. У Scilab реалізовані концепція процедурного програмування і використання кодів мовами C і Fortran. Дуже важливою є функція maple2scilab, що дає змогу перетворити багато об'єктів Maple в коди мовою Scilab. Той факт, що система SciLab дуже близька до системи MatLab, дає підстави розглядати тандем програм Maple і SciLab як вдалий для вивчення в ЗОШ. У старших класах цей зв'язок може бути трансформований в навчання на базі останніх версій Maple і MatLab. Такий підхід до вибору програмних засобів дає змогу багато в чому зберегти наслідковий характер їх вивчення, дозволяючи при цьому школярам використовувати новітні і досконаліші розробки програмного забезпечення.

СКМ Maxima (перша назва – Macsyma) була створена в кінці 1960-х років в Массачусетському технологічному інституті (США). Інсталяційний модуль системи займає 10 МВ. Його разом з документацією і підручником по системі можна одержати на Web-вузлі системи www.maxima.sourceforge.net. Якщо Scilab – система чисельної математики, то, використовуючи Maxima, можна вирішувати складні аналітичні задачі.

«Уміння» виконувати складні аналітичні операції і перетворення є головною ознакою Maxima. Серед її основних операцій – операції аналізу (диференціювання, інтегрування, обчислення границь), досконалий механізм векторно-матричних операцій, подання виразів у розгорнутій формі, розкладання функцій рядами, спрощення, перетворення, підстановки і т. п. За допомогою Maxima можна вирішувати рівняння і системи різних типів.

Графічні функції Maxima, мабуть, скромніші, ніж у інших СКМ, але загалом дозволяють одержати якісні графіки для практичних додатків.

Мова програмування підтримує складні конструкції, допускаючі оператори розгалуження, циклів, введення/виведення і т.д., має гнучкі засоби для роботи з масивами. Maxima написана мовою Lisp і безпосередньо підтримує багато її команд. Практично Lisp є ядром системи і до нього допускається звертатися при «низькорівневому» програмуванні.

MuPAD – сумісний проект фірми SciFace Software і дослідницької групи Університету м. Падерборн (Німеччина). Вартість MuPAD дає змогу розглядати можливість його легального використання СКМ з навчальною метою. При цьому для більшості вирішуваних задач MuPAD практично не поступається найбільш відомим розробкам даного класу. Ця СКМ має сильне аналітичне ядро, яке підтримує всі стандартні операції аналізу і видає результат в загальноприйнятій математичній нотації, а для наближеного розв'язання складних завдань в системі є спеціальний пакет Numeric, що містить чисельні алгоритми для диференціальних рівнянь, лінійної алгебри, знаходження коренів функціональних рівнянь і систем, а також чисельного інтегрування. У MuPAD є пакет Linalg, що виконує лінійні операції алгебри, і пакети, що підтримують функції комбінаторики, лінійного програмування, інтегральних перетворень, статистики, теорії чисел, теорії графів тощо.

Остання версія MuPAD дає змогу будувати практично всі основні види дво-, тривимірних і спеціальних графіків з детальними налаштуваннями їх властивостей, створювати анімаційні ефекти, включати в документи зовнішні графічні файли. Побудовані графіки зберігаються в поширених растрових і векторних форматах, а також у форматі цифрового відео AVI. Система має розвинену мову для написання програм, що ввібрала в себе найбільш загальні конструкції C/C++, Pascal і Fortran. З пакетом можна ознайомитися на Web-вузлі www.mupad.de, з нього ж можна завантажити і використати протягом 30 днів останню його версію.

З усіх згаданих СКМ, ми обираємо Maple через цілий ряд переваг, серед яких варто особливо виокремити такі, як розвинені графічні засоби, достатньо ефективні засоби розв'язування різних класів задач, засоби створення графічних інтерфейсів користувача, могутня бібліотека математичних функцій, великий набір супутніх пакетів для різних додатків, сучасна вбудована мова програмування інтерпретуючого типу, доступність ряду інших Windows-додатків, перспективна концептуальна підтримка тощо [291].

На нашу думку, ця система є найкращою для формулювання, розв'язування і дослідження різних математичних моделей. Її засоби алгебри істотно розширюють діапазон проблем, які можуть бути вирішені на якісному рівні. Важливим аргументом на користь Maple є можливість його інтеграції з математичним довідником (Standard Math Interactive) CRC Standard Mathematical Tables and Formulae видавництва CRC Press Inc. і математичним словником (Interactive Math Dictionary) канадської фірми MathResources Inc.

Із досвіду використання ряду СКМ відзначимо, що Maple, мабуть, найбільш зручна і відкрита система, що використовує розвинену вбудовану мову інтерпретуючого типу. Основний конкурент – Mathematica – має не такий зручний синтаксис мови. Нарешті, Maple має розвиненіші інструментальні засоби (надання використовуваного алгоритму розв'язування задачі, налаштування графічної оболонки користувачем на конкретні додатки та ін.), а також широкий спектр безкоштовних додатків для використання в різних галузях. Дуже істотним чинником є те, що символьні додатки СКМ MathCAD і MatLab використовують ядро саме Maple.

Використовуючи пакет Maple, можна реалізувати новітню технологію символьних обчислень і чисельних обчислень з будь-якою точністю, адже він містить інноваційні Web-компоненти і розвинені математичні алгоритми для розв'язання складних математичних задач. Останніми роками розробники приділяють багато уваги розширюваній технології призначеного для користувача інтерфейсу (Maplets), яка повинна дати змогу усунути істотний недолік (втім, властивий всім СКМ): складність організації розвиненого

призначеного для користувача інтерфейсу. Наразі пакет використовують безліч вчителів, студентів, вчених, дослідників і фахівців з різних галузей. Практично кожен провідний університет і науково-дослідний інститут в світі, включаючи Cambridge, Stanford, Oxford, Waterloo тощо, використовують пакет для навчальних і дослідницьких цілей. Із промисловою метою пакет використовується такими провідними корпораціями як Boeing, Bosch, Canon, NASA, Toyota, Sun Microsystems, Hewlett Packard, Motorola, General Electric, Daimler-Chrysler, Ford і ін.

Детально проаналізувавши кожну із запропонованих СКМ, проведемо їх порівняльний аналіз та узагальнимо всю картину у таб. 2. Для оцінювання вибрана 10-бальна шкала. Бали відповідають лише порівняльним оцінкам і допомагають раціонально вибрати СКМ за переліком критеріїв.

Таблиця 2

Порівняльна характеристика СКМ

СКМ **Критерії оцінювання системи**	**MathCAD**	**MatLab**	**Mathematica**	**Maple**	**Maxima**	**MuPAD**	**SciLab**
1	*2*	*3*	*4*	*5*	*6*	*7*	*8*
Мінімальні вимоги до апаратних ресурсів	процесор Pentium II; 128 Мбайт оперативної пам'яті; 200-400 Мбайт дискового простору	процесор Pentium III; 256 Мбайт оперативної пам'яті; 400 Мбайт дискового простору	процесор Pentium II; 128 Мбайт оперативної пам'яті; 400-550 Мбайт дискового простору	процесор Pentium III; 128 Мбайт оперативної пам'яті; 400 Мбайт дискового простору	процесор Pentium III; 256 Мбайт оперативної пам'яті; 80 Мбайт дискового простору	процесор Pentium II; 256 Мбайт оперативної пам'яті; 400-500 Мбайт дискового простору	процесор Pentium III; 128 Мбайт оперативної пам'яті; 20 Мбайт дискового простору
Оцінка	*(10)*	*(7)*	*(7)*	*(9)*	*(7)*	*(6)*	*(8)*
Повнота ядра символьних операцій	близько 300 можливих функцій	близько 2000 можливих функцій	близько 800 можливих функцій	близько 2500 можливих функцій	близько 2000 можливих функцій	близько 2000 можливих функцій	близько 1500 можливих функцій
Оцінка	***(4)***	***(9)***	***(6)***	***(10)***	***(9)***	***(9)***	***(8)***
Повнота системи (можливості не тільки ядра, але й усіх	незначна кількість пакетів та 3 бібліотеки	пакети розширень додаються окремо	велика кількість бібліотек	велика кількість бібліотек	пакети розширення x і wx Maxima	великий набір математичних об'єктів і алгоритмів	звичайні та Xcos модулі

зовнішніх бібліотек і пакетів)							
Оцінка	***(4)***	***(0)***	***(10)***	***(10)***	***(7)***	***(8)***	***(9)***
Вартість системи	≈ 495$	≈ 1055$	≈ 995$	≈ 695$	без-коштовна	≈ 700$	без-коштовна
Оцінка	***(9)***	***(2)***	***(4)***	***(8)***	***(10)***	***(7)***	***(10)***
Продуктивність системи для простих операцій	швидкий, майже миттєвий, результат	швидкий результат	дуже повільний результат	швидкий результат	достатньо швидкий результат	швидкий результат	достатньо швидкий результат
Оцінка	***(10)***	***(9)***	***(4)***	***(9)***	***(5)***	***(9)***	***(5)***
Правильність результату	правильний результат у 70%	правильний результат у 70%	правильний результат у 50%	правильний результат у 70%	правильний результат у 60%	правильний результат у 65%	правильний результат у 70%
Оцінка	***(8)***	***(8)***	***(4)***	***(8)***	***(5)***	***(7)***	***(8)***
Графічні можливості	стандартна математична графіка	багатий вибір двох- і тривимірної графіки	багатий вибір двох- і тривимірної графіки, діаграм і різних фігур, звуку	дещо менший вибір графіки, порівняно з Mathematica	якісна побудова графіків	двох- і тривимірна графіка	наявність графічних вікон
Оцінка	***(8)***	***(9)***	***(10)***	***(9)***	***(8)***	***(8)***	***(7)***
Інтерфейс користувача	зручне меню, багато-віконний режим, піктограми тощо	зручний інтерфейс користувача	багатий інтерфейс користувача	достатній інтерфейс користувача	кілька графічних інтерфейсів користувача	робота відбувається у вікні блокноту	достатньо комфортний інтерфейс користувача
Оцінка	***(10)***	***(8)***	***(10)***	***(8)***	***(7)***	***(5)***	***(6)***
1	*2*	*3*	*4*	*5*	*6*	*7*	*8*
Можливості редагування документів	високі можливості редагування	високі можливості редагування	достатні можливості редагування	вбудований текстовий редактор	середні можливості редагування	достатні можливості редагування	малі можливості редагування
Оцінка	***(10)***	***(10)***	***(8)***	***(9)***	***(6)***	***(8)***	***(5)***
Система довідки	детальне меню, система пошуку, поширена інформація	достатньо поширена	відсутня база даних ядра програми, кількість прикладів застосування функцій мала	поширена система довідки, наявна підказка для редагування	наявна	наявна	наявна, не поширена
Оцінка	***(9)***	***(8)***	***(7)***	***(9)***	***(5)***	***(5)***	***(5)***
Мова програмування	слабо розвинена мова	розвинена мова	високо розвинена структурна мова	достатньо розвинена структурна мова	структурна мова	паскале-подібна мова	структурна мова з підтримкою об’єктів
Оцінка	***(4)***	***(10)***	***(10)***	***(9)***	***(8)***	***(8)***	***(8)***
Безаварійність роботи	достатньо надійна	надійна	часті помилки системи	достатньо надійна	достатньо надійна, нечасті помилки	надійна	з перемінним успіхом
Оцінка	***(8)***	***(10)***	***(4)***	***(8)***	***(7)***	***(9)***	***(7)***
Загальна оцінка	**94**	**90**	**84**	**106**	**93**	**89**	**86**

Резюмуючи вищесказане, вважаємо, що в навчальному процесі при вивченні алгоритмізації та основ програмування доцільно використовувати пакет Maple. І це повністю виправдано, адже Maple – детально апробована розвинена потужна СКМ. Безперечно, ця система повинна бути застосована в кожному навчальному закладі з поглибленим вивченням інформатики. Цьому істотно сприяє і творчий альянс Waterloo Maple зі всесвітньо відомим розробником математичного програмного забезпечення NAG Ltd. І це при тому, що останній має свою достатньо пристойну СКМ – AXIOM. Разом з цим реалізовується альянс розробників Maple і MatLab, до теперішнього часу існує інтерфейс між цими СКМ (втім, поки дуже громіздкий і складний у використанні). Якщо в майбутньому ці два пакети будуть інтегровані, то можна буде говорити про вищий рівень розвитку СКМ, на якому перебуває одна система. Крім того, пакет Maple постійно відвойовує позиції у Mathematica і починає домінувати в освіті, що є істотним з орієнтацією на перспективу; використовувана Maple ідеологія посідає все більше місце при створенні електронних матеріалів математичного характеру. У зв'язку з цим Maple вважається найбільш перспективною і цікавою СКМ при використанні в освітніх і дослідницьких програмах. Останнім доказом на користь Maple є те, що версія Maple V R4 вільно розповсюджується у вигляді демо-версії. Термін «демо-версія» не зовсім точно відображає її функції. У цьому випадку – це практично повноцінна рання версія Maple, що не вимагає для установки паролів, реєстрації і т. п. Отримати її можна з освітнього сайту http://www.exponenta.ru/educat/free/free.asp#maple1.

Як зазначає Ю. І. Сінько, сьогодні особливої уваги заслуговують програмні продукти, що створені українськими розробниками [220]. Ці програми розробляються конкретно під українські методичні системи навчання інформатики та математики. В Україні створено кілька СКМ, рівень розробки яких відповідає світовим і які рекомендовані Міністерством освіти і науки України для використання у навчальному процесі ЗОШ.

Це, зокрема [182]:

− Gran1 (автори М.І. Жалдак, Ю.В. Горошко; Національний педагогічний університет імені М.П. Драгоманова), створена для операційних системи MS-DOS, набула широкої популярності і використовується в багатьох ЗОШ і ВНЗ України для супроводу вивчення математики; містить режим динамічних параметрів);

− Gran-2D (автори М.І. Жалдак, О. Вітюк; Національний педагогічний університет імені М.П. Драгоманова) і DG (автори С.А. Раков, К. Осенков; Харківський національний педагогічний університет ім. Г.С. Сковороди), призначені для графічного аналізу систем геометричних об’єктів на площині;

− Gran-3D (автори М.І. Жалдак, О. Вітюк; Національний педагогічний університет імені М.П. Драгоманова) призначені для графічного аналізу просторових (тривимірних) об’єктів;

− ТерМ (автор М. Львов; Херсонський державний університет), призначена для підтримки навчання алгебри в ЗОШ.

На базі цих програмних засобів створено програмно-методичні комплекси (ПМК) Gran, DG, ТерМ, що успішно використовуються у вітчизняних ЗОШ і ВНЗ. Відомі вони і за межами України. Як бачимо, в нинішній ситуації розвитку і динаміки проникнення ІКТ в науку й освіту важливо не відстати від передових світових технологій. Враховуючи наші реалії, навіть не так важливо правильно вибрати СКМ, скільки використовувати в навчальному процесі хоча б одну з них, підвищуючи ерудицію та освітній рівень учнів.

Отже, мовне середовище СКМ, маючи всі функції універсальної мови програмування для вивчення алгоритмізації, після закінчення навчання одночасно з навичками програмування дає учневі потужний інструмент для виконання навчальних, а згодом і наукових розрахунків. Можливість паралельно і в межах вивчення алгоритмізації вирішувати завдання інших дисциплін неодмінно підвищує мотивацію до вивчення курсу інформатики. Вивчення алгоритмізації в СКМ як потужному середовищі математичного

моделювання дає змогу інтегрувати різні дисципліни і будувати єдину взаємопов’язану несуперечливу концепцію навчання, нарощуючи рівень і складність вирішуваних завдань з кожним роком. Важливою є та обставина, що вивчення алгоритмізації з використанням СКМ дає змогу одночасно знизити проблему «страху» перед математикою старшокласників, позбутися від рутини в обчисленнях при виконанні розрахункових робіт, уникнути дублювання завдань і знань у різних навчальних дисциплінах.

Нарешті, варто відзначити і наступний чинник. Триває дискусія між прихильниками класичного вивчення інформатики як дисципліни програмування і тими, хто вважає, що нині учнів треба вчити тільки роботі з прикладними пакетами. Використання середовища СКМ, дає змогу знайти компроміс між прихильниками цих різних підходів у вивченні інформатики. Суттєвою є проблема вартості використання легальних систем програмування, яка вирішується тим, що розробники системи Maple надають безкоштовні версії ранніх (але цілком достатніх для розв’язання шкільних завдань) версій пакета.

Використання вчителем на уроці будь-якої СКМ залишає йому час на виконання кожним учнем більшої кількості індивідуальних завдань з теми, а також на перевірку результатів [170]. Разом з тим опанування однією СКМ, зазвичай, полегшує процес використання СКМ інших типів. Важливо лише грамотно застосовувати могутній арсенал засобів, які пропонують розробники програмного забезпечення. Також з’являється можливість включити в роботу всіх школярів, навіть з дуже невисоким рівнем знань, що важко дається при традиційному навчанні. Результати впровадження комп’ютерної техніки в навчальний процес засвідчують, що учні краще засвоюють матеріал і усвідомлюють взаємозв’язок між поняттями, які вивчаються, вони набувають досвіду використання сучасних ІКТ у майбутній професійній діяльності.

Все вищесказане дає змогу зробити такі висновки:

 СКМ мають розвинений інструментарій програмування, що містить повний набір базових операторів, повністю відповідний набору операторів мови Паскаль;

 СКМ наочно відображають вміст простих змінних, структурованих даних, що підвищує розуміння процесу їх опрацювання;

 СКМ можуть бути повноцінним інструментарієм для реалізації базових і типових алгоритмів з використанням ООП, при вивченні алгоритмізації та програмування в курсі інформатики для старшокласників.

Важливо відзначити, що вивчення СКМ дає змогу учневі сформувати алгоритмічний стиль мислення, наочно демонструючи йому формальний, алгоритмічний характер процесу розв’язування задачі, освоїти на досить високому рівні основи програмування, опанувати сучасні ІКТ (створення веб-сторінок, користувацького графічного інтерфейсу, підготовка вихідних мультимедійних документів у вигляді графіків і анімаційних кліпів тощо) і отримати розвинений інструмент для розв’язування прикладних завдань, використовуючи ООП. Включення систем комп’ютерної математики в навчальний процес сприяє не тільки підвищенню загального рівня знань у галузі інформатики, а й активізації інтересу у старшокласників до застосування ІКТ при розв’язуванні тематичних завдань з інших навчальних дисциплін.

2.4. Методика навчання алгоритмізації та основ програмування із використанням СКМ Maple

Розглянемо методичну систему навчання алгоритмізації та основ програмування у старшій школі із поглибленим вивченням інформатики та обґрунтуємо основні її аспекти. Загалом необхідно конкретизувати мету навчання і розробити зміст цього розділу курсу, що ґрунтуються на використанні СКМ як засобів навчання в класах з поглибленим вивченням інформатики, а також вибрати методи та засоби навчання і адаптувати їх до використання комп'ютерного математичного пакета, розробити необхідне демонстраційне та методичне забезпечення.

Мета навчання є компонентом, який визначає зміст інших компонентів системи, а загалом усі вони взаємозв'язані. Головна мета навчального процесу – максимальне підвищення активізації пізнавальної діяльності учнів, формування в них інформатичних компетентностей, що призведе до розвитку системно-логічного мислення. У процесі навчання передбачується розв'язування різних класів завдань на основі здобутих знань. При цьому важлива роль належить формуванню навичок щодо отримання, впорядкування, аналізу, опрацювання, збереження та передавання даних. Фактично основою для цього є моделювання творчого процесу за рахунок створення проблемної ситуації та управління пошуком розв'язку проблеми. При цьому осмислення, прийняття та вирішення цих проблемних ситуацій відбувається при достатній самостійності учнів, але під керівництвом педагога в процесі їх взаємодії. Цей аспект надзвичайно важливий, оскільки в ньому і полягає основна відмінність проблемного навчання від евристичного, яке передбачає, що відбувається навчання не тільки учня, а й учителя. Крім підвищення мотивації, одним з ефектів використання проблемних методів навчання є розвиток уваги, волі, підвищення самооцінки школярів. Це все, відповідно, вдало відображається на опануванні знань, умінь та навичок, а також на підвищенні творчого потенціалу учнів.

Навчання школярів розділу «Алгоритмізація та програмування» зводиться до формування та розвитку алгоритмічного, логічного і математичного мислення. Використання СКМ Maple під час поглибленого вивчення алгоритмізації та програмування в ЗОШ дозволяє реалізувати такі завдання:

1) зробити процес навчання інформатики більш наочним, динамічним та цікавим, а, отже, значно ефективнішим;

2) досягти тіснішої відповідності принципів структурного програмування логіці системного мислення для розвитку у старшокласників алгоритмічного мислення та якісного засвоєння ними основ програмування;

3) покращити підготовку учнів у галузі алгоритмізації та програмування;

4) розвинути системне мислення учнів їх творчі та дослідницькі здібності, використовуючи функції СКМ Maple і графічну інтерпретацію результатів програмування;

5) навчити школярів створювати авторські програмні продукти на основі комп'ютерного математичного пакета Maple, активізуючи творчу і пізнавальну діяльність;

6) підвищити професійну зорієнтованість учнів у природничому та технічному напрямках, розвинути професійні компетентності та практичні вміння застосування ІКТ в урочній та позаурочній діяльності;

7) здійснити інтеграцію освітніх предметів (математики, інформатики, фізики) через виконання проектних робіт.

Для досягнення цієї мети необхідно визначити зміст навчання, враховуючи використання СКМ Maple як одного із засобів організації навчального процесу (рис. 2.17).

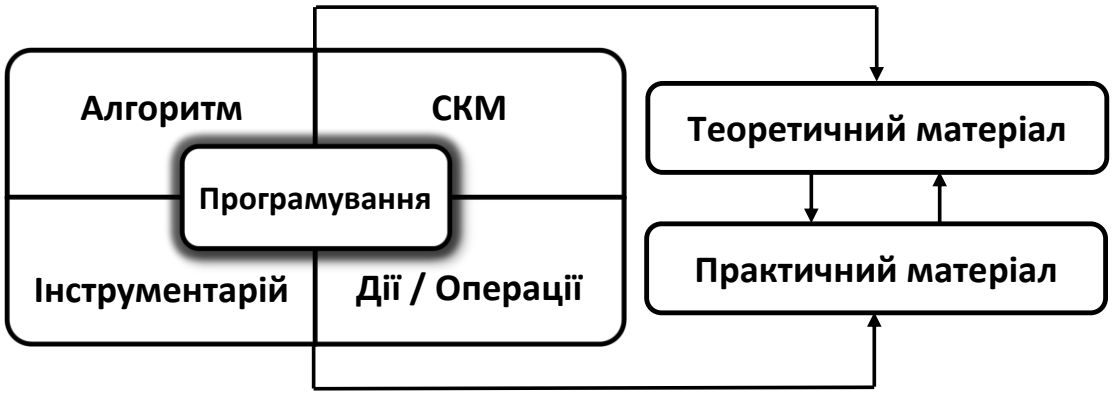


Рис. 2.17. Структура змісту навчання алгоритмізації та програмування у старших класах

У результаті вивчення алгоритмізації та основ програмування з використанням СКМ Maple учні повинні **знати:**

− основні поняття: алгоритм, слідування, розгалуження, оператори присвоєння, цикли, масиви, процедури, функції, ряди;

− основні алгоритмічні структури;

− можливості використання СКМ;

− основні поняття мови програмування Maple;

− межі застосування та властивості вбудованих процедур і функцій.

Учні повинні **вміти:**

− побудувати алгоритм розв’язування задачі;

− реалізовувати основні алгоритмічні структури за допомогою операторів Maple;

− розв’язувати задачі, використовуючи основні типи алгоритмічних структур: слідування, цикл, розгалуження;

− розв’язувати задачі з масивами та рядками;

− описувати і використовувати при розв’язуванні задач процедури та функції;

− створювати графічні об’єкти та їх анімацію;

− розв’язувати комбіновані задачі.

На рис. 2.18 зображено модель цілей методики навчання учнів старших класів ЗОШ з поглибленим вивченням алгоритмізації та програмування, яка відображає необхідність опанування відповідними знаннями, уміннями та інформатичними компетентностями.

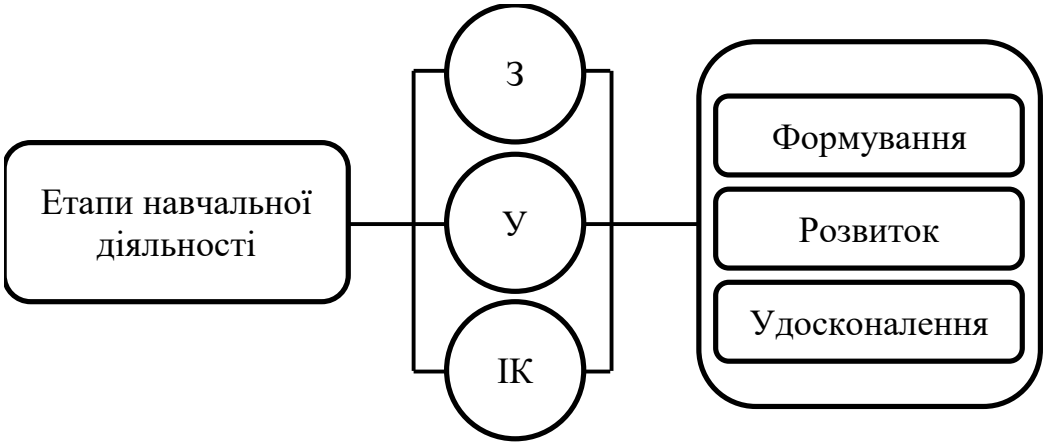


Рис. 2.18. Модель цілей методики навчання алгоритмізації та програмування старшокласників

Для досягнення поставлених завдань пропонуємо два напрямки: впровадження допоміжного курсу «Вивчення системи комп'ютерної математики Maple» та розробка програми для вивчення розділу курсу інформатики «Програмування в середовищі Maple».

Допоміжний курс може проводитися 1 раз на тиждень у 10 або 11 класі залежно від навчального плану і календарного планування курсу інформатики в школі. Для його методичного забезпечення розроблене тематичне планування [див. Додаток Б]. Курс спрямований на вивчення можливостей використання комп'ютерного математичного пакета Maple і є базою для вивчення основ програмування в школі при використанні Maple як системи програмування.

Наступним компонентом впровадження СКМ Maple в поглиблений курс інформатики є навчання програмування із заміною середовища програмування Basic чи Pascal на цей пакет. Розглянемо тематичне планування за

розробленою програмою для вивчення розділу курсу інформатики «Програмування в середовищі Maple» (20 год.).

Після закінчення вивчення пропонованого розділу курсу інформатики учнями виконуються проектні роботи з використанням СКМ Maple.

Методи, форми та засоби навчання, необхідні для завершення створення структури методичної системи навчання алгоритмізації та програмування на поглибленому рівні із застосуванням СКМ Maple, побудовані на існуючих методах і формах, скорегованих із врахуванням особливостей нового засобу навчання.

Розглянемо ***методи*** навчання алгоритмізації та основ програмування із використанням Maple. За характером пізнавальної діяльності учнів є чотири основні методи навчання: пояснююче-ілюстративний, репродуктивний, проблемний, дослідницький. Спроектуємо їх на зміст навчання, враховуючи використання СКМ Maple:

− **пояснювально-ілюстративний метод** навчання реалізує використання комп'ютерного пакета Maple як засобу для створення демонстраційних матеріалів до уроків та посібників інтерактивного навчання;

− **репродуктивний метод** навчання із застосуванням комп'ютерних засобів передбачає засвоєння знань, які повідомляє учневі вчитель за допомогою комп'ютера, та організацію діяльності з відтворення вивченого матеріалу і його використання в аналогічних ситуаціях. Знання та вміння програмування відпрацьовуються школярами з СКМ Maple;

− **проблемний метод** навчання використовує функціональність комп'ютерного пакета Maple для організації навчального процесу як постановки та пошуків способів розв'язування деякої проблеми. Як відомо, проблемне навчання є типом розвиваючого навчання, в якому поєднуються систематична, самостійна та пошукова діяльність учнів із опануванням ними готових висновків науки, а система методів побудована з урахуванням принципу проблемності; процес навчання орієнтований на формування пізнавальної самостійності учнів, стійкості мотивів учіння і мислительних (в

тому числі, творчих) здібностей в процесі засвоєння ними наукових понять та способів діяльності. Саме паралельне навчання старшокласників основ роботи в Maple і можливостей використання програмування в цьому комп'ютерному математичному середовищі дає можливість для ефективного застосування проблемного методу навчання алгоритмізації та програмування;

− **дослідницький метод** навчання із застосуванням СКМ Maple забезпечує самостійну творчу діяльність учнів в процесі проведення науково-технічних досліджень у межах визначеної тематики із розробкою програмних продуктів. При використанні цього методу знання та уміння є результатом активного дослідження, відкриття; внаслідок цього навчання, як правило, є набагато приємнішим та успішнішим, ніж при використанні інших вищезазначених методів. Успішну реалізацію дослідницького методу при навчанні програмування дає використання функцій СКМ Maple.

− **метод проектів** є невід'ємною складовою дослідницького методу навчання. У педагогіці він розглядається як сукупність прийомів операцій, використання яких допомагає опануванню теоретичних знань і практичних умінь в тій чи іншій діяльності. З методичного погляду, проектна діяльність школярів є теоретичним або практичним проблемно-орієнтованим дослідженням, яке учні проводять з навчальною метою під керівництвом одного або кількох учителів. Головний основоположний принцип методу проектів –передусім враховувати інтереси дитини.

Робота за методом проектів – це відносно високий рівень складності педагогічної діяльності, який передбачає значну кваліфікацію вчителя. Якщо більшість загальновідомих методів навчання потребують наявності лише традиційних компонентів навчального процесу – вчителя, учня (класу) і навчального матеріалу, який учням необхідно опанувати, то вимоги до навчального проекту абсолютно особливі. Робота над проектом – це реалізація наступних п'яти етапів:

1. Необхідна наявність соціально значимого завдання (проблеми) – інформаційної, практичної, дослідницької. Подальша робота над проектом є

вирішенням даної проблеми. Пошук соціально значимого завдання – одна з найважчих організаційних задач, яку потрібно розв'язувати учителеві – керівнику проекту разом з учнями-проектантами.

2. Виконання проекту розпочинається із планування дій щодо розв'язування проблеми, інакше кажучи, – з проектування процесу розв'язування проблеми, зокрема щодо визначення виду продукту і типу презентації.

3. Кожний проект обов'язково потребує дослідницької роботи учнів. Отже, відмінною рисою проекту є пошук відомостей, які згодом будуть опрацьовані, осмислені, презентовані учасниками проектної групи.

4. Результатом роботи над проектом є продукт. Загалом це засіб, який розробили учасники проектної групи для розв'язування поставленої проблеми.

5. Узагальнений проект повинен бути дуже переконливо показаний як найбільш прийнятний засіб розв'язування проблеми. На своїй завершальній стадії проект потребує презентації свого продукту.

Проектна діяльність вимагає від вчителя не стільки пояснення матеріалу, скільки створення умов для розширення пізнавальних інтересів дітей і на цій базі – можливостей самоосвіти в процесі практичного застосування знань. Саме тому вчитель-керівник проекту повинен володіти високим рівнем загальної культури, комплексом творчих здібностей. І насамперед розвинутою фантазією, без якої він не зможе бути генератором розвитку інтересів дитини і її творчого потенціалу. Найбільш складним є питання самостійності учнів, які працюють над проектом. Очевидно, що ступінь самостійності школярів залежить від багатьох факторів: вікових та індивідуальних особливостей, їх попереднього досвіду проектної діяльності, складності теми проекту, характеру стосунків у групі тощо. Для вчителя важливо уникати надмірностей в той чи інший бік, необхідно знайти баланс ступенів допомоги вчителя і самостійності учнів.

Розглянемо ***форми та засоби*** навчання алгоритмізації та основ програмування із використанням СКМ Maple. Форма організації навчання є зовнішнім вираженням діяльності учителя та учня, яка здійснюється у визначеному порядку і конкретному режимі. В межах форми навчання реалізуються зміст та методи навчання.

Навчання алгоритмізації та програмування успадковує все багатство української педагогіки: урочну систему, домашні завдання, практичну форму занять, контрольні роботи і т. п. Форми організації навчання програмування залежать від мети уроку, його конкретного етапу і типу: формування нових знань, формування вмінь, використання знань на практиці, повторення, систематизації знань і контролю їх засвоєння. При поясненні нового матеріалу, зазвичай, використовуються форми фронтальної роботи із класом, а при опрацюванні учнями знань та умінь програмування на різних етапах – групові та індивідуальні. В реалізації проектного та дослідницького методів доцільними є індивідуальна робота або робота в парах із врахуванням психологічних особливостей учнів. Під час індивідуальної роботи, як правило, відбувається самостійне засвоєння знань, формуються пізнавальність, самостійність і адекватність самооцінки. Робота в групах або парах супроводжується розвитком комунікативних здібностей учнів, почуття відповідальності за результат спільної роботи і здатності об’єктивної оцінки. Формування знань учнів, крім уроку відповідного типу, можна проводити на лекції, конференції, семінарі. З метою формування вказаних умінь школярів урок проводиться у формі навчально-практичного заняття.

Обов’язковим компонентом процесу навчання є контроль або перевірка результатів навчання. Суть перевірки результатів навчання полягає у визначенні рівня засвоєння знань та умінь, досвіду, формування відповідних компетентностей учнів, які повинні відповідати освітньому стандарту навчальної дисципліни. Контроль засвоєння знань може відбуватися у формі заліку, контрольного теоретичного і навчально-практичного заняття. Учні

отримують певні практичні завдання, за виконання яких звітують перед вчителем.

Розглянемо використання СКМ не тільки як засобу розв'язування прикладних задач, а і як середовища для вивчення алгоритмізації і програмування. Для реалізації такого підходу базовими є системи Matlab, Maple, Mathematica і MathCAD, що включають сотні математичних функцій як для чисельних, так і для аналітичних обчислень, мають зручний інтерфейс, засоби опрацювання текстових і графічних даних, мультимедійні додатки тощо (отже, маємо можливість широкого вибору системи для використання набору команд виконавця алгоритму).

При поглибленому вивченні алгоритмізації та програмування мови СКМ Maple є хорошим прикладом лінгвістичного забезпечення при розробці ефективних структурованих програм, який поєднує кращі традиції структурно-модульної технології з орієнтацією на математичні додатки і непрофесійних користувачів.

Уроки та заняття з алгоритмізації та програмування із використанням комп'ютерного математичного пакета Maple доцільно проводити із застосуванням мультимедійного проектора – як на етапі пояснення нового матеріалу, так і на уроках-практикумах. Одночасне виконання етапів розв'язування задач вчителем із використанням проектора і учнями на своїх комп'ютерах, що супроводжується концентрацією уваги школярів на виконанні завдання, значно підвищує продуктивність роботи, а також економить час при відповідях на однотипні запитання учнів, які виникають у них протягом уроку.

Таблиця 3

Подання базових алгоритмічних структур

(у формі НАМ і СКМ Maple)

Навчальна алгоритмічна мова [222]:	**СКМ Maple:**
Якщо <*умова*> то <*серія 1*> все	if <*умова*> then < *серія 1*> end if
Якщо <*умова*> то <*серія 1*> інакше <*серія 2*> все	if <*умова*> then < *серія 1*> else < *серія 2*> endif
Команда повторення – цикл «поки»:	
поки <*умова*> пц <*серія*> Кц	while<*умова*> do<*серія*> enddo
Команда повторення з параметром – цикл «для»:	
для *x* від $x_{поч}$ до $x_{кін}$[крок $x_{крок}$] пц <*серія*> кц	for<ім'я>from $x_{поч}$ by $x_{поч}$ *крок* to $x_{кін}$<*серія*> end do
Команда вибору:	
вибір при <*умова 1*>: <*серія 1*> при <*умова 2*>: <*серія 2*> …… при <*умова k*>: <*серія k*> [інакше <*серія*>] все	 if<*умова 1*>then<*серія 1*> elif<*умова 2*>then<*серія 2*> …… elif<*умова k*>then<*серія k*> else<*серія*> endif

До засобів навчання алгоритмізації та програмування із використанням СКМ Maple належать методичні розробки вчителя, інформаційне середовище, комп'ютери, програмне забезпечення Maple, збірники задач з програмування.

Важливим аргументом на користь застосування СКМ Maple в процесі вивчення алгоритмізації є і те, що в ході розробки і виконання програм користувач одержує структурований, зручний в користуванні та вивченні документ, який може бути оформлений у будь-якому вигляді: звіт, технічна документація, HTML-документ і т. д. Для прикладу наведемо подання базових алгоритмічних структур (розгалуження, повторення) навчальною алгоритмічною мовою (НАМ) і мовою СКМ Maple (табл. 3).

Розглянемо це на прикладах.

Приклад 1.

Знайти множину значень функції Y(x) = A(x) - sin(x),

де A(x) = $\begin{cases} 5x + 3, \text{коли } x > 3 \\ x - 1, \text{коли } x \leq 3 \end{cases}$

x змінюється від 0 до 10 з кроком 0,5.

Навчальна алгоритмічна мова [222]:

поч
дійсн *x,a,y*
для *x* від *0* до 10 крок *0,5*
пц
якщо *x>3* то *a:=5x+3*
інакше *a:=x-1*
все
y:=a-sin(x)
вивести *x,y*
кц
кін

СКМ Maple*:*

```
for x from 0 by 0.5 to 10 do
 if x > 3 then a := 5*x + 3
          else a := x-1;
fi;
 y := a - sin(x);
 end do;

 plot(y);
```

Із розглянутого прикладу бачимо, що набір операторів НАМ практично не відрізняється від операторів *M*-файла, але сприйняття учнем розв'язуваної задачі значно підвищується завдяки включенню в програму команди побудови графіка функції (включення функції **plot(y)** дає змогу отримати графік). Немає необхідності визначати й описувати тип використовуваних даних.

Приклад 2.

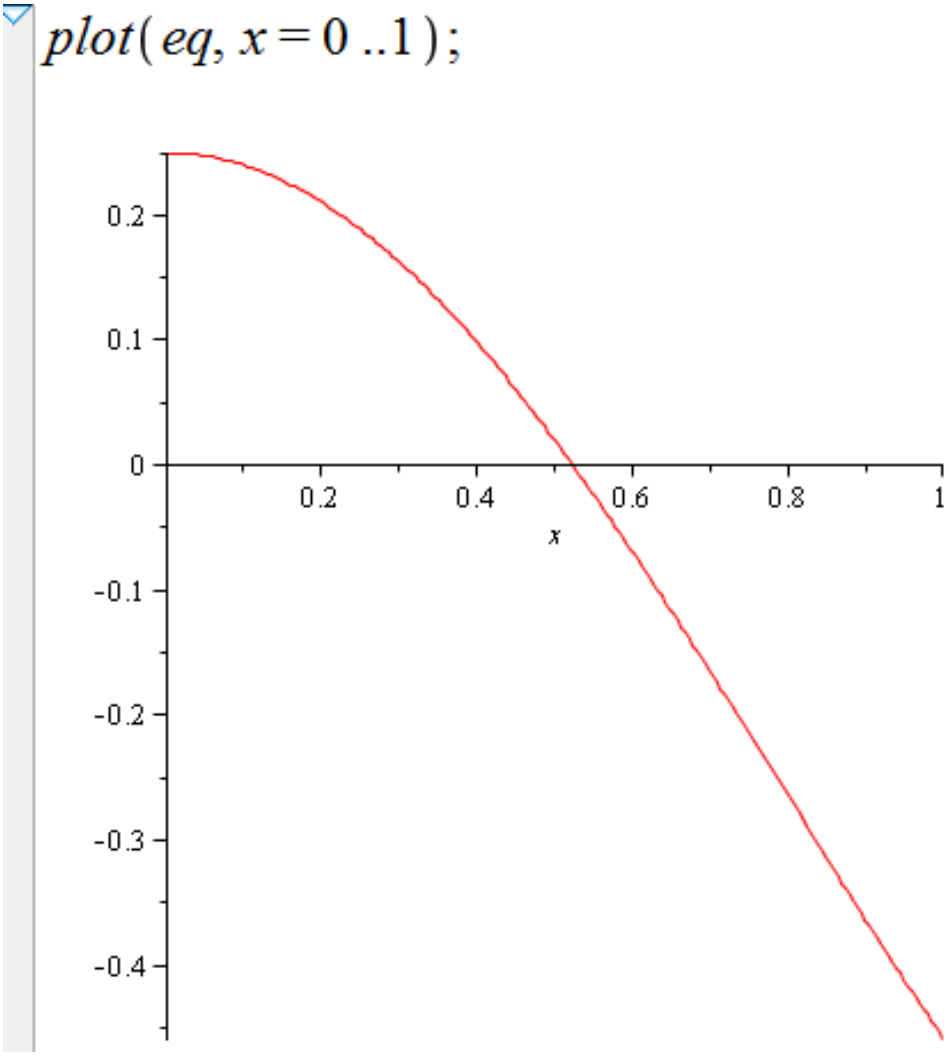
Знайти корені рівняння методом ділення відрізка навпіл. Як відомо, цей метод полягає у тому, що відрізок, який містить єдиний корінь даного рівняння, ділиться навпіл і з двох відрізків обирається той, що має різні знаки функції на своїх кінцях, тобто містить корінь. Цей вибраний відрізок ділиться навпіл і т. д. Процедура, що реалізує знаходження коренів, виглядає так:

```
restart :
HS :=proc(eq, x, xs, xe, e)
 local F, a, b, pres, c, result, n, i :: integer;
 a := xs·1. :
 b := xe·1. :
 pres := e·1. :
 n := ceil( ln( (b − a) / pres ) / ln(2) ) :
 F := unapply(eq, x) :
 for i from 1 to n do
 c := (a + b) / 2 :
 if evalb(F(c)·F(a) = 0) then i := n
 elif evalb(F(c)·F(a) < 0) then
 b := c
 else
 a := c
 fi
 od:
 c;
 end:
```

Тепер, якщо задати розв’язуване рівняння, точність розрахунків та інтервал пошуку, то можна одержати результат:

```
eq := cos(x)^2 − 0.75 :
 F := evalf(HS(eq, x, 0., 1., 0.001));
                0.5244140625
```

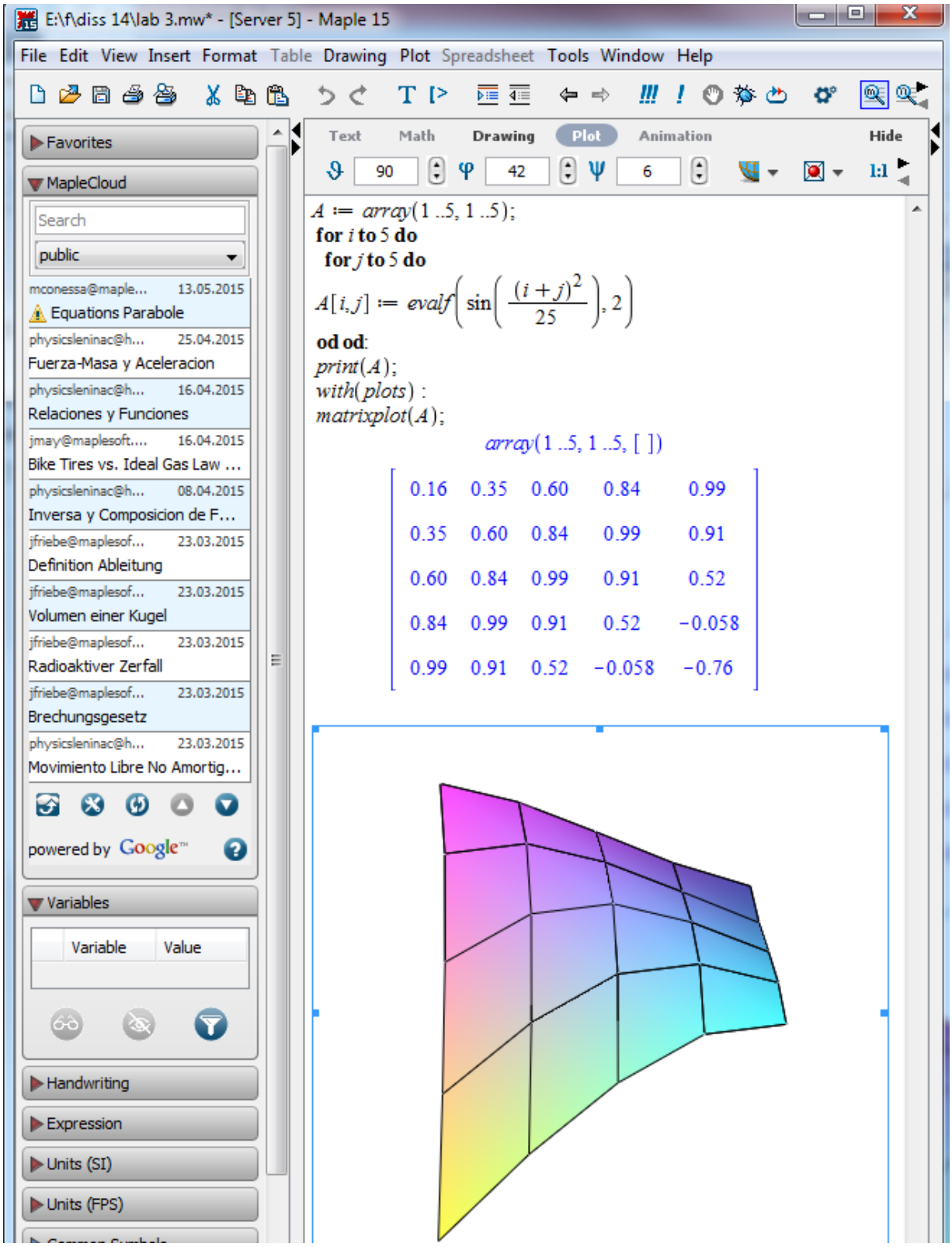
Перевіримо графічно:



Для більш детального уявлення про методичну систему навчання алгоритмізації та програмування наводимо приклад конспекту розробленого уроку інформатики з використанням комп'ютерного математичного пакету Maple [див. Додаток Д]. Також нами розроблені система практичних робіт [див. Додаток С] та тести для контролю успішності старшокласників при вивченні алгоритмізації та програмування [див. Додаток Ж].

Приклад 3.

Побудувати матрицю 5x5, знайти значення заданої функції в точках і відобразити графічно.



2.5. Система доцільно дібраних задач, спрямована на формування системно-логічного мислення старшокласників при вивченні алгоритмізації та програмування

Головною метою навчання є не лише озброєння учня певними знаннями, а й формування у них вмінь самостійно здобувати знання, тобто, врешті-решт, розвиток продуктивного мислення. Психологами доведено, що розвиток мислення того, хто навчається (учня, студента), визначається освітою і вихованням, відбувається в процесі освіти та виховання і є їх продуктом. Розвиток мислення не надбудовується над освітою і вихованням, метою яких є опанування учнями певних знань і певних вмінь, а органічно включається в освіту і виховання. Разом з тим розвиваючі можливості освіти і виховання реалізуються повною мірою лише за умови, що той, хто навчається, стає суб'єктом навчальної діяльності [277].

Забезпечивши опанування школярем предмета і мети діяльності, вчитель повинен уникати підміни власної діяльності учня своєю, незалежно від того, з якою її частиною він має справу: орієнтаційно-плануючою (добір способу і засобів), виконавчою (реалізація дібраних способів і засобів), контрольно-корегуючою (перевірка відповідності продукту діяльності її меті, внесенням поправок).

Разом з тим, навіть коли учні реалізують усі частини діяльності повністю значною мірою самостійно, не завжди вдається досягнути впливу освіти і виховання на розвиток мислення. Власна навчальна діяльність учня позитивно впливає на розвиток мислення лише тоді, коли учні у кожному конкретному акті навчання переборюють посильні труднощі спочатку у співробітництві з викладачем або ж з опорою на навчальний посібник (зона найближчого розвитку), а пізніше – самостійно (зона актуального розвитку). Навчання повинно опиратися не на вчорашні здобутки, проте бути спрямованим на майбутні досягнення, орієнтуючись не на минуле, а на сучасну спочатку

близьку, а згодом – віддалену перспективу. Лише у цьому випадку стимулюються процеси, які лежать в основі розвитку мислення.

У сучасній дидактичній теорії склалися передумови для застосування систем задач як засобу розвитку культури мислення учнів: із закріплення і контролю знань за допомогою задач у навчальному процесі вони стають способом здобуття цих знань, їх розширення і поглиблення, а також сприяють розвитку вищих психічних функцій (мислення, пам'яті, уваги). Найбільш детально вивчено питання розвитку основ теоретичного мислення на основі використання навчальних задач. Проте, наразі можна констатувати, що використання розвиваючого потенціалу задач є не до кінця вирішеною педагогічною проблемою. По-перше, не зафіксоване чітко поняття розвиваючої задачі, не вивчена детально її структура. По-друге, склалася думка, що такі задачі доцільно застосовувати не для всієї категорії учнів, оскільки є об'єктивно складними, з їх застосуванням пов'язують процеси розвитку творчої діяльності учнів. По-третє, використання їх у навчанні відбувається шляхом включення поодиноких задач у стандартну систему задач із досліджуваної теми і застосовується в додатковий час.

Проблема застосування задач для розвитку системно-логічного мислення може бути вирішена на основі формування спеціальних систем задач. Кожна задача системи повинна мати набір функцій, взаємопов'язаний з усіма складовими системи і підпорядкований досягненню поставлених освітніх цілей.

Розглянемо питання щодо використання терміна «задача» в педагогічній літературі. У педагогічних словниках задачею називається особливий вид завдання, що дається вчителем учням. Особливо виокремлюється лише термін «пізнавальна задача» як завдання, для розв'язання якої суб'єкту необхідні нові знання, способи дій і т. д. [251].

Завдання – це певна вимога здійснити якусь дію або отримати певний результат. При цьому можливі два випадки. Перший – учні вже знайомі з усіма необхідними їм діями та операціями і завдання даються для того, щоб

зміцнити їхні вміння, перевіривши цим самим вміння і навички, довести їх виконання до автоматизму. Такі завдання можна назвати тренувальними вправами. Другий випадок – учні не знайомі зі всіма діями та операціями, які потрібні для виконання завдання, такі завдання вже можна охарактеризувати як задачі.

В.І Загвязинський визначає задачу як «ситуацію, що вимагає від суб'єкта деякої дії, спрямованої на знаходження невідомого на основі використання його зв'язків з відомим. Джерелом задачі є проблемна ситуація: суб'єкт у своїй діяльності зустрічає перешкоду. Якщо суб'єкт усвідомив цю перешкоду і захотів її усунути, то він «увійшов» в проблемну ситуацію, прийняв її. Аналіз проблемної ситуації, виявлення її зв'язків, відношень, закріплених у мові, виражаються у вигляді задач» [80, с. 84].

З урахуванням вищесказаного називатимемо задачею таке завдання на знаходження якогось результату, коли послідовність дій для знаходження його результату не вказана, але учень повинен володіти кожною дією з цієї послідовності. Запитання – це завдання, сформульоване у запитальній формі і, на відміну від задачі, не містить специфічних даних, необхідних для його виконання. Вправа – це завдання, запропоноване у будь-якій формі, але спрямоване лише на відтворення в учнів умінь і навичок, якими вони повинні вміти володіти.

Г. Шолом зазначає, що на сучасному етапі великого значення набуває задачний підхід до процесу навчання, який в основному проявляється в концепції «навчання через задачі». На її думку, знати інформатику – означає вміти застосовувати знання при розв'язуванні задач і така позиція передбачає створення умов для оволодіння певним обсягом знань і умінь учнів через їхню активну пізнавальну діяльність [265, с. 120–125].

Як зазначає М.І Жалдак, щоби добре розв'язувати задачі, в тому числі пов'язані з алгоритмізацією і програмуванням, треба їх розв'язувати, причому якомога більше від найпростіших до якомога складніших. При цьому формуються відповідні знання, вміння, розвивається логічне мислення,

навички аналізу задач, інтуїція, евристичне бачення шляхів розв’язування задачі тощо. [74, с. 3-15].

Розв’язування навчальних задач взагалі й з інформатики зокрема, спрямоване на опанування учнями фахових і світоглядних знань та розвиток у них системно-логічного мислення. Останнє має певну структуру і його функціонування детерміноване певними правилами і законами, про що було сказано у першому розділі нашого дослідження.

Особливо сприятливі умови для власної навчальної діяльності учня, орієнтованої на подолання посильних труднощів, забезпечує використання системи різноманітних за змістом навчальних задач.

Задачі, які учні розв’язують під керівництвом вчителя чи самостійно, насамперед повинні бути за змістом дібрані так, щоб вони відповідали обсягу знань, якими повинен володіти учень за результатами попереднього навчання, розкривалась можливість цих знань та практичні застосування. Це визначає доцільність розв’язування задач, які:

− ілюструють, конкретизують і розвивають теоретичний і експериментально-демонстраційний матеріал;

− за належної адаптації можуть бути застосовані у власній навчальній діяльності в ЗОШ, школах поглибленого вивчення інформатики, коледжах тощо;

− значущі у світоглядному, пізнавальному, прагматичному відношеннях; визначають національні пріоритети у наукових здобутках та сучасних технологіях [277].

Загалом функції задач, які застосовуються в процесі навчання, можна умовно розбити на дві великі групи [277]:

1. Освітні – застосовуються для: опанування необхідних знань, умінь, навичок і способів діяльності, визначених стандартами освіти; пропедевтики вивчення подальшого матеріалу; закріплення й поглиблення раніше вивченого (у навчальних посібниках набір задач з даною функцією представлений

найбільш повно з орієнтацією на «середнього» учня; наявна велика кількість однорідних задач одного рівня складності).

2. Розвиваючі –це лише поодинокі у навчальних посібниках задачі, які часто можна охарактеризувати як задачі з невідомим алгоритмом розв'язування (необхідно знайти цей алгоритм шляхом «нестандартних» міркувань, зразки яких не зустрічалися учнем раніше).

Особливо треба звернути увагу на розвиваючу функцію, роль якої згідно з сучасними підходами до організації процесу навчання зростає. У зв'язку з цим нам видається необхідним уточнити, яку ж задачу потрібно вважати «розвиваючою», а яку – ні. Для відповіді на поставлене питання достатньо зрозуміти, що розвиває «розвиваюча» задача.

Позитивну динаміку в розвитку системно-логічного мислення (залежно від різних психологічних теорій) констатують в одному з напрямів процесу формування і розвитку: 1) системи знань; 2) нових узагальнених прийомів і способів розумової діяльності; 3) функцій мислення; 4) загального методу міркування; 5) системи розумових дій. Тому назвемо розвиваючою задачу, результат розв'язування якої передбачає деяке «зміщення» в одному із зазначених напрямів.

Зрозуміло, що в результаті розв'язування однієї задачі просування дуже мале. Тому говорити про досягнення розвиваючого ефекту можна лише в результаті проведеного навчання, організованого протягом деякого часу, яке включає цілеспрямовано створену систему розвиваючих задач. У ній кожна задача спрямована на розвиток одного із зазначених вище параметрів, який у конкретному випадку буде домінувати.

О.М. Кривонос вказує на те, що головною метою задачного підходу є організація викладачем процесу опанування знань шляхом структурування навчального матеріалу у вигляді послідовності задач, які мають певний логічний зв'язок одна з одною [118, с. 83–91].

Оскільки задачі у навчанні, як правило, взаємопов'язані між собою та з різними психологічними, методичними та іншими компонентами навчального

процесу, необхідний аналіз не окремо взятих задач, а їх системи. Система виникає тоді, коли певна кількість окремих елементів об'єднуються за наявністю певних взаємозв'язків між ними. Такі взаємозв'язки називають структурою системи. Структура – це внутрішній спосіб організації системи.

Проблема побудови сукупності задач (добору задачного матеріалу) постає перед учителем при підготовці до кожного уроку. Педагогу необхідно враховувати дуже багато факторів: особливості навчального матеріалу, вік учнів, їх інтелектуальні здібності, тип уроку, дидактичну мету уроку, застосовувані технології і засоби навчання, часовий фактор, індивідуальний стиль діяльності педагога, доступні навчальні посібники та ін. Важливою є і система принципів навчання, на яку спирається вчитель.

Враховуючи всі ці умови, необхідно скласти систему задач, що дозволяє досягти вагомих результатів навчання. Звичайно, в освітніх стандартах закріплені принципи добору змісту освіти, однак механізми реалізації цих вимог не описані, тому вони є дуже загальними і часто виявляються непридатними в педагогічній практиці. Наше завдання полягає в тому, щоб довести їх до рівня вимог з урахуванням спрямованості на розвиток системно-логічного мислення учнів.

Під системою задач будемо розуміти деяку кількість пов'язаних між собою задач, орієнтованих на досягнення конкретної дидактичної мети.

Як і будь-яка система, система завдань, використання якої сприяє розвитку культури мислення учнів, повинна мати такі інтегративні властивості, яких не мають її елементи окремо. Насамперед, на нашу думку, цією властивістю є спрямованість на розвиток в єдності всіх складових системно-логічного мислення учнів. Потрібно відзначити і такі якості системи задач, як ефективність, доцільність, відкритість для нового змісту і нових технологій. Система задач буде ефективною, якщо дотримуватись загальновідомих методичних вимог та принципів: науковості – відповідність змісту задач теоретичному матеріалу, який учні вже мають опанувати, а теоретичний матеріал повинен відповідати стану розвитку відповідної науки;

диференційованої реалізованості – система задач має бути розрахована на реалізацію рівневої диференціації в процесі навчання інформатики [109].

Підсумуємо сказане вище виокремленням основних етапів формування системи задач, що задовольняє перераховані принципи:

1. Визначення освітніх цілей, яких необхідно досягти в процесі вивчення певного блоку програмного матеріалу. Ці цілі конкретизуються залежно від специфіки досліджуваного предмета, вікових та індивідуально-особистісних особливостей учнів.

2. Визначення функцій (освітніх і розвиваючих) задач, спрямованих на досягнення зазначених цілей.

3. Визначення типів задач, що мають визначені функції.

4. Складання базового набору задач з освітніми функціями. Ця сукупність задач повинна відповідати вимогам і до типів включених задач, і до взаємозв'язків між ними, а також до всієї системи.

5. Зміна структури деяких задач базового набору для зміщення освітньої функції на розвиваючу (задоволення вимог, спрямованих на розвиток системно-логічного мислення), а також доповнення системи іншими задачами із домінуючою розвиваючою функцією.

Ці етапи відображають дидактичний рівень формування систем задач і потребують конкретизації з урахуванням специфіки предметної галузі. Тому розглянемо це на прикладі інформатики, а саме: розділу алгоритмізації та програмування.

З метою розвитку системно-логічного мислення під час розв'язування навчальних задач з алгоритмізації та програмування необхідно надати власній навчальній діяльності учнів характеру пошукової, враховуючи, що така навчальна діяльність стимулює розвиток мислення певного типу лише за умови її належної організації, певного структурування.

Таким чином, раціональний добір системи навчальних задач із алгоритмізації та програмування за змістом і складністю та раціональна організація їх розв'язування учнями повинні і можуть забезпечити комплексну

реалізацію функцій навчання у класах з поглибленим вивченням інформатики: фахову підготовку на основі опанування знань освітнього і виховного змісту, формування відповідних вмінь їх застосовувати у повсякденному житті і фаховій діяльності, розвиток мислення системно-логічного типу.

Раціональна організація процесу навчання має забезпечити комплексну реалізацію функцій навчання – освіту, виховання і розвиток мислення. Мається на увазі наступне. Учні старших класів, розв'язуючи задачі, повинні якісно засвоїти закономірності протікання алгоритмічних процесів, значущих пізнавально, технологічно та інформатично, навчитися враховувати ці закономірності та застосовувати відповідні закони у самостійному аналізі елементів програмування, а система розумових операцій, які використовуються під час розв'язування задач, будучи структурованою певним чином, має набути статусу системно-логічного мислення (див. розділ 1).

Зупинимось детальніше на опануванні знань про алгоритми. Навчальні задачі, які доцільно розв'язати учням і які повинні бути включені у НІС, повинні передусім мати зміст, який конкретизує і розвиває навчальний матеріал, поданий на лекційних заняттях або у підручнику чи в інших джерелах, самостійно знайдених учнем. Тому під час розв'язування задач особливу увагу треба приділити еволюції основних понять, які використовують в програмуванні для побудови алгоритму (масиви, цикли, рядки і т.д.). Також значна роль відводиться вивченню поняття виконавця алгоритму, наприклад, машини Тюрінга. Учні повинні усвідомити, що виокремлення тих чи інших складових програми (модуль, функція, тип даних тощо) проводиться відповідно до перерахованих вище понять. Розв'язування кожної задачі повинно розпочинатись з перевірки застосовності тих чи інших понять до умови задачі, перевірки, чи входить конкретна умова в обсяг певного поняття чи ні. У такому разі учні мають змогу усвідомити межі застосовності умов задачі, а умови наповнюються конкретним змістом: поняття як узагальнене знання про клас явищ конкретизується.

Це ж стосується застосування законів алгоритмізації та програмування високого рівня абстрактності. У цьому зв'язку під час розв'язування задач з програмування використання СКМ спрощує процес розв'язування задач, які вимагають застосування витонченого математичного апарату чи громіздких математичних перетворень. У іншому разі алгоритмічна реальність заміщується формально-математичними викладками, а те, заради чого власне і розв'язувалась задача – отримати знання про побудову раціонального алгоритму, відступить на задній план, залишиться поза увагою учнів і не буде ними засвоєна. Під час розв'язування задач із алгоритмізації та програмування потрібно мати на увазі, що програмування (ООП) – це реалізація ідей, побудова ідеальних моделей (алгоритмів), а не алгоритмізація спеціалізованих експериментальних та математичних методів. Тут основну увагу варто приділяти глибокому алгоритмічному аналізу задачі, перевірці застосовності до задачі тих чи інших алгоритмічних понять і структур, побудові ідеальної моделі (алгоритму), тоді як у формальній частині розв'язування перевагу надають простим формально-математичним методам або ж застосувати об'єкти ООП в СКМ (побудова графіків, використання функцій, складні обчислення тощо).

Це зауваження є істотним ще й з тих міркувань, що у зв'язку із профільною підготовкою учнів старших класів з поглибленим вивченням інформатики як зміст задач, так і методи їх розв'язування повинні «проектуватись» відповідно до профілю школи. Розв'язування задач з використанням СКМ надає широкі можливості розвитку світогляду учнів, підсилення їх інтересу до інформатики як науки, а також інтересу до майбутньої фахової діяльності шляхом формування змісту задач на основі фактів, значущих у науковому чи технологічному відношеннях, оцінювання практичної значущості отриманих внаслідок розв'язування формальних результатів. Не менш важливим щодо цього є відображення у змісті задач історії програмування, звернення до історії побудови алгоритмів, в процесі інтерпретації результатів розв'язування.

Останнє дає змогу довести до свідомості учнів факти, які свідчать про важливу роль вітчизняних та зарубіжних вчених у розвитку алгоритмізації та програмування, що відіграє важливу роль у формуванні національної самосвідомості. У цьому зв'язку лише згадаємо про роль Леонардо да Вінчі (перший у світі ескізний рисунок тринадцятирозрядного десяткового підсумовуючого пристрою), Віктор Глушков (теорія цифрових автоматів), Олександр Щукарєв («Машина логічного мислення»).

Ю.С. Рамський зауважує, що система прикладних задач, які доцільно пропонувати учням, повинна відповідати ще наступним вимогам: по-перше, у змісті задач мають бути відображені реальні об'єкти, процеси, явища, актуальні проблеми з різних галузей знань, техніки і виробництва. По-друге, доцільно добирати ті задачі, процес розв'язування яких сприяє розкриттю практичної значущості вивчених методів, усвідомленню того, що добір методу, засобів розв'язування залежить від особливостей поставленої задачі. Корисно також пропонувати практичні задачі, математичні моделі яких «чутливі» до похибок у вхідних даних. Головна вимога до такого класу задач полягає в тому, щоб навчальна діяльність була максимально наближеною до діяльності інформатика-дослідника [201].

Розв'язування задач з програмування взагалі та алгоритмізації як розділу інформатики зокрема відіграє першорядну роль у розвитку мислення учнів. У цьому зв'язку під час розв'язування задач важливою є організація розумової діяльності згідно із закономірностями побудови алгоритмічної теорії, тобто приведення структури узагальненого способу навчального пізнання у відповідність до структури наукового пізнання. Цей висновок більш-менш враховується під час вивчення алгоритмізації та програмування за підручником або конспектом лекцій у зв'язку із реалізацією принципу генералізації навчального матеріалу. Однак структура узагальнених способів розв'язування задач з програмування лишається далекою від структури процесу пізнання у інформатиці. Внаслідок цього втрачається єдність у вивченні і застосуванні

знань про інформатичні явища, що негативно впливає на результати вивчення інформатики, на розвиток в учнів системно-логічного мислення.

Можливості застосування СКМ Maple для розв'язування різноманітних алгоритмічних задач з використанням об'єктів величезні. Учень, використовуючи цей пакет, розв'язує поставлену перед ним задачу, тож у нього не виникає психологічного бар'єру у застосуванні інформатичного та математичного апарату. У школярів, які поглиблено вивчають алгоритмізацію та програмування, покращується сприйняття абстракцій і розуміння навчального матеріалу, формуються інформатичні компетентності. Розв'язування задач прикладного характеру з використанням ООП в таких системах надає знанням і вмінням учнів практично значущого характеру та сприяє формуванню в них системно-логічного мислення.

У нашому дослідженні значну увагу приділили практичним роботам, які займають чільне місце у системі задач, спрямованих на розвиток системно-логічного мислення старшокласників у процесі поглибленого вивчення алгоритмізації та програмування.

Практичні роботи – це проведення учнями за завданням учителя дослідів з використанням приладів, застосуванням інструментів та інших технічних пристосувань, тобто вивчення учнями предмета за допомогою спеціального обладнання [277].

Одна з цілей нашого дослідження полягає в розробці набору практичних завдань, спрямованих на формування навичок системно-логічного мислення старшокласників у процесі поглибленого вивчення алгоритмізації та програмування із застосуванням ООП (на прикладі СКМ Maple). Оскільки вказане мислення розвивається тільки в діяльності, а єдиною можливістю для інформаційної діяльності учнів є розв'язування задач, то одним із шляхів вирішення поставленої проблеми можна вважати розробку та впровадження в процес навчання системи практичних завдань, спрямованої на формування таких навичок системно-логічного мислення, як аналіз, синтез, порівняння, узагальнення.

Запропоновані практичні роботи призначені для старшокласників, інтереси яких спрямовані на вивчення алгоритмізації і програмування із застосуванням ООП [див. Додаток В]. Передбачається знання основ алгоритмізації і структурного програмування, а також базові користувацькі навички роботи із додатками Windows.

Загальна характеристика змісту системи практичних робіт:

1). опис завдання і докладний розгляд його складових;

2). набір задач різної складності, що сприяють формуванню навичок системно-логічного мислення.

Пропонована система практичних робіт дозволяє послідовно (від простого до складного) формувати в учнів уявлення про об’єктно-орієнтоване програмування і сприяти розвитку навичок системно-логічного мислення при розробці проектів. Добір задач передбачає реалізацію вимог формування в учнів узагальнених умінь працювати з різноманітними проектами, незалежних від предметної галузі, мовних, програмних або технічних засобів.

Виокремимо такі методичні умови, які сприяють формуванню цих навичок на основі використання СКМ Maple: поетапний розвиток, робота з навчальною книгою, проведення практичних робіт, розв’язування задач проблемного характеру, складання абстрактних схем, діаграм, таблиць порівняння тощо.

Як було зазначено вище, навчання інформатики загалом і алгоритмізації та програмування зокрема характеризується великою диференціацією знань, здібностей, інтересів, які зумовлені сучасним рівнем інформатизації суспільства. Різний рівень складності задач, наведених в практичних роботах, дозволяє здійснювати диференційований підхід до навчання.

Розглянемо основні типи алгоритмічних структур: лінійні, циклічні, з розгалуженнями.

Простежимо на прикладі обчислення значення функції в системі Maple (рис. 2.19), використовуючи розгалуження.

Задаємо функцію (1), застосовуючи об’єкт *plot(1)*, виконуємо побудову графіка функції. Присвоївши значення змінній *x* і описавши функцію, використовуючи структуру розгалуження, отримуємо значення функції в заданій точці.

Також в Maple доцільно описувати алгоритми з розгалуженням різних типів, використовуючи вбудовані об’єкти.

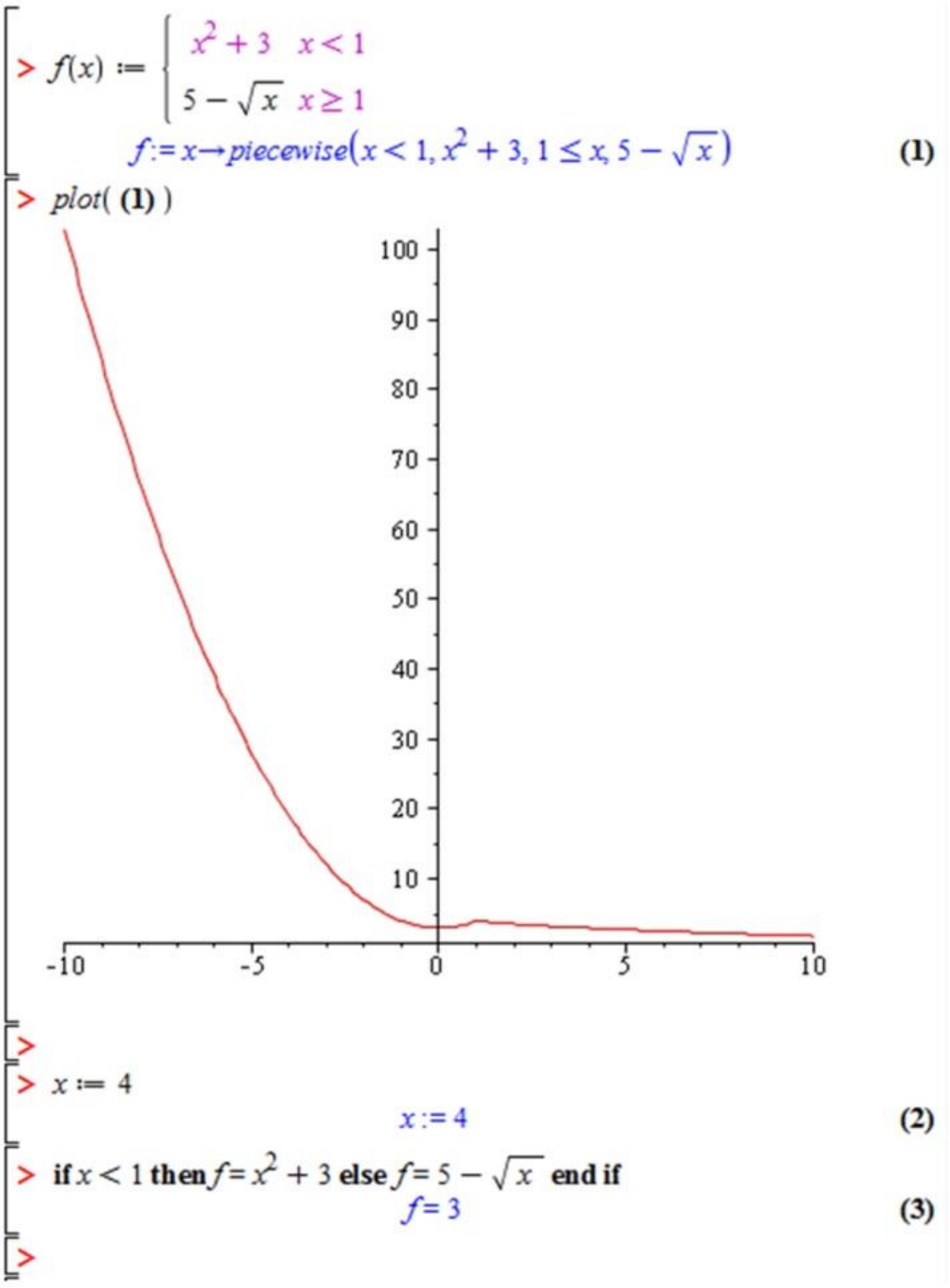


Рис. 2.19. Обчислення значення функції в системі Maple

СКМ Maple можна ефективно використовувати для розв’язування задач всіх типів циклічних алгоритмів, які є стандартними для мов програмування.

Наприклад, розв’язуючи задачу 1 про вибірку простих чисел від 1 до n, можна використати як об’єкт вбудовану функцію під назвою *Maple IsPrime*, яка повертає значення істини, якщо її аргумент простий та непомилковий. Для розв’язування цієї задачі використовуємо цикл з параметром та команду розгалуження. Описуємо процедуру, як *prosti*.

Цикл повторюватиметься з початку, поки змінна *i* не набуде всіх значень та стане рівною *n*. На рис. 2.20 показано результати виконання програми для *n=10*.

```
> prosti := proc(n)
  local i;
  for i from 1 to n do
  if isprime(i) = true then print(i); fi;
  od;
  end;

prosti := proc(n)                                   (1)
    local i;
    for i to n do
        if isprime(i) = true then print(i)
        end if
    end do
end proc
=
> prosti(10);
                        2
                        3
                        5
                        7                           (2)
```

Рис. 2.20. Вибірка простих чисел від 1 до 10 в системі Maple

Задача 2.

Ще один приклад використання циклу з параметром та вбудованого оператора композиції для складання і обчислення ланцюгового радикала. Цикл повторюватиметься з початку, поки змінна *k* не набуде всіх значень та стане рівною 10. На рис. 2.21 показано виконання програми.

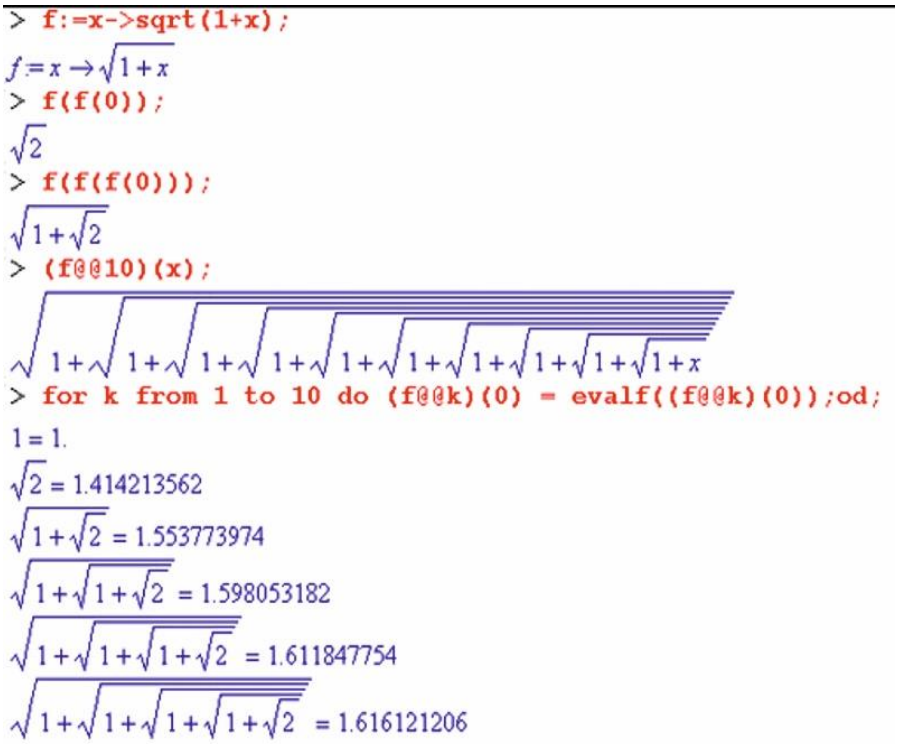
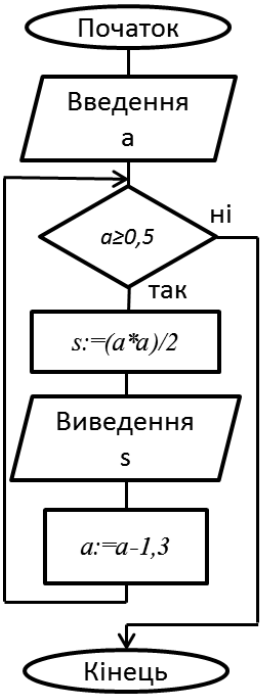


Рис. 2.21 складання і обчислення ланцюгового радикала в системі Maple

Задача 3.

Розробіть алгоритм і програму, за допомогою яких можна обчислити і вивести на екран площі прямокутних рівнобедрених трикутників, якщо відомо, що катет першого дорівнює a (a>2) см, катет кожного наступного зменшується на 1,3 см. Обчислення площ припиняється, коли катет трикутника стає меншим 0,5 см.

Будуємо алгоритм:



У програмі *Maple:*

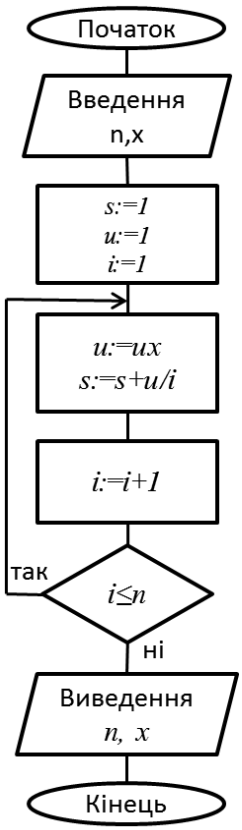
```
s := 0 :
a := 6 :
while a ≥ 0.5 do s := a·a/2 ; print("s=", s); a := a − 1.3 : end do:
```

```
"s=", 18
"s=", 11.04500000
"s=", 5.780000000
"s=", 2.205000000
"s=", 0.3200000000
```

Задача 4. Розробіть алгоритм і програму обчислення виразу:

$$1 + x + \frac{x^2}{2} + \frac{x^3}{3} + \frac{x^4}{4} + \ldots + \frac{x^n}{n}$$

Будуємо алгоритм:



У програмі *Maple:*

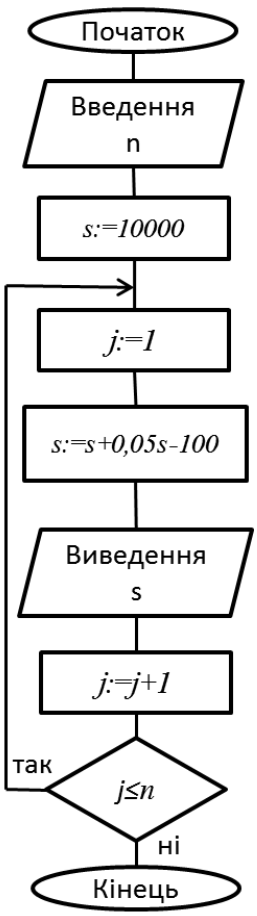
```
u := 1 :
n := 4 :
x := 3 :
S := 1 :
for i from 1 to n do u := u·x : S := S + u/i : end do:
print("сума =", S);
```

```
"сума =", 151/4
```

Задача 5.

Первинний вклад у банку становить 10000 грн. За кожний рік нараховується 5 % річних і кожного року з накопиченої суми знімається 100 грн. Розробіть алгоритм і програму визначення вкладу за кожний з перших n років.

Будуємо алгоритм:



У програмі *Maple:*

```
n := 6 :
s := 10000 :
for j from 1 to n do s := s + 0.05 · s − 100 : print("за", j, "-ий рік=", s)
    end do;
```

```
                    10400.00
        "за", 1, "-ий рік=", 10400.00
                    10820.0000
        "за", 2, "-ий рік=", 10820.0000
                    11261.00000
        "за", 3, "-ий рік=", 11261.00000
                    11724.05000
        "за", 4, "-ий рік=", 11724.05000
                    12210.25250
        "за", 5, "-ий рік=", 12210.25250
                    12720.76512
        "за", 6, "-ий рік=", 12720.76512
```

На цих простих прикладах ми бачимо, що синтаксис Maple схожий із синтаксисом мов програмування високого рівня. Звичайно, є деякі відмінності, проте вони легко сприймаються користувачем, а об’єкти, які є в Maple, полегшують складання алгоритмів та написання програм. Інші приклади доцільно дібраних задач наведені у практичних роботах [див. Додаток В].

Таким чином, резюмуючи вищесказане, зазначимо, що система задач – це функціональна за своїм характером система, спрямована на одержання наступного результату: розвиток культури мислення учнів та досягнення системи освітніх цілей. Кожен компонент такої системи вносить свою частку

сприяння в отриманні загального результату. Система задач має також властивість цілісно відображати навчальний процес і тому відкриває можливість комплексного підходу до його вдосконалення, при якому зміни окремих компонентів так чи інакше відбиваються й іншими компонентами, й загальними характеристиками процесу.

Основною функцією сучасної парадигми освіти є покращення таких її параметрів, які розвивають культуру мислення і дають змогу вийти за рамки обмеженого процесу оволодіння теоретичним матеріалом. Тому раціональний добір системи навчальних задач із загальної алгоритмізації та програмування за змістом і складністю та раціональна організація їх розв’язування учнями повинні і можуть забезпечити комплексну реалізацію функцій навчання у класах з поглибленим вивченням інформатики: фахову підготовку на основі засвоєння знань освітнього і виховного змісту, формування відповідних вмінь їх застосовувати у повсякденному житті і фаховій діяльності, розвиток мислення системно-логічного типу.

Висновки до другого розділу

Другий розділ дисертаційного дослідження присвячений розробці власної методичної системи навчання алгоритмізації та програмування старшокласників на поглибленому рівні із застосуванням ІКТ «ІнфоНІС» та MOODLE, використовуючи як засіб програмування СКМ Maple.

Для формування інформатичних компетентностей учнів старших класів у процесі навчання алгоритмізації та програмування передбачено систематичне педагогічно доцільне і виважене використання засобів сучасних ІКТ загального, спеціального призначення (MOODLE, ІнфоНІС), зокрема навчального (СКМ Maple). На основі «ІнфоНІС» та MOODLE розроблено дидактичне забезпечення курсу «Основи алгоритмізації та програмування», використання якого учнями на різних етапах навчання (під час підготовки до занять, виконання лабораторних робіт, розробки навчальних проектів) сприяє розвитку системно-логічного типу мислення при розв’язуванні задач навчального та професійного спрямування.

Запропонована методична система навчання алгоритмізації та основ програмування старшокласників в процесі поглибленого вивчення інформатики в ЗОШ спрямована на здобуття учнями необхідних знань, умінь та навичок, які є базою для розуміння можливостей та обмежень використання персональних комп’ютерів і програмного забезпечення у житті суспільства. Вивчення курсу алгоритмізації сприяє здобуттю учнями фундаментальних знань в галузі інформатики; введення ООП дає змогу адаптувати здобуті знання до швидких змін у сфері нових ІКТ, що, відповідно, допомагає на якісно новому рівні використовувати ці технології у навчальному процесі та покращувати міжпредметні зв’язки.

Детально охарактеризовано навчальні середовища («ІнфоНІС», MOODLE і Atutor), з’ясовано їх основні позитивні моменти, зроблено порівняльний аналіз систем управління навчальним процесом MOODLE і Atutor, використання яких дає змогу педагогам легко організовувати різні курси навчання, під час яких учні отримують зручне навчальне середовище.

Таким чином, можна зробити такі висновки.

− Системи з відкритим кодом дають можливість вирішувати ті ж задачі, що й комерційні системи, але при цьому у користувачів є можливість доопрацювання та адаптації конкретної системи до своїх потреб та поточної освітньої ситуації.

− Більшість систем з відкритим кодом є крос-платформними рішеннями і не прив’язані до конкретних операційних систем та конкретних WEB-браузерів.

− Використання комерційних систем управління навчальними курсами не доступне більшості ЗНЗ і ВНЗ через високу вартість, необхідність продовження ліцензії на кожен навчальний рік, прив’язку вартості ліцензій та їх продовження до кількості користувачів системи.

− Сучасні тенденції розвитку LMS спрямовані в бік універсалізації та збільшення функціональності систем. За своїми функціями найбільш передові системи не поступаються комерційним аналогам, а деякі навіть перевершують їх.

− Системи управління навчальними курсами з відкритим вихідним кодом дають змогу реалізувати той набір функціональних можливостей, що й комерційні рішення з істотно меншими економічними витратами.

Проведено аналіз найбільш популярних СКМ, вивчення яких дає змогу учневі сформувати алгоритмічний стиль мислення, опанувати сучасні ІКТ (побудову алгоритмічних моделей, вихідних мультимедійних документів у вигляді графіків та анімаційних кліпів) та оволодіти розвиненим інструментарієм програмування для розв’язування прикладних завдань, використовуючи ООП. Використання СКМ у навчальному процесі ЗОШ сприяє не тільки підвищенню загального рівня знань у галузі інформатики, а й активізації інтересу старшокласників до застосування ІКТ під час розв’язування тематичних завдань з інших навчальних дисциплін.

Це можливо тому, що СКМ:

– мають розвинений інструментарій програмування, який містить повний набір базових операторів, повністю відповідний набору операторів мови Паскаль;

– наочно відображають вміст простих змінних, структурованих даних, що підвищує розуміння процесу їх опрацювання;

– можуть бути повноцінним інструментарієм для реалізації базових і типових алгоритмів з використанням ООП, під час вивчення алгоритмізації та програмування в курсі інформатики для старшокласників.

До курсу «Основи алгоритмізації та програмування» розроблено систему практичних робіт (в яких включено доцільно дібрані задачі репродуктивного, частково-пошукового, дослідницького, творчого спрямування; задачі практичного змісту, завдання із суміжних дисциплін тощо), використання яких спрямовано на набуття старшокласниками досвіду навчально-пізнавальної, предметно-практичної, дослідницької, творчої діяльності, що є необхідною умовою формування інформатичних компетентностей. Виконання учнями завдань із алгоритмізації та програмування у середовищі СКМ сприяє формуванню в них не лише інформатичних компетентностей, а й розвитку системно логічного мислення.

Основні результати розділу представлено в опублікованих працях [277; 278; 275].

РОЗДІЛ 3.
ЕКСПЕРИМЕНТАЛЬНА РОБОТА З ВИЗНАЧЕННЯ ЕФЕКТИВНОСТІ ВИКОРИСТАННЯ РОЗРОБЛЕНОЇ МЕТОДИЧНОЇ СИСТЕМИ

3.1. Мета та зміст експериментальної роботи

Для підтвердження ефективності використання розробленої методичної системи навчання алгоритмізації та основ програмування старшокласників було проведено ***педагогічний експеримент***. Його ***метою*** стала практична перевірка ефективності впровадження в процес навчання інформатики на поглибленому рівні учнів старших класів нових організаційних методів, прийомів і форм застосування комп'ютерів, підґрунтям яких є використання сучасних комп'ютерних математичних пакетів. Експериментальна робота була розподілена на кілька етапів відповідно до технології підготовки та проведення експерименту [108].

Гіпотеза проведеного дослідження полягала в наступному: методично обґрунтоване цілеспрямоване використання СКМ для навчання алгоритмізації та програмування сприяє розвитку системно-логічного мислення та особистості учнів, активізації навчального інтересу в умовах сьогодення і тим самим формуванню інформатичних компетентностей, що підвищує ефективність навчального процесу. Педагогічний експеримент, який перевіряє ефективність розробленої методичної системи, за своєю природою довготривалий і трудомісткий. Впродовж нього здійснюється педагогічне дослідження, пов'язане з аналізом результатів навчання контрольної та експериментальної груп, перша з яких навчається за традиційною методичною системою, а друга – за авторською методичною системою.

Апробація методичної системи в навчальному процесі старшої школи протягом трьох років показала достатньо високий рівень засвоєння навчального матеріалу: майже 75% учнів складали правильну послідовність дій для розв'язання контрольних завдань [див. Додаток Є]. Спостерігались

зацікавленість в опануванні навчального матеріалу, прагнення досягти правильного розв'язку, продукування власних алгоритмічних задач. Одним із позитивних показників є виявлення бажання старшокласників повернутись до вивченого матеріалу, пов'язаного з алгоритмізацією та програмуванням, і продовжувати роботу з ним.

Як відомо, педагогічний експеримент складається з кількох етапів: *констатувальний* та *пошуковий* етапи, основним завданням яких є збирання та аналіз необхідних емпіричних даних для уточнення гіпотези дослідження, а також *формувальний* етап, на якому будується теоретична модель і проводиться перевірка її ефективності.

Для встановлення необхідності розробки та реалізації на практиці авторської методики навчання алгоритмізації та основ програмування старшокласників у процес поглибленого вивчення інформатики на початковому етапі експериментальної роботи проведений *констатувальний* експеримент (2010–2011 рр.), який був спрямований на розв'язування наступних завдань:

1) аналіз стану, потреб та перспектив практики, орієнтованої на підготовку учнів старших класів з питань використання об'єктно-орієнтованої парадигми в навчанні алгоритмізації та програмування (на прикладі СКМ Maple);

2) опитування учителів, які навчають у класах з поглибленим вивченням інформатики, та виявлення проблем навчання, пов'язаних з використанням ООП, а також причин, які зумовили їх появу;

3) визначення рівня володіння старшокласниками уміннями та навичками алгоритмізації і програмування, уміннями перетворювати і синтезувати знання, аналізувати, визначати істотні зв'язки і співвідношення, які існують в певному явищі, розробляти алгоритми, складати програми, виправляти, перевіряти і оцінювати правильність результату, що сприяє розвитку системно-логічного мислення школярів.

Для вирішення *першого завдання констатувального етапу* педагогічного експерименту було проаналізовано психолого-педагогічні та методичні джерела із зазначеної проблеми; розглянуто досвід організації навчання в школі на основі ООП (на прикладі використання СКМ Maple). Було з'ясовано, що об'єктно-орієнтована парадигма навчання алгоритмізації та програмування слабко опанована методистами.

Для вирішення *другого завдання констатувального етапу* педагогічного експерименту – з'ясування у вчителів інформатики труднощів у процесі вивчення аналізованого розділу з використанням СКМ Maple і причин такого стану – проведено анкетування. У дослідженні взяли участь 11 учителів у класах із поглибленим вивченням інформатики ЗОШ Тернопільської області. Їм була запропоновано анкету**,** питання якої дали змогу з'ясувати такі аспекти:

1) теми, які викликали найбільші труднощі і які згодом потребували найбільшого доопрацювання;

2) актуальність розробки методичної системи навчання розділу «Алгоритмізація та основи програмування» із використанням СКМ Maple;

3) причини виникнення труднощів під час вивчення цього розділу інформатики.

На *констатувальному* етапі педагогічного експерименту визначено рівень володіння старшокласниками уміннями та навичками алгоритмізації і програмування і встановлено рівень розвитку системно-логічного мислення школярів, тобто вирішено *третє завдання* цього етапу. Для цього ми використовували метод розв'язування задач та їх аналіз.

Результати *констатувального експерименту* виявили слабке використання ООП до навчання алгоритмізації і програмування старшокласників та необхідність розробки методичної системи такого навчання, а також недостатній рівень сформованості системно-логічного мислення школярів.

На другому, *пошуковому* етапі педагогічного експерименту проведено розробку компонентів власної методичної системи навчання розділу

«Алгоритмізація та програмування» із застосуванням СКМ Maple (детально описана у підрозділах 2.4, 2.5). Результати роботи оформлені у вигляді створеного у середовищі Moodle навчального курсу, системи дібраних задач, конспектів уроків із вказаної теми, лабораторного практикуму та підготовленого пакету тестів для перевірки знань, умінь і навичок учнів. У реалізації цього етапу експерименту брали участь учні 10–11 класів. Основними формами роботи були уроки засвоєння нових знань та практичні роботи. Крім того, під час виконання завдань на практичних заняттях учні працювали групами.

Третій етап експерименту – *формувальний* – був спрямований на перевірку гіпотези та основних концептуальних положень дослідження; апробацію та перевірку ефективності використання компонентів методичної системи навчання алгоритмізації та програмування; порівнянні результатів навчання в контрольному класі з результатами навчання на основі розробленої нами методичної системи навчання алгоритмізації і програмування на засадах об’єктно-орієнтованої парадигми (із застосуванням СКМ Maple) в експериментальному класі; формулювання загальних висновків виконання дисертаційного дослідження.

Для підвищення достовірності результатів експериментальної роботи, як доповнення до вказаних форм педагогічного експерименту, проведено систему контрольних робіт, метою якої була перевірка рівня опанування учнями знань, умінь та навичок здійснювати діяльність, пов’язану з алгоритмізацією та програмуванням. Для констатації результатів експерименту опитано учнів із тем, безпосередньо пов’язаних із пропонованим навчальним курсом [див. Додаток Е].

На основі аналізу результатів зроблено висновок про те, що запропонована методика дозволяє досягти мети навчання курсу інформатики і дає учням можливість засвоїти основні поняття і принципи алгоритмізації і програмування за допомогою ООП (на прикладі застосування СКМ у навчальному процесі ЗОШ). Важливо, що експеримент виявив підвищення

здатності старшокласників до модифікації цієї парадигми програмування залежно від специфіки умов і процесу розв’язання поставленої задачі.

Також важливою ціллю третього етапу педагогічного експерименту була демонстрація впровадження розробленої методики навчання алгоритмізації та основ програмування із застосування СКМ Maple у навчальний процес в старших класах ЗОШ. Для її досягнення були визначені необхідні знання, вміння та навички, які повинні здобути учні в процесі вивчення прийомів алгоритмізації та програмування на основі ООП, а саме:

- знання понять ідентифікаторів, описів, змінних, функцій і процедур;
- знання основних типів даних;
- знання елементарних понять математичної логіки;
- знання сумісності типів формальних і фактичних параметрів, вміння правильно організовувати їх;
- знання мови програмування та її основних понять;
- вміння розробляти та оформляти програми;
- вміння оформити алгоритм розв’язання задачі;
- навички роботи з підпрограмами;
- знання етапів технології програмування;
- навички налагодження програм;
- вміння реалізовувати всі частини програми;
- вміння застосовувати нові ІТ в процесі розв’язування практичних задач.

Крім того, для досягнення мети, учням необхідно достатньою мірою оволодіти математичними методами, що дозволяють алгоритмізувати процес розв’язування поставленої задачі, знати відмінності різноманітних СКМ з точки зору стислості, економічності та ефективності алгоритмів, які необхідно реалізувати на їх основі.

Формувальний етап експерименту проводився в 10–11 класах Лозовецької ЗОШ. Під час цього процесу було виявлено, що в учнів сформувалися навички програмування, вміння використовувати конструкції

мови і математичні постановки задач. Робота була організована за допомогою вироблених методичних прийомів із використанням опрацьованих дидактичних матеріалів і спеціальних завдань. Для визначення рівня знань та умінь старшокласників із вищеперелічених тем проводилися спеціальні опитування, а також контрольні роботи.

3.2. Аналіз результатів педагогічного експерименту

Кількість контрольних робіт була достатньо великою, тому в нашому дослідженні лише відзначимо, що всі вони підтвердили високу ефективність використання об'єктно-орієнтованого підходу до навчання алгоритмізації та програмування (на прикладі застосування СКМ Maple).

Наведемо деякі числові дані, які засвідчують суттєву перевагу пропонованого нами навчального підходу. Про ефективність розробленої методики дозволяють говорити результати та аналіз алгоритмів і програм, створених учнями контрольної та експериментальної груп. Для перевірки гіпотези про відсутність відмінностей між рівнями знань учнів контрольних та експериментальних груп, на початку навчального року проведено «нульову» контрольну роботу, результати якої опрацьовані статистично. Крім того, збалансовано інші фактори, що впливають на процес навчання: кількісний склад учнів в експериментальних і контрольних групах істотно не відрізнявся; заняття проведені одним і тим же викладачем.

Перевірено достовірність гіпотези про відсутність, зі статистичного погляду, відмінностей між рівнями знань учнів контрольних й експериментальних груп за результатами підсумкового контролю з курсу алгоритмізації та програмування. Для цього використано χ^2 – критерій Пірсона.

Гіпотеза H_0: рівень знань учнів у контрольних групах не відрізняється від рівня знань учнів в експериментальних групах, тобто запропонована методика не вплинула на рівень навчальних досягнень учнів.

Гіпотеза H_1: рівень знань учнів у контрольних групах відрізняється від рівня знань учнів в експериментальних групах, тобто рівень навчальних досягнень змінився завдяки впровадженню запропонованої методики.

Значення досліджуваного критерію Пірсона обчислене за формулою:

$$\chi^2_{емп} = \frac{1}{n_1 n_2} \sum_{i=0}^{C-1} \frac{(n_1 Q_{2i} - n_2 Q_{1i})^2}{Q_{1i} + Q_{2i}},$$

де

Q_{1i} − кількість учасників контрольної групи, які набрали *i* балів;

Q_{2i} − кількість учасників експериментальної групи, що набрали *i* балів.

Результати обчислення статистики χ^2 названих вибірок подано в табл. 4.

Таблиця 4

Обчислення χ^2 для контрольної та експериментальної груп після формувального етапу експерименту

I	***Q*1*i***	***Q*2*i***	***S*12*i***
0 (низький)	16	12	1883868,18
1 (середній)	20	24	593558,64
2 (високий)	19	20	119230,23
$\chi^2_{емп}$			**8.38**

Із таблиці значень χ^2 для рівня значущості α=0,05 і кількості ступенів свободи $v = C - 1 = 2$ знаходимо критичне значення $\chi^2_{крит} = 5{,}99$.

Оскільки отримане значення $\chi^2_{емп} > \chi^2_{крит}$ (8,38 > 5,99), тобто гіпотеза H_0 спростована, натомість підтверджена гіпотеза H_1. Це означає, що достовірність розходжень характеристик експериментальної та контрольної груп за результатами підсумкового контролю з курсу «Основи алгоритмізації та програмування» становить 95 %.

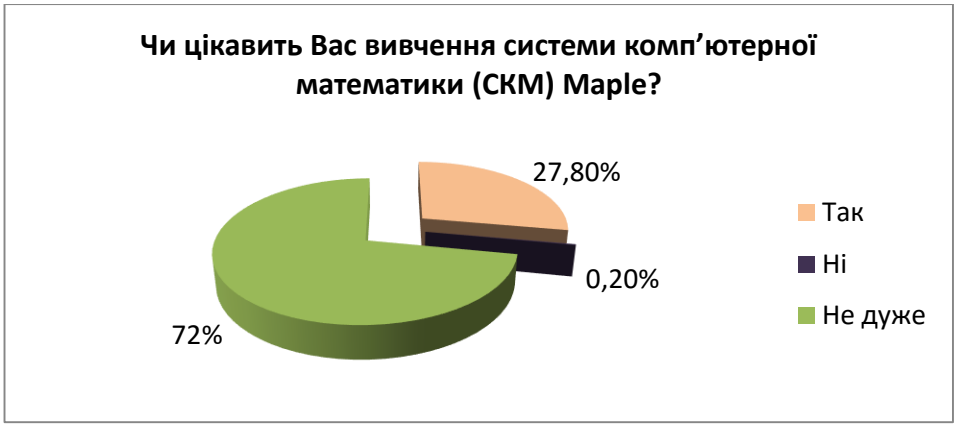
За допомогою χ^2 − критерію Пірсона визначено достовірність відмінностей між рівнями знань учнів контрольних та експериментальних груп після формувального етапу.

Також в межах педагогічного експерименту досліджувалося питання про вплив використання математичного пакета Maple на професійну орієнтацію старшокласників. Робота проводилась в 11 класі Острівської ЗОШ.

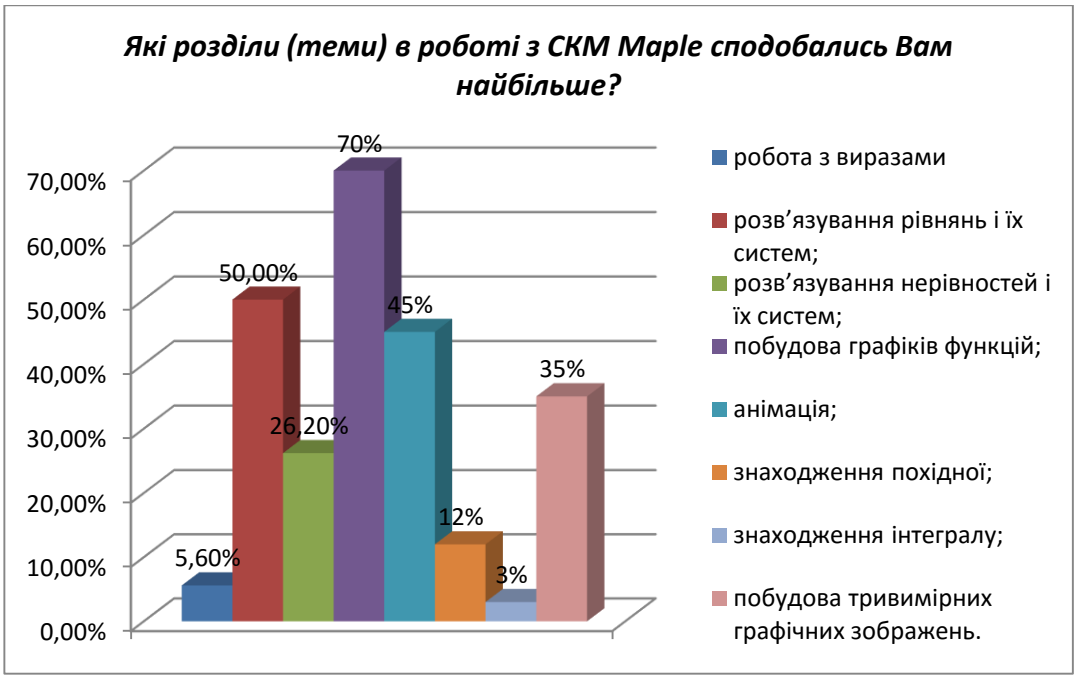
Сучасний освітній процес в ЗНЗ характеризується високою інтенсивністю. Будь-яка дисципліна, пов'язана із використанням інформатичних технологій, передбачає теоретичні, практичні та лабораторні заняття. Вони спрямовані на набуття учнями необхідних знань, формування у них відповідних умінь, зокрема уміння застосовувати теоретичні знання для розв'язування практичних завдань. Базою для цього є знання математичних методів розв'язування та вміння формалізувати поставлену задачу. Під час розв'язування прикладних задач фахівцями всього світу широко застосовуються програмні системи комп'ютерної математики, вагома роль яких у навчальному-виховному процесі в ЗНЗ зростає з кожним днем. Здобувши в школі необхідні знання та навички роботи з певною ефективною системою комп'ютерної математики, майбутній студент без особливих проблем зможе освоїти будь-яку іншу подібну систему. Після завершення навчання у школі учні, які вже мають сформовані вміння та навички практичного застосування математичних пакетів, випереджають інших, в яких немає досвіду роботи з ними, у процесі застосування сучасних ІКТ в професійній діяльності. Саме тому, на нашу думку, є необхідним впровадження СКМ в систему шкільної освіти. Уміння працювати і програмувати в такій системі (у нашому випадку – Maple) можуть знайти своє застосування в різноманітних професійних сферах діяльності: інформатиці, математиці, фізиці, техніці та різноманітних технологіях.

З метою визначення підвищення пізнавального інтересу учнів і межах використання пакету символьної математики Maple було проведено письмове опитування у формі групового анкетування (див. Додаток А). За допомогою методу анкетування можна отримати високий рівень масовості дослідження. Особливістю цього методу є анонімність респондентів. Питання анкети були відкритого і закритого типів. На них відповідали учні 11 класу Лозовецької та

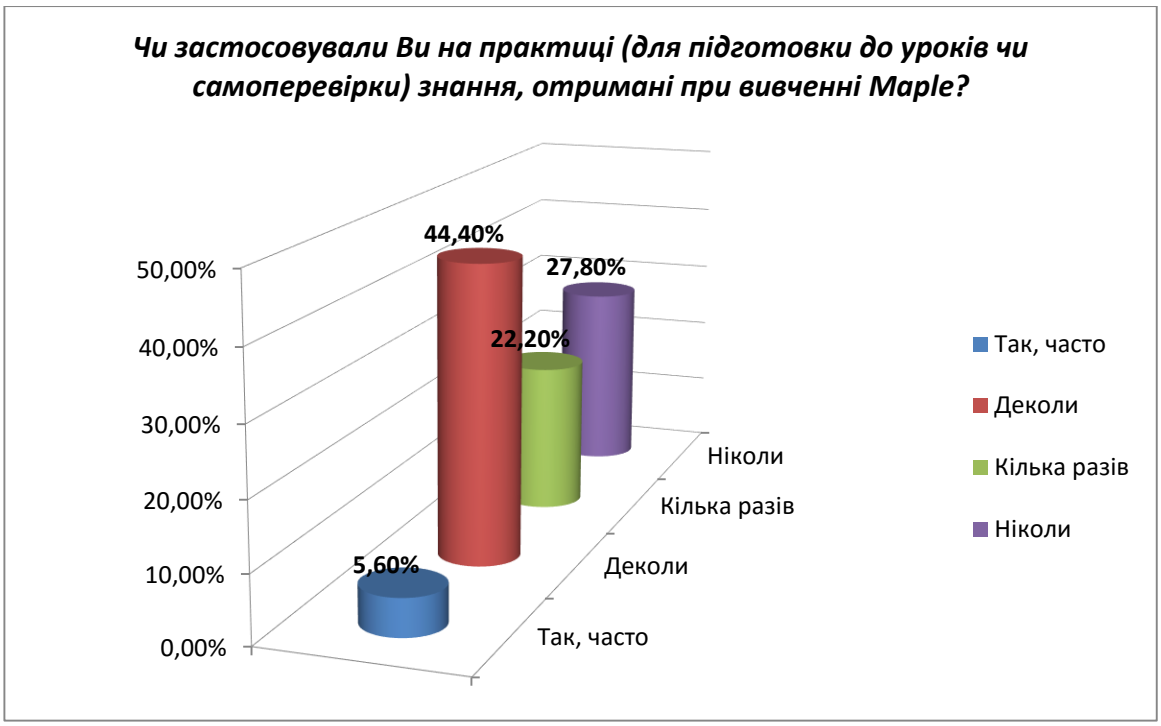
Островської ЗОШ І–ІІІ ступенів. Результати анкетування представлені у формі діаграм.



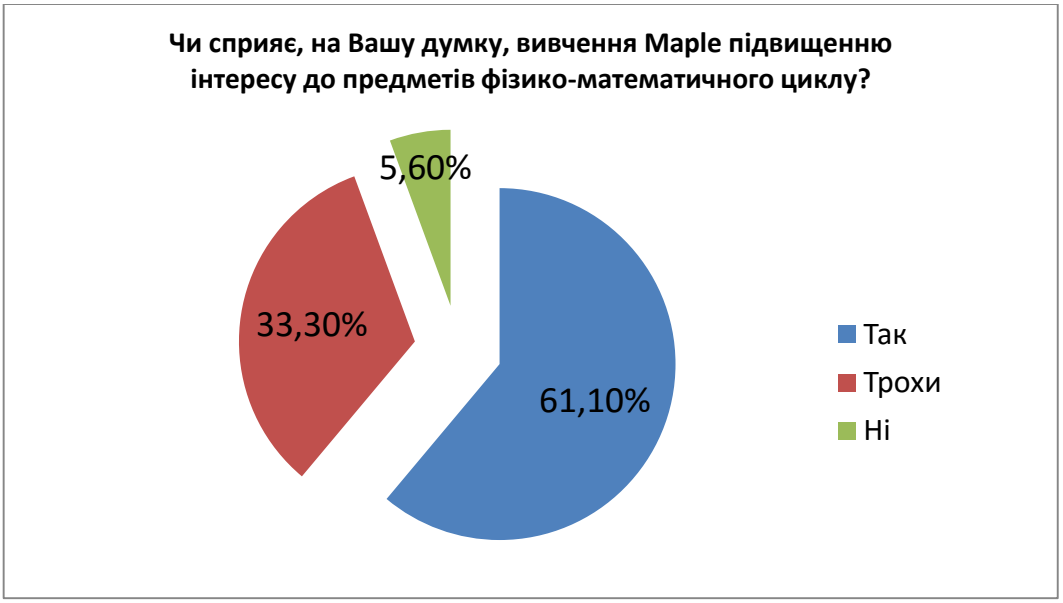
Із наступної гістограми видно, що найбільшою популярністю в пакеті Maple є його графічні функції. Тут реалізується принципи наочності. Візуалізоване і більш наочне повідомлення мотивує пізнавальний інтерес учнів і, отже, сприяє підвищенню якості навчання.



Майже 70 % опитаних учнів застосовує на практиці знання, отримані при вивченні пакета Maple в процесі поглибленого вивчення інформатики. Учні протягом підготовки до уроків математики, фізики, для самоперевірки та просто для цікавості користувалися функціональними можливостями використання Maple як вдома, так і в школі.

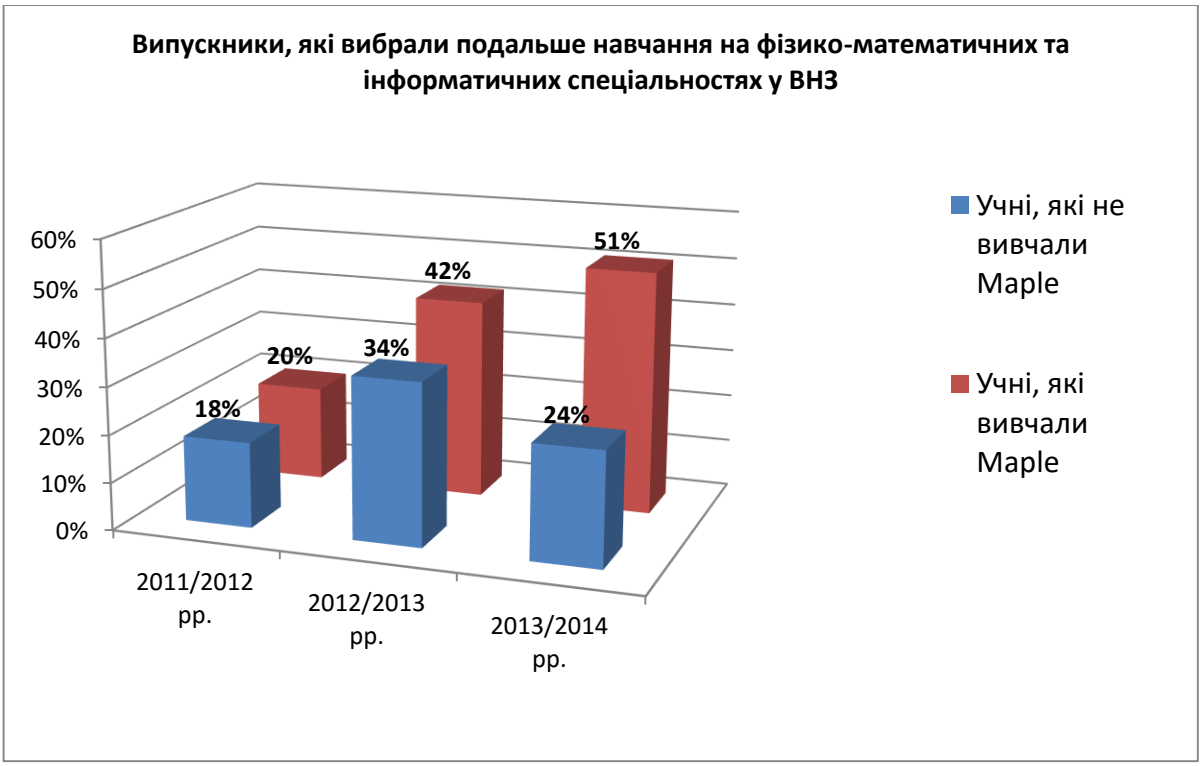


Також старшокласники відзначили, що вивчення СКМ Maple сприяє розвитку логіки і мислення, формує уважність та терпіння при роботі з програмою, дає змогу розширити свої знання в галузі математики та інформатики.



Анкетування показало, що більшість учнів не виключають можливості подальшого вивчення комп'ютерного математичного пакета Maple і застосування отриманих знань у своїй майбутній професії.

Однією із важливих цілей шкільної освіти є розвиток конкурентоспроможної особистості випускника. Період навчання в школі для майбутнього молодого фахівця є надзвичайно важливим як період закладки освітнього фундаменту. На наступній діаграмі зображено кількість старшокласників, які вибрали технічні ВНЗ для отримання своєї професійної діяльності.



Опрацювання результатів проведеної апробації поглибленого курсу інформатики, що містить розділи, які вивчались за допомогою ООП, а саме на прикладі застосування СКМ Maple при поглибленому вивченні алгоритмізації та програмування учнями старших класів ЗОШ, дозволяє стверджувати, що засвоєння основних принципів алгоритмізації та програмування і перехід від вивчення специфічних засобів мови програмування до фундаментальних проблем автоматизації процесів опрацювання даних сприяють підвищенню ефективності навчального процесу, спрямованого на розвиток інформатичних компетентностей і системно-логічного мислення старшокласників, що підтверджує гіпотезу нашого дослідження.

Таким чином, правильність створення та використання системи задач в методиці навчання алгоритмізації та основ програмування старшокласників із застосування СКМ Maple підтверджена високими оцінками. За результатами експерименту усунені окремі неточності, упущення, допрацьована система задач. Результатом педагогічного експерименту є позитивний досвід використання розробленої системи задач як важливого елемента методики навчання алгоритмізації та основ програмування; підтвердження значного впливу застосування НІС при поглибленому вивченні інформатики в старших класах на формування системно-логічного мислення та активізацію навчального інтересу учнів.

Висновки до третього розділу

Результати проведеного педагогічного експерименту дають змогу стверджувати, що методично обґрунтоване цілеспрямоване використання для навчання алгоритмізації та програмування систем комп’ютерної математики сприяє формуванню інформатичних компетентностей, системно-логічного мислення учнів, активізації навчального інтересу, розвитку особистості в умовах сьогодення, загалом підвищенню ефективності навчального процесу.

У третьому розділі підведені підсумки експериментальної роботи, які підтверджують результативність та переваги розробленої нами методики

навчання алгоритмізації та основ програмування із використанням СКМ Maple старшокласників під час поглибленого вивчення інформатики, спрямованого на формування та подальший розвиток системно-логічного мислення школярів.

Після вивчення курсу «Основи алгоритмізації та програмування» проводилася контрольна робота з метою виявлення рівня сформованості компонентів системи інформатичних компетентностей старшокласників та їх готовності до подальшої навчальної діяльності у ВНЗ. До змісту контрольної роботи були включені як тестові завдання [див. Додаток Ж] (для з'ясування рівня теоретичних знань, що лежать в основі інформатичних компетентностей), так і завдання практичного змісту, прикладного характеру [див. Додаток Є] (для визначення чи вміють учні застосовувати здобуті знання у різних ситуаціях).

За показники сформованості компонентів системи інформатичних компетентностей старшокласників у межах вивчення алгоритмізації та програмування були обрані такі:

1) *компетентності у галузі алгоритмізації:*

- розробка математичних алгоритмічних моделей практичних задач;
- дослідження задач на обумовленість (встановлення міри залежності розв'язків задач до змін у вхідних даних);
- добір ефективного програмного методу реалізації математичної моделі;

2) *компетентності у галузі програмування:*

- порівняння алгоритмів чисельного розв'язування задач за часовою складністю;
- розробка алгоритмів розв'язування практичних задач з врахуванням їх універсальності, простоти організації обчислювального процесу, швидкості збіжності, стійкості тощо;

- реалізація алгоритмів розв’язування задач у середовищі програмування СКМ.

Перелік показників сформованості компонентів системи інформатичних компетентностей старшокласників дещо скорочений порівняно із змістом компонентів інформатичних компетентностей, формування яких передбачено у процесі навчання алгоритмізації та програмування. Це зумовлено існуючими обмеженнями щодо можливості прояву у межах процесу навчання алгоритмізації та програмування набутих учнями інформатичних компетентностей. Адже остаточний висновок щодо рівня сформованості інформатичних компетентностей учня можна зробити лише на основі аналізу його студентської, професійної діяльності та процесу розв’язування ситуацій, що виникають у реальному процесі навчання інформатики у ВНЗ.

Результати нашого дослідження також вказують на перспективи використання комп’ютерних математичних пакетів в шкільній освіті.

Матеріали розділу подано в публікаціях автора [277; 272].

ЗАГАЛЬНІ ВИСНОВКИ

У результаті дослідження розроблено і теоретично обґрунтовано методику навчання алгоритмізації та програмування старшокласників на рівні поглибленого вивчення інформатики й отримано такі **результати:**

1. Визначено, що вивчення алгоритмізації та програмування розвиває системно-логічне мислення (стійкі навички і вміння учнів самостійно у новій ситуації визначати завдання, системно розглядати їх, визначати вхідні і вихідні дані, висувати власні гіпотези, обґрунтовувати їх, пропонувати ефективне розв'язання, перевіряти його на практиці та вміти пояснити отриманий результат), що відповідно формує інформатичні компетентності старшокласників.

2. Визначено компоненти інформатичних компетентностей старшокласників, що формуються в процесі навчання алгоритмізації та програмування.

3. Проаналізовано чинні методичні підходи до навчання алгоритмізації та програмування у старшокласників ЗОШ, з'ясовано необхідність їх вдосконалення шляхом переходу до нової об'єктно-орієнтованої парадигми програмування із використанням СКМ та впровадження у навчальний процес в системі базової освіти дітей з поглибленим вивченням інформатики.

4. Розроблено методичну систему навчання алгоритмізації та програмування, засновану на застосуванні СКМ Maple із використанням навчальних середовищ «ІнфоНІС» і MOODLE: конкретизовано цілі навчання і розроблено зміст вивчення алгоритмізації та програмування, підґрунтям яких є використання комп'ютерних математичних пакетів як засобів навчання у класах із поглибленим вивченням інформатики; розроблено систему навчальних задач із загальної алгоритмізації та програмування, практичні роботи із зазначеної теми, конспекти уроків, навчальні програми допоміжних курсів та їх тематичне планування, збірку тестових завдань для перевірки знань учнів.

5. Експериментальним шляхом перевірено доцільність та ефективність запропонованої методичної системи. За підсумками педагогічного експерименту усунено окремі неточності, упущення, допрацьовано практичні роботи. Результатом педагогічного експерименту є позитивний досвід використання розробленої системи задач у практичних роботах, як важливого елемента методичної системи навчання алгоритмізації та програмування; підтверджено значний ефективний вплив застосування інформаційно-навчальних середовищ під час поглибленого вивчення інформатики в старших класах на розвиток системно-логічного мислення та формування інформатичних компетентностей учнів.

Отримані результати дають можливість зробити такі висновки:

1. Вивчення алгоритмізації та програмування в процесі навчання інформатики з використанням запропонованої методичної системи на основі ООП є інструментом формування інформатичних компетентностей школярів, зокрема, сприяє розвитку системно-логічного мислення учнів старших класів.

2. Доцільно використовувати СКМ, як засіб навчання учнів старших класів алгоритмізації та програмуванню на рівні поглибленого вивчення інформатики. Ці системи утворюють зручне інтелектуальне середовище для досліджень та реалізації міжпредметних зв’язків, зокрема інформатики та математики.

3. Впровадження розробленої методичної системи сприяє комплексній реалізації функцій навчання у класах із поглибленим вивченням інформатики – формуванню інформатичних компетентностей на основі здобутих знань та навичок із алгоритмізації та програмування, вміння застосовувати їх у повсякденному житті, подальшій фаховій діяльності, розвитку в учнів мислення системно-логічного типу.

4. Використання засобів ІКТ у процесі навчання алгоритмізації та програмування сприяє формуванню основних компонентів системи інформатичних компетентностей старшокласників, оволодінню цими

засобами як інструментами пізнання, конструюванню власної системи знань, системи предметних та загально професійних компетентностей.

Одержані результати дослідження дають змогу окреслити напрями подальших досліджень:

- дослідження методичних систем навчання інших інформатичних дисциплін із погляду ефективності їх застосування для формування предметних та загальних професійних компетентностей;
- розробка ефективних методів оцінювання та моніторингу рівня сформованості інформатичних компетентностей студентів на різних етапах навчання у педагогічному університеті.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Абдеев Р. Ф. Философия информационной цивилизации / Р. Ф. Абдєєв. – М. : ВЛАДОС, 1994. – 336 с.
2. Аверьянов А. Н. Системное познание мира : методологические проблемы / А. Н. Аверьянов. – М., 1985. – 263 с.
3. Аладьев В. З. Основы информатики : учебное пособие / В. З. Аладьев, Ю. Я. Хунт, М. Л. Шишаков. – М. : Информационно-издательскийдом «Филинъ», 1998. – 496 с.
4. Аладьев В. З. Основы информатики : учебное пособие / В. З. Аладьев, Ю. Я. Хунт, М. Л. Шишаков. – М. : Информационно-издательский дом «Филинъ», 1998. – 496 с.
5. Андросова Е. Г. Некоторые вопросы соержания курса программирования на основе объектно-ориентированного похода (ООП) / Елена Григорьевна Андросова, Галина Юрьевна Хачева // ИТО-2000. – Брянск : Брянский государственный педагогический университет (БГПУ), 2000. – С. 5 – 6.
6. Андросова Е. Г. Методические и содержательные аспекты построения курса программирования на основе объектно-ориентированного подхода (для физико-математических специальностей педагогических вузов) : дисс. … канд. пед. наук. / Е. Г. Андросова. – М., 1996. – 325 с.
7. Апатова Н. В. Інформатика : підр. для учн. 10-11 кл. серед. загальноосв. шк / Н. В. Апатова, А. Ф. Верлань. – К. : Форум, 2000. – 200 с.
8. Апатова Н. В. Інформатика для економістів : підручник / Н. В. Апатова, О. М. Гончарова, Ю. Ю. Дюлічева. – К. : Центр учбової літератури, 2011. – 456 с.
9. Апатова Н. В. Основы информатики. 8-9 кл. : учеб. для общеобразоват. учеб. заведений. 2-ое изд. / Н. В. Апатова, А. А. Кузнецов. – М. : Дрофа, 2000. – 176 с.
10. Архіпова Т. Л. Активізація навчально-пізнавальної діяльності учнів 7-9 класів у процесі вивчення геометрії з використанням комп’ютера : дис. … канд. пед. наук : 13.00.02 / Т. Л. Архіпова ; Національний педагогічний університет імені М. П. Драгоманова. – К., 2002. – 236 с.
11. Асмолов А. Г. Как проектировать универсальные учебные действия в начальной школе : от действия к мысли : пособие для учителя /

[А. Г. Асмолов [и др.] ; под ред. А. Г. Асмолова. – М. : Просвещение, 2008. – 152 с.

12. Бабанский Ю. К. Избранные педагогические труды / [сост. М. Ю. Бабанский ; авт. вступ. ст. Г. Н. Филонов, Г. А. Победоносцев, А. М. Моисеев ; авт. коммент. А. М. Моисеев] ; Акад. пед. наук СССР. – М. : Педагогика, 1989. – 558 с.
13. Бадд Т. ООП в действии / Т. Бадд. – СПб. : Питер, 1997. – 464 с.
14. Баженов Р. И. Использование технологии объектно-ориентированного подхода для развития мыслительных действий учащихся при изучении базового курса информатики : автореф. дисс. … канд. пед. наук / Р. И. Баженов. – Омск, 1998. – 36 с.
15. Баженов Р. И. Применение объектно-ориентированного подхода для развития у школьников интеллектуальных учений : дисс. … канд. пед. наук : 13.00.02 / Р. И. Баженов. – Биробиджан : Изд-во БГПИ, 1998. – 133 с.
16. Балик Н. Р. Активне навчання з використанням технологій ВЕБ 2.0 / Н. Р. Балик, О. О. Лялик. – Тернопіль : Навчальна книга – Богдан, 2009. – 88 с.
17. Балик Н. Р. Методичні рекомендації для проведення практики з Web-програмування / Н. Р. Балик, В. П. Олексюк, В. І. Мандзюк. – Тернопіль : ТНПУ, 2005. – 56 с.
18. Баловсяк Н. Інформаційна компетентність фахівця / Н. Баловсяк // Педагогіка і психологія професійної освіти. – 2004. – № 5. – С. 21 – 28.
19. Биков В. Ю. Упровадження інформаційно-комунікаційних технологій в освіті – імператив її модернізації / В. Ю. Биков // Національна доповідь розвитку освіти України, 2011. – С. 118 – 124.
20. Бичков О. С. Опрацювання статистичної інформації у середовищі MathCad [Електронний ресурс] / О. С. Бичков, С. І. Доценко. – Режим доступу : http://www.unicyb.kiev.ua/ Library/OKZ2/index.htm.
21. Білоусова Л. І. Лабораторний практикум з чисельних методів на базі пакету MathCAD: навч. посібник / Т. В. Бєлявцева, О. Г. Колгатін, Л. С. Пономарьова. – К., 1998. – 164 с.

22. Богоявленский Д. Н. Формирование приемов умственной работы учащихся как путь развития мышления и активизации учения / Д. Н. Богоявленский // Вопросы психологии. – 1962. – № 4. – С. 74 – 82.
23. Бугаєць Н. О. Використання програм математичного призначення для знаходження ектремумів функцій / Н. О. Бугаєць // Науковий часопис Національного педагогічного університету імені М. П. Драгоманова. Серія 2. Комп'ютерно-орієнтовані системи навчання. – 2011. – Вип. 10 (17). – С. 96 – 103.
24. Бугаєць Н. О. Програмування візуально-орієнтованого інтерфейсу в програмі Maple / Н. О. Бугаєць // Науковий часопис Національного педагогічного університету імені М. П. Драгоманова. Серія 2. Комп'ютерно-орієнтовані системи навчання. – 2012. – Вип. 12 (19). – С. 67 – 71.
25. Буч Г. Объектно-ориентированное проектирование с примерами приложений на C++ / Г. Буч. – СПб. : Бином, 1998. – 558 с.
26. Буч Г. Объектно-ориентированный анализ с примерами приложений на C++ / Г. Буч. – М. : Бином, СПб. : Невский диалект. – 1998. – 560 с.
27. Ваграменко Я. А. Електронно-обчислювальна техніка : проб. навч. посібник для 8-9 кл. серед. школи / Я. А. Ваграменко, І. М. Антипов, Е. І. Кузнецов та ін. ; за ред. Я. А. Ваграменка. – К. : Рад. шк., 1991. – 144с.
28. Варламов С. А. Методические приемы в преподавании информатики / С. А. Варламов // Информатика и образование. – 2000. – № 23. – С. 11 – 18.
29. Варпаховский Ф. Л. Элементы теории алгоритмов : учебное пособие / Ф. Л. Варпаховский. – М. : Просвещение, 1970. – 25 с.
30. Васильєва Л. В. Чисельні методи розв'язання інженерних задач в пакеті MathCAD. Курс лекцій та індивідуальні завдання : навч. посібник з дисципліни «Інформатика» для студентів вищих навчальних закладів / Л. В. Васильєва, О. А. Гончаров, В. А. Коновалов, Н. А. Соловйова. – Краматорськ : ДДМА, 2006. – 108 с.
31. Веселовская Н. С. Компетентностный подход в образовании — основа подготовки высококвалифицированного специалиста / Н. С. Веселовская. – Омск, 2004.

32. Воробієнко П. П. Інформатизація загальноосвітніх закладів України [Електронний ресурс] / П. П. Воробієнко, А. М. Гуржій, В. А. Коляденко // Комп'ютер у школі та сім'ї. – 2014. – № 1. – С. 3 – 5. – Режим доступу : http://nbuv.gov.ua/UJRN/komp_2014_1_2.
33. Гейн А. Г. Информатика : учебное пособие для 8-9 кл. / А.Г. Гейн и др. – М. : Просвещение, 1997. – 254 с.
34. Гейн А. Г. Информатика / А. Г. Гейн, Е. В. Линецкий, М. В. Сапир, В. Ф. Шолохович. – М. : Просвещение, 1994. – 254 с.
35. Гелемб'юк Р. В. Об'єктно-орієнтоване програмування в СКА Maple [Електронний ресурс] / Р. В. Гелемб'юк // Математичні машини і системи. – 2009. – № 1. – С. 102 – 109. – Режим доступу : http://www.immsp.kiev.ua/publications/ articles/2009/2009_1/Gelembyk_01_2009.pdf.
36. Гершунский Б. С. Компьютеризация в сфере образования : проблемы и перспективы / Б. С. Гершунский. – М. : Педагогика, 1987. – 264 с.
37. Гетманова А. Д. Логика : для пед. учеб. заведений / А. Д. Гетманова. – М. : Новая школа, 1995. – 415 с.
38. Гетманова А. Д. Логика : учебник для студентов вузов / А. Д. Гетманова. – М. : Омега-Л, 2007. – 416 с.
39. Горошко Ю. В. Про часову складність алгоритмів [Електронний ресурс] / Ю. В. Горошко // Науковий часопис НПУ імені М. П. Драгоманова. Серія 2. Комп'ютерно-орієнтовані системи навчання. – 2015. – Вип. 15. – С. 27 – 31. – Режим доступу : http://nbuv.gov.ua/UJRN/Nchnpu_2_2015_ 15_7.
40. Горошко Ю. В. Система інформаційного моделювання у підготовці майбутніх учителів математики та інформатики: дис. … док. пед. наук: 13.00.02 / Ю. В. Горошко. – К., 2013. – 470 с.
41. Горошко Ю. В. Застосування вільно поширюваного програмного забезпечення до розв'язування задач лінійного програмування / Ю. В. Горошко, Г. Ю. Цибко // Комп'ютер у школі та сім'ї. – 2015. – № 3. – С. 11 – 14.

42. Горячев А. В. Программа курса информатики для 1-9 класса средней школы / А. В. Горячев, А. С. Лесневский // Информатика и образование. – 1997. – № 7. – С. 12 – 17.
43. Грамаков Д. А. Методология использования ИКТ в системе дистанционного обучения сельских школьников // Дистанционное обучение сельских школьников : метод. пособие / Под редакцией Д. А. Грамакова. – М. : Изд-во МГОУ, 2003 г. – С. 17 – 63.
44. Гриценко В. І. Дистанційне навчання : теорія та практика / В. І. Гриценко, С. П. Кудрявцева, В. В. Колос, О. В. Веренич. – К. : Наукова думка, 2004. – 376 с.
45. Гришко Л. В. Концептуальні підходи до навчання основ програмування у вищій школі / Л. В. Гришко // Науковий часопис НПУ імені М. П. Драгоманова. Серія 2. Комп'ютерно-орієнтовані системи навчання. – 2004. – Вип. 1(8). – С. 134 – 148.
46. Грудецький Р. Я. Взаємодія delphi-додатків та програмного середовища Matlab / Р. Я. Грудецький // Науковий журнал «Комп'ютерно-інтегровані технології : освіта, наука, виробництво. – Луцьк, 2013. – Вип. 12. – С. 18 – 22.
47. Гуржій А. М. Основи програмування : навчальний посібник / А. М. Гуржій, М. С. Львов, О. В. Співаковський. – Кривий Ріг : Наукова думка, 2004. – 355 с.
48. Давыдов В. В. Виды обобщения в обучении / В. В. Давыдов. – М. : Педагогика, 1971. – 423 с.
49. Данилов М. А. Дидактика / М. А. Данилов, Б. П. Есипов; под общ. ред. Б. П. Есипова; – М. : Изд-во АПН РСФСР, 1957. – 518 с.
50. Дем'яненко В. Б. Розвиток особистості учня шляхом використання інформаційно-комунікаційних технологій / В. М. Дем'яненко, В. Б. Дем'яненко // Матеріали Всеукраїнської конференції «Освіта обдарованої та талановитої молоді – національна проблема». Київ, 1 грудня 2011 р., Ч.1. – К. : Інститут обдарованої дитини НАПН України, 2011. – С. 137 – 143.
51. Дем'яненко В. Б. Забезпечення засад неперервної освіти шляхом використання інформаційно-комунікаційних технологій /

В. Б. Дем’яненко, В. М. Дем’яненко // Актуальні проблеми методології та методики навчання фізико-математичних дисциплін : матеріали Міжнародної наукової конференції, 18-19 січня 2013 р. – К. : Вид-во НПУ імені М. П. Драгоманова, 2013. – С. 91 – 94.

52. Дем’яненко В. Б. Основні вимоги до організації науково-дослідницької роботи учнів Малої академії наук України. Відділення комп’ютерних наук : методичні рекомендації / В. Б. Дем’яненко. – К. : КПНЗ «Київська Мала академія наук учнівської молоді», 2012. – 56 с.
53. Дем’яненко В. Б. Формування інформаційно-комунікаційних компетентностей учнівської молоді шляхом активізації пізнавальної діяльності в позашкільній освіті / В. Б. Дем’яненко // «Інноваційні технології навчання обдарованої молоді». Міждисциплінарна науково-практична конференція 08–09 грудня 2010 р., м. Київ. – К. : ТОВ «Інфосистем», 2010. – С. 40 – 45.
54. Державний стандарт базової і повної загальної середньої освіти (2013-2018 рр.) [Електронний ресурс]. – Режим доступу : http://www.mon.gov.ua/ua/often-requested/state-standards/.
55. Довгий Б. П. Використання математичного пакета MATLAB для розв’язування прикладних задач / Б. П. Довгий, Є. С. Вакал, Ю. Є. Вакал, А. В. Попов. – К. : Фітосоціоцентр, 2012. – 77 с.
56. Дрозденко О. Л. Використання пакету символьних обчислень Maple при розв’язування деяких задач аналітичної геометрії // Науковий часопис Національного педагогічного університету імені М. П. Драгоманова. Серія 2. Комп'ютерно-орієнтовані системи навчання. – 2007. – Вип. 5 (12). – С. 55 – 59.
57. Дубова Т. В. Розвиток пізнавальної активності учнів 5–6 класів на основі нових інформаційних технологій навчання на уроках математики : автореф. дис. на здобуття наук. ступеня канд. педагог. наук : спец. 13.00.02 «Теорія та методика навчання (математика)» / Т. В. Дубова. – К., 2002. – 18 с.
58. Дьяконов В. П. Справочник по применению системы PC MatLab. – М. : Наука, 1993. – 109 с.

59. Дьяконов В. П. Энциклопедия компьютерной алгебры. – М. : ДМК Пресс, 2012. – 1263 с.
60. Дьяконов В. П. Система MathCAD : справочник / В. П. Дьяконов. – М. : Радио и связь, 1993. – 127 с.
61. Дьяконов В. П. Компьютерная математика. Теория и практика / В. П. Дьяконов. – М. : Нолидж, 2001. – 1296 с.
62. Єршов А. П. Введение в теоретическое программирование : Беседы о методе/ А. П. Єршов. – М. : Наука, Главная редакция физико-математической литературы, 1977. – 288 с.
63. Єршов А. П. Основы информатики и вычислительной техники : пробное учебное пособие для средних учебных заведений. В 2-х частях. Часть первая / А. П. Єршов, В. М. Монахов, С. А. Бешенков ; под редакцией А. П. Єршова и В. М. Монахова. – М. : Просвещение, 1985. – 96 с.
64. Жалдак М. І. Інформатика – 7 : експериментальний навчальний посібник для учнів 7 класу загальноосвітньої школи / М. І. Жалдак, Н. В. Морзе. – К. : ДіаСофт, 2000. – 207 с.
65. Жалдак М. І. Комп'ютер на уроках математики : посібник для вчителів / М. І. Жалдак. – К. : РННЦ «Дініт», 2003. – 324 с.
66. Жалдак М. І. Основи інформаційної культури вчителя / М. І. Жалдак // Використання інформаційних технологій в навчальному процесі : зб. наук. праць. – К. : КДПІ, 1990. – С. 3 – 24.
67. Жалдак М. І. Математика з комп'ютером : посібник для вчителів / М. І. Жалдак, Ю. В. Горошко, Є. Ф. Вінниченко. – К. : РННЦ «ДІНІТ». – 2004. – 255 с.
68. Жалдак М. І. Математика з комп'ютером : посібник для вчителів. – К. : НПУ ім. М. П. Драгоманова, 2009. – 282 с.
69. Жалдак М. І. Модель системи соціально-професійних компетентностей вчителя інформатики / М. І. Жалдак // Науковий часопис НПУ імені М. П Драгоманова. Серія 2. Комп'ютерно-орієнтовані системи навчання. – 2009. – Вип. 7 (14). – С. 4.
70. Жалдак М. Інформатика : навчальний посібник / М. І. Жалдак, Ю. С. Рамський ; за ред. М. І. Шкіля. – К. : Вища школа, 1991. – 319 с.

71. Жалдак М. І. Вивчення мов програмування в школі / М. І. Жалдак, М. І. Шкіль , Н. В. Морзе , Ю. С. Рамський. – К. : Радянська школа, 1988. – 368 с.
72. Жалдак М. І. Комп'ютерно-орієнтовані засоби навчання математики, фізики, інформатики : посібник для вчителів / М. І. Жалдак, В. В. Лапінський, М. І. Шут // Інформатика. – 2006 – № 3–4. – 96 с.
73. Жалдак М. І. Програма шкільного курсу «Інформатика» для базової школи (7-9 класи) / М. І. Жалдак, Н. В. Морзе, Г. Г. Науменко // Інформатика. – 2003. – 26 с.
74. Жалдак М. І. Система підготовки вчителя до використання інформаційно-комунікаційних технологій в навчальному процесі / М. І. Жалдак // Науковий часопис НПУ імені М. П. Драгоманова. Серія 2. Комп'ютерно-орієнтовані системи навчання. – 2011. – Вип. 11. – С. 3 – 15.
75. Жалдак М. І. Формування системи інформатичних компетентностей майбутніх учителів інформатики у процесі навчання в педагогічному університеті / М. І. Жалдак, Ю. С. Рамський, М. В. Рафальська // Вища школа. – Освітні технології – 2009. – № 10. – С. 44 – 52.
76. Жилин С. А. Малые компонентные среды и языки программирования в школьном информационном образовании / С. А. Жилин, И. Б. Жилина // Информатика и образование. – 2006. – № 7. – С. 44 – 50.
77. Жученко А. І. Динамічна оптимізація з використанням MATLAB та SIMULINK / А. І. Жученко, Л. Р. Ладієва, Р. М. Дубік. – К. : НТУУ «КПІ», 2010. – 209 с.
78. Заболотный В. П. Философские проблемы информатизации / В. П. Заболотный // Проблемы информатизации. – 1999. – № 1. – С. 129.
79. Завадський І. О. Практикум і робочий зошит з інформатики. 10 клас. Академічний рівень / І. О. Завадський, О. В. Пасічник, В. В. Бойчук. – К. : Вид. група BHV, 2010. – 190 с.
80. Загвязинский В. И. Теория обучения : современная интерпретация : учеб. пособие для студ. высш. пед. учеб. заведений / В. И. Загвязинский. – М. : Издательский центр «Академия», 2001. – 192 с.

81. Зайнутдинова Л. Х. Создание и применение электронных учебников / Л. Х. Зайнутдинова. – Астрахань, ООО «ЦНТЭП», 2003. – 364 с.
82. Закон України «Про освіту» № 1060-12 ; редакція від 19.02.2016 / [Електронний ресурс]. – Режим доступу : http://zakon0.rada.gov.ua/laws/show/1060-12.
83. Зеер Э. Ф. Личностно ориентированные технологии профессионального развития специалиста : науч-метод. пособие / Э. Ф. Зеер, О. Н. Шахматова. – Екатеринбург : Изд-во Урал. гос. проф.-пед. ун-та, 1999. – 245 с.
84. Зеленська Т. С. Використання пакету Mathcad при розв'язуванні диференціальних рівнянь в частинних похідних [Електронний ресурс] / Т. С. Зеленська // Науковий часопис НПУ імені М. П. Драгоманова. Серія 2. Комп'ютерно-орієнтовані системи навчання. – 2011. – Вип. 11. – С. 86 – 91. – Режим доступу : http://nbuv.gov.ua/j-pdf/Nchnpu_2_2011_11_16.pdf.
85. Зимняя И. А. Интегральный подход к оценке единой социально-профессиональной компетентности выпускников вузов / И. А. Зимняя, Е. В. Земцова // Высшее образование сегодня. – 2008. – № 5. – С. 14 – 19.
86. Зимняя И. А. Ключевые компетенции – новая парадигма результата образования / И. А. Зимняя // Высшее образование сегодня. – 2003. – № 5. – С. 34 – 42.
87. Иванова Д. С. Совершенствование информационной подготовки будущих учителей физики (на примере курса информатики «Основы объектно-ориентированного программирования педагогических приложений по физике») : дисс. … . канд. пед. наук / Д. С. Иванова. – М., 2004. – 230 с.
88. Іваськів І. С. Програмний комплекс «Денвер» : можливості використання у процесі вивчення основ Web-програмування / І. С. Іваськів, Ю. С. Рамський, В. П. Олексюк // Науковий часопис НПУ імені М. П. Драгоманова. Серія 2. Комп'ютерно-орієнтовані системи навчання. – 2006. – Вип. 4 (11). – С. 66 – 69.

89. Інформатика : підруч. для 10 кл. загальноосвітн. навч. закл. : рівень стандарт / Н. В. Морзе, В. П. Вембер, О. Г. Кузьмінська, Н. А. Саражинська. – К. : Школяр, 2010. – 304 с.
90. Інформатика : підруч. для 11 кл. загальноосвіт. навч. закл. : рівень стандарту / Н. В. Морзе, О. В. Барна, В. П. Вембер, О. Г. Кузьмінська. – К. : Школяр, 2011. – 304 с.
91. Інформатика : підруч. для 4 класу загальноосвіт. навч. закл. / М. М. Корнієнко, С. М. Крамаровська, І. Т. Зарецька. – Х. : Вид-во «Ранок», 2015. – 160 с.
92. Інформатика : підручник для 5 класу загальноосвітніх навчальних закладів / Н. В. Морзе, О. В. Барна, В. П. Вембер, О. Г. Кузьмінська, Н. А. Саражинська. – К. : Видавничий дім «Освіта», 2013. – 256 с.
93. Інформатика : підручник для 6 класу загальноосвітніх навчальних закладів / Н. В. Морзе, О. В. Барна, В. П. Вембер, О. Г. Кузьмінська, Н. А. Саражинська. – К. : Видавничий дім «Освіта», 2014. – 240 с.
94. Інформатика : підручник для 9 кл. загальноосвітніх навчальних закладів / В. В. Володін, І. Л. Володіна. – Х. : Видавництво «Гімназія», 2009. – 384 с.
95. Інформатика : 9 кл. : підруч. для загальноосвіт. навч. закл. / І. О. Завадський, І. В. Стеценко, О. М. Левченко. – К. : Видавнича група BHV, 2009. – 320 с.
96. Інформатика. 10 клас / Й. Я. Ривкінд, Т. І. Лисенко, Л. А. Чернікова, В. В. Шакотько ; за заг. ред. М. З. Згуровського. – К. : Генеза, 2010. – 296 с.
97. Інформатика. 10 клас : академічний рівень, профільний рівень / Й. Я. Ривкінд, Т. І. Лисенко, Л. А. Чернікова, В. В. Шакотько ; за заг. ред. М. З. Згуровського. – К. : Генеза, 2010. – 304 с.
98. Інформатика. 11 клас : академічний рівень, профільний рівень / Й. Я. Ривкінд, Т. І. Лисенко, Л. А. Чернікова, В. В. Шакотько ; за заг. ред. М. З. Згуровського. – К. : Генеза, 2011. – 304 с.
99. Інформатика. 11 клас : рівень стандарту / Й. Я. Ривкінд, Т. І. Лисенко, Л. А. Чернікова, В. В. Шакотько ; за заг. ред. М. З. Згуровського. – К. : Генеза, 2011. – 304 с.

100. Інформатика. 5 клас / Й. Я. Ривкінд, Т. І. Лисенко, Л. А. Чернікова, В. В. Шакотько. – К. : Генеза, 2013. – 200 с.
101. Інформатика. 6 клас / Й. Я. Ривкінд, Т. І. Лисенко, Л. А. Чернікова, В. В. Шакотько. – К. : Генеза, 2014. – 256 с.
102. Інформатика. 9 клас / Й. Я. Ривкінд, Т. І. Лисенко, Л. А. Чернікова, В. В. Шакотько ; за заг. ред. М. З. Згуровського. – К. : Генеза, 2009. – 296 с.
103. Каймин В. А. Информатика : учебник / В. А. Каймин. – М. : ИНФРА-М, 2000. – 232 с.
104. Керниган Б. У. Язык программирования С (Си). 2-е изд. / Брайан У. Керниган, Деннис М. Ритчи. – «ВИЛЬЯМС», 2013. – 304 с.
105. Кинг Д. Создание эффективного программного обеспечения / Д. Кинг. – М. : Мир, 1991. – 288 с.
106. Кирсанов М. Н. Графы в Maple. Задачи, алгоритмы, программы / М. Н. Кирсанов. – М. : Физматлит, 2007. – 168 с.
107. Классификация по соответствию методов обучения логике познания и по дидактической цели [Электронный ресурс]. – Режим доступа : http://islameducation.net/methodology/classification.html.
108. Кловак Г. Т. Теорія і практика педагогічного експерименту. Основи педагогічних досліджень : навч. посіб. / Г. Т. Кловак. – Чернігів : Черніг. держ. центр наук.-техн. і екон. інформації, 2003. – 260 с.
109. Клочко В. І. Система задач як засіб формування професійно значущих знань з інформатики студентів економічних спеціальностей : монографія / В. І. Клочко, Н. І. Праворська. – Вінниця : УНІВЕРСУМ – Вінниця, 2008. – 140 с.
110. Клочко В. І. Система задач як засіб формування професійно значущих знань з інформатики студентів економічних спеціальностей : монографія / В. І. Клочко, Н. І. Праворська ; Вінниц. нац. техн. ун-т. – Вінниця : Універсум – Вінниця, 2008. – 139 с.
111. Клочко В. І. НІТ навчання математики в технічній вищій школі : дис. ... доктора пед. наук : 13.00.02 / В. І. Клочко. – Вінниця, 1998. – 396 с.
112. Клочко В. І. Розвиток пізнавальної самостійності студентів засобами інформа-цій-них технологій / В. І. Клочко // Досвід та проблеми країн

Європи (Вели-ко-британії, Німеччини, Франції, Іспанії, України) з реалізації ідей Болонської концепції. Матеріали Міжнар. наук.-практ. конф. Ч. 2. – Біла Церква, 2007. – С. 87 – 91.

113. Клочко В. І. Середовище Delphi як засіб активізації пізнавальної діяльності студентів з метою оволодіння курсом інформатики та комп'ютерної техніки / В. І. Клочко, Я. І. Плаксій // Науковий часопис НПУ імені М. П. Драгоманова. Серія 2. Комп'ютерно-орієнтовані системи навчання. – 2008. – Вип. 6 (13). – С. 86 – 93.
114. Компетентнісний підхід у сучасній освіті : світовий досвід та українські перспективи : Бібліотека з освітньої політики / Під заг. ред. О. В. Овчарук. – К. : К.І.С., 2004. – 112 с.
115. Компетентностный подход в педагогическом образовании : коллективная монография / Под ред. проф. В. А. Козырева и проф. И. Ф. Радионовой. – СПб. : Изд-во РГПУ им. А. И. Герцена, 2004. – 392 с.
116. Короткий словник системи психологічних понять / Упоряд. К. К. Платонов. – М. : Вища школа, 1981. – 175 с.
117. Краткая философская энциклопедия. – М. : Изд. группа «Прогресс» – «Энциклопедия», 1994. – 586 с.
118. Кривонос О. М. Використання задачного підходу в процесі навчання програмування майбутніх учителів інформатики / О. М. Кривонос // Інформаційні технології і засоби навчання, 2014. – Т. 40. – № 2. – С. 83 – 91.
119. Кримський С. Запити філософських смислів / С. Б. Кримський. – К. : ПАРАПАН, 2003. – 240 с.
120. Критерії оцінювання навчальних досягнень учнів у системі загальної середньої освіти / В. О. Огнев'юк (за ред) ; Академія педагогічних наук України. – К. : Ірпінь і Перун, 2004. – 176 с.
121. Кузнецов А. А. Основные направления совершенствования методической подготовки учи- телей информатики в педагогических вузах / А. А. Кузнецов, С. В. Кариев // Ин- форматика и образование. –1997. – № 5. – С. 13 – 20.
122. Кузнецов А. А. Основы информатики. 8-9 кл. : учебн. для общеобраз. учебных заведений / А. А. Кузнецов, Н. В. Апатова. – М. : Дрофа, 1999. – 176 с.

123. Кузнецов А. Б. Методика обучения учащихся классов с углубленным изучением информатики объектно-ориентированному проектированию программ : дисс. ... канд. пед. наук : 13.00.02 / А. Б. Кузнецов. – Екатеринбург, 1999. – 268 с.
124. Кузьміна Н. М. Компетентнісний підхід до навчання інформаційних систем і технологій майбутніх учителів економіки / Н. М. Кузьміна, О. В. Струтинська // Інформаційні технології в освіті. – 2011. – № 9. – С. 57 – 63.
125. Кузьміна Н. М. Деякі методичні аспекти навчання основ теорії і методів оптимізації з комп'ютерною підтримкою / Н. М. Кузьміна // Науковий часопис НПУ імені М. П. Драгоманова. Серія 2. Комп'ютерно-орієнтовані системи навчання. – 2015. – Вип. 15. – С. 42 – 49. – Режим доступу : http://nbuv.gov.ua/UJRN/Nchnpu_2_2015_15_9.
126. Кустовська О. В. Методологія системного підходу та наукових досліджень : курс лекцій / О.В. Кустовська. – Тернопіль : Економічна думка, 2005. – 124 с.
127. Кухаренко В. М. Дистанційне навчання : умови застосування. Дистанційний курс : навчальний посібник. – 2-е вид, доп. / В. М. Кухаренко, О. В. Рибалко, Н. Г. Сиротинко ; за ред. В. М. Кухаренка. – Харків : НТУ «ХПІ», «Торсінг», 2001. – 320 с.
128. Кушниренко А. Г., Лебедев Г. В. 12 лекций о том, для чего нужен школьный курс информатики и как его преподавать. — М.: Лаборатория Базовых Знаний, 2000. — 464 с.
129. Кушниренко А. Г., Лебедев Г. В. Программирование для математиков: Учебное пособие для вузов по специальностям «Математика» и «Прикладная математика». — М.: Наука, 1988. — 384 с.
130. Лабораторний практикум з інформатики : навч. посіб. / Н. А. Дроговоз та ін. ; [за заг. ред. О. В. Рєзіної] // Кіровогр. держ. пед. ун-т ім. Володимира Винниченка. – Кіровоград : РВВ КДПУ ім. В. Винниченка, 2013. – 80 с.
131. Лапчик М. П. Введение в программирование / М. П. Лапчик. – М. : Просвещение, 1979. – 144 с.

132. Лапчик М. П. Методика преподавания информатики : учеб. пособие для студ. пед. вузов / М. П. Лапчик, И. Г. Семакин, Е. К. Хеннер ; под общей ред. М. П. Лапчика. – М. : Издательский центр «Академия», 2006. – 624 с.
133. Леонова Л. М. Преподавание объектно ориентированного программирования в системе лицей-вуз / Н. М. Леонова, Н. В. Коробков // ИТО-97. – М. : МИФИ, 1997. – С. 51 – 52.
134. Леонтьєв О. О. Педагогическое общение / О. О. Леонтьєв. – М. : Знание, 1979. – 47 с.
135. Лесневский А. С. Объектно-ориентированное программирование для начинающих / А. С. Лесневский. – М. : Бином. Лаборатория знаний, 2005. – 232 с.
136. Лещук С. О. Навчально-інформаційне середовище як засіб активізації пізнавальної діяльності учнів старшої школи у процесі навчання інформатики : дис. .. канд. пед. наук : 13.00.02 / С. О. Лещук // Національний педагогічний ун-т ім. М. П. Драгоманова. – К., 2006. – 225 с.
137. Лещук С. О. Приклад реалізації навчального інформаційного середовища у вигляді електронного підручника з фізики / С. О. Лещук, П. М. Маланюк // Наукові записки Тернопільського державного педагогічного університету. Серія : Педагогіка. – 2002. – № 6. – С. 169 – 173.
138. Лещук С. О. Навчально-інформаційне середовище як засіб організації пізнавальної діяльності учнів / С. О. Лещук // Науковий часопис НПУ імені М. П. Драгоманова. Серія 2. Комп’ютерно-орієнтовані системи навчання. – 2004. – Вип. 1 (8). – С. 305 – 313.
139. Лукаш І. М. Формування інтелектуальних умінь старшокласників у процесі навчання інформатики / автореф. дис. … кандидата педагогічних наук за спеціальністю 13.00.02 – теорія та методика навчання інформатики / І. М. Лукаш. – К. : Національний педагогічний університет імені М. П. Драгоманова, 2002. – 21 с.
140. Лукаш І. М. Формування інтелектуальних умінь старшокласників у процесі навчання інформатики : дис. … канд. пед. наук : 13.00.02 /

І. М. Лукаш. – К. : Національний педагогічний ун-т ім. М. П. Драгоманова, 2003. – 304 с.

141. Лупан І. В. Комп’ютерні статистичні пакети : навчально-методичний посібник / І. В. Лупан, О. В. Авраменко. – Кіровоград, 2010. – 218 с.
142. Львов М. Алгебра з комп’ютером / М. Львов, Н. Львова. – К. : Шк. світ, 2007. – 128 с.
143. Маланюк П. М. Повышение эффективности самостоятельной работы учащихся при изучении физики на основании использования компьютерной техники : дисс. ... канд. пед. наук : 13.00.02 / П. М. Маланюк. – К., 1991. – 167 с.
144. Маланюк П. М. Про нові підходи до підбору матеріалу для викладання «Основ інформатики та обчислювальної техніки» / П. М. Маланюк , С. В. Мартинюк // Друга всеукраїнська конференція Молодих науковців «Інформаційні технології в науці та освіті». 18–20 квітня 2000 р. – Черкаси, 2000. – С. 46 – 47.
145. Малежик М. П. Модель оцінювання знань на основі дистанційних технологій / М. П. Малежик, О. В. Терещенко, В. М. Терещенко // Науковий часопис НПУ імені М. П. Драгоманова. Серія 5. Педагогічні науки : реалії та перспективи. – 2009. – Вип. 20. – С. 147 – 151.
146. Малежик М. П. Модель платформно-незалежної системи дистанційого оцінювання знань / М. П. Малежик, О. В. Терещенко, В. М. Терещенко // Науковий часопис НПУ імені М. П. Драгоманова. Серія 5. Педагогічні науки : реалії та перспективи. – 2010. – Вип. 23. – С. 320 – 324.
147. Малежик П. М. Система для дистанційного предметного тестування знань майбутніх вчителів технологій / П. М. Малежик, В. В. Даруга, Т. В. Сіткар // Вища освіта України : теоретичний та науково-методичний часопис. № 2 (додаток 2). Тематичний випуск «Науково-методичні засади управління якістю освіти у вищих навчальних закладах». – Луцьк : ВолиньПоліграф, 2013. – С. 357 – 364.
148. Маркова А. К. Психология профессионализма / А. К. Маркова. – М. : Знания, 1996. – 312 с.

149. Мартиненко С. Методи навчання та їх класифікація [Електронний ресурс] / С. Мартиненко, Л. Хоружа. – Режим доступу : http://osvita.ua/school/theory/780.
150. Матяш Н. В. Психология проектной деятельности школьников в условиях технологического образования / Н. В. Матяш. – Мозырь : Белый ветер, 2000. – 288 с.
151. Машбиц Е. И. Компьютеризация обучения : проблемы и перспективы / Е. И. Машбиц. – М., 1986. – 80 с.
152. Машина Тюрінга [Електронний ресурс] // Навчальні матеріали з інформатики. Основні поняття інформатики. – Режим доступу : http://www.ua5.org/osnovi/158-mashina-tjuringa.html.
153. Михалін Г. О. Проблема формування професійної культури майбутнього вчителя математики у наукових роботах академіка Жалдака Мирослава Івановича / Г. О. Михалін // Науковий часопис НПУ імені М. П. Драгоманова. Серія 2. Комп'ютерно-орієнтовані системи навчання. – 2012. – Вип. 13 (20). – С. 41 – 50.
154. Михалін Г. О. Вивчення основних елементарних функцій дійсної і комплексної змінної з використанням комп'ютерних засобів математики / Г. О. Михалін, С. Я. Деканов // Науковий часопис НПУ імені М. П. Драгоманова. Серія 2. Комп'ютерно-орієнтовані системи навчання. – 2010. – Вип. 9. – С. 49 – 71.
155. Морзе Н. В. Методика навчання інформатики. Ч. 1. Загальна методика навчання інформатики / Н. В. Морзе. – К. : Навчальна книга, 2003. – 256 с.
156. Морзе Н. В. Методика навчання інформатики. Ч. 4. Методика навчання основ алгоритмізації та програмування. – К. : Навчальна книга, 2004. – 368 с.
157. Морзе Н. В. Основи методичної підготовки вчителя інформатики : монографія / Н. В. Морзе. – К. : Курс, 2003. – 372 с.
158. Морзе Н.В. Методика навчання інформатики : посібник для студентів пед. університетів / Н. В. Морзе. – К. : Курс, 2002. – 895 с.
159. Мышление : прогресс, деятельность общение / под ред. А. В. Брушлинского. – М., 1982. – 287 с.

160. Немов Р. С. Психология : учеб. пособие для учащихся пед. училищ, студентов пед. институтов и работников подготовки, повышения квалификации и переподготовки пед. кадров / Р. С. Немов – М. : Просвещение, 1990. – 301 с.
161. Новик И. Б. Системный стиль мышления / И. Б. Новик. – М. : Знание, 1985. – 64 с.
162. Новоженов Ю. В. Объектно-ориентированные технологии разработки сложных программных систем / Ю. В. Новоженов. – М., 1996. – 114 с.
163. Окулов С. М. Основы программирования / С. М. Окулов. – М. : ЮНИМЕДИАСТАИЛ, 2002. – 424 с.
164. Олексюк В. Організація комп'ютерної локальної мережі / В. Олексюк, Н. Балик, А. Балик. – Тернопіль : Підручники і посібники, 2006. – 80 с.
165. Олексюк В. П. Створення та використання Інтернет-серверів у навчальному процесі / В. П. Олексюк // Науковий часопис НПУ імені М. П. Драгоманова. Серія 2. Комп'ютерно-орієнтовані системи навчання. – 2004. – Вип. 1 (8). – С. 313 – 320.
166. Орлов В. А. Структурно-функциональный исторический аспекты целостности личности / В. А. Орлов, Е. А. Андриянов, В. И. Коляда // В кн. Социальная форма движения : проблемы целостности. – Саратов : Изд.-во СГУ, 1990. – 177 с.
167. Основи програмування [Електронний ресурс] // Електронна бібліотека Libr.Org.Ua. – Режим доступу : http://libr.org.ua/books/74.html.
168. Основні засади розвитку вищої освіти України в контексті Болонського процесу (документи і матеріали 2003–2004 років) / За редакцією В. Г. Кременя ; Авторський колектив : М. Ф. Степко, Я. Я. Болюбаш, В. Д. Шинкарук, В. В. Грубінко, І. І. Бабин. – Тернопіль: Вид-во ТДПУ імені В. Гнатюка, 2004. – 147 с.
169. Паламарчук В. Ф. Школа учить мыслить / В. Ф. Паламарчук. – М. : Просвещение, 1989. – 224 с.
170. Парфьонова Н. Д. Нові підходи до використання вільно поширюваної системи комп'ютерної математики Махіма у навчанні функцій комплексної змінної [Електронний ресурс] / Н. Д. Парфьонова. –

Інформаційні технології і засоби навчання. – 2012. – № 1 (27). – Режим доступу до журналу : http://www.journal.iitta.gov.ua.

171. Петухова Л. Є. Актуальні питання формування інформатичних компетентностей майбутніх учителів початкових класів / Л. Є. Петухова, О. В. Співаковський // Комп'ютер у школі та сім'ї. – 2011. – № 1 (89). – С. 7 – 11.
172. Пильщиков В. Н. Машина Тюрінга и алгоритмы Маркова. Решение задач : учебно-методическое пособие / В. Н. Пильщиков, В. Г. Абрамов, А. А. Вылиток, И. В. Горячая. – М. : МГУ, 2006. – 47 с.
173. Підгорна Т. В. Структура інформатичних компетентностей / Т. В. Підгорна // Науковий часопис НПУ імені М. П. Драгоманова. Серія 2. Комп'ютерно-орієнтовані системи навчання. – 2012. – Вип. 12. – С. 109 – 116. – Режим доступу : http://nbuv.gov.ua/UJRN/ Nchnpu_2_ 2012_12_17.
174. Подгорецкая Н. А. Изучение приемов логического мышления у взрослых / Н. А. Подгорецкая. – М. : Изд-во МГУ, 1980. – 150 с.
175. Подгорецька Н. А. Вивчення прийомів логічного мислення у дорослих / Н. А. Подгорецька. – М. : Вид-во МГУ, 1980. – 147 с.
176. Пойа Д. Математическое открытие : Решение задач : основные понятия, изучение и преподавание / Д. Пойа. – М. : Наука, 1976. – 448 с.
177. Попов В. Б. Turbo Pascal для школьников : учебно-метод. пособие / В. Б. Попов. – М. : Финансы и статистика, 2010. – 352 с.
178. Поспелов Н. Н. Формирование мыслительных операций у старшеклассников. / Н. Н. Поспелов, И. Н. Поспелов. – М. : Педагогика, 1989. – 152 с.
179. Проблема компетенции и результатов образования (панельная дискуссия по теме конференции) [Электронный ресурс] / [И. Д. Фрумин, Г. А. Цукерман, Б. И. Хасан, Т. Шанин, Е. Л. Ленская, А. П. Тряпицина, К. Н. Поливанова, Б. Д. Эльконин] // Материалы X конференции. – Режим доступа : http://www.ippd.ru/resources/library?file=612.
180. Програма курсу за вибором «Основи алгоритмізації та програмування» для організації профільного навчання у старших класах загальноосвітніх навчальних закладів [Електронний ресурс] / Автори : Т. П. Караванова,

В. П. Костюков. – Режим доступу : http://www.mon.gov.ua/activity/education/zagalna-serednya/navchalni-programy.htmla.

181. Програми з інформатики для 7–11 класів на 2014–2015 рр. [Електронний ресурс]. – Режим доступу : http://www.mon.gov.ua/activity/education/zagalna-serednya/navchalni-programy.html.
182. Програмний засіб Gran [Електронний ресурс]. – Режим доступу : http://www.zhaldak.npu.edu.ua/index.php/prohramnyi-zasib-gran/.
183. Програмні засоби навчального призначення [Електронний ресурс]. – Режим доступу : http://www.kspu.edu/About/Faculty/FPhysMathemInformatics/ChairInformatics/Staff/Lvov.aspx.
184. Просис Дж. Строительство продолжается / Дж. Просис, М. Миллер. – CHICAGO PC Magazine. – 1994. – № 4. – С. 146 – 161.
185. Психологічний словник / За ред. В. І. Войтка. – К. : Вища школа, 1982. – 218 с.
186. Работин И. Познавательная активность школьников на уроке / И. Работин // Педагогика. – 1996. – № 3. – С. 123 – 125.
187. Равен Дж. Компетентность в современном обществе : выявление, развитие и реализация / Дж. Равен ; пер. с англ. – М. : Когито-Центр, 2002. – 396 с.
188. Рак В. І. Об’єктно-орієнтований підхід до створення та використання засобів сучасних комп’ютерних технологій при поглибленому вивченні інформатики / В. І. Рак, О. Б. Ящик // Матеріали всеукраїнської конференції «Проблеми та перспективи наук в умовах глобалізації». – ТНПУ ім. В. Гнатюка, 2012. – С. 291 – 297.
189. Раков С. А. Програмно-методичний комплекс DG як крок від традиційної до інформаційної технології навчання геометрії / С. А. Раков, В. П. Горох // Комп’ютер у школі і сім’ї. – 2003. – № 1. – С. 20 – 23.
190. Рамський Ю. С. Методичні основи вивчення експертних систем у школі / Ю. С. Рамський , Н. Р. Балик. – К. : Логос, 1997. – 114 с.
191. Рамський Ю. С. Складові інформаційної культури майбутнього вчителя математики / Ю. С. Рамський, М. А. Умрик // Науковий часопис НПУ

імені М. П. Драгоманова. Серія 2. Комп'ютерно-орієнтовані системи навчання. – 2011. – Вип. 11. – С. 16 – 25.

192. Рамський Ю. С. Активізація пізнавальної діяльності школярів засобами «ІнфоНІС» / С. О. Лещук, Ю. С. Рамський / Науковий часопис НПУ імені М. П. Драгоманова. Серія 2. Комп'ютерно-орієнтовані системи навчання. – 2007. – Вип. 5 (12). – С. 120 – 125.
193. Рамський Ю. С. Зміни в професійній діяльності вчителя в епоху інформатизації освіти / Ю. С. Рамський // Науковий часопис НПУ імені М. П. Драгоманова. Серія 2. Комп'ютерно-орієнтовані системи навчання. – 2007. – Вип. 5 (12). – С. 10 – 12.
194. Рамський Ю. С. Компоненти інформаційної культури майбутнього вчителя математики / Ю. С. Рамський, М. А. Умрик // Науковий часопис НПУ імені М. П. Драгоманова. Серія 2. Комп'ютерно-орієнтовані системи навчання. – 2011. – Вип. 11 (18). – С. 16 – 26.
195. Рамський Ю. С. Методика навчання основ об'єктно-орієнтованого програмування / Ю. С. Рамський, І. М. Лукаш // Комп'ютер у школі та сім'ї. – 2002. – № 1. – С. 3 – 7 ; № 2. – С. 3 – 8 ; № 3. – С. 7 – 13 ; № 4. – С. 17 – 22 ; № 5. – С. 10 – 17 ; № 6. – С. 16 – 21.
196. Рамський Ю. С. Методика навчання основ об'єктно-орієнтованого програмування. Об'єктно-орієнтований аналіз / Ю. С. Рамський, І. М. Лукаш// Комп'ютер у школі та сім'ї. – 2003. – № 1. – С. 3 – 9.
197. Рамський Ю. С. Методична підготовка вчителя інформатики та розвиток його фахових компетентностей / Ю. С. Рамський, Н. Р. Балик // Науковий часопис НПУ імені М. П. Драгоманова. Серія 2. Комп'ютерно-орієнтовані системи навчання. – 2009. – Вип. 7 (14). – С. 32 – 35.
198. Рамський Ю. С. Методична система формування інформаційної культури майбутніх вчителів математики : дис. … докт. пед. наук : 13.00.02 / Ю. С. Рамський ; Національний педагогічний ун-т імені М. П. Драгоманова. – К., 2013. – 650 с.
199. Рамський Ю. С. Підвищення рівня фундаментальної підготовки з інформатики майбутніх вчителів математики та інформатики // Науковий

часопис НПУ імені М. П. Драгоманова. Серія 2. Комп'ютерно-орієнтовані системи навчання. – 2010. – Вип. 9 (16). – С. 95 – 98.

200. Рамський Ю. С. Про роль математики і деякі тенденції розвитку математичної освіти в інформаційному суспільстві / Ю. С. Рамський, К. І. Рамська // Науковий часопис НПУ імені М. П. Драгоманова. Серія 2. Комп'ютерно-орієнтовані системи навчання. – 2008. – Вип. 6 (13). – 182 с.
201. Рамський Ю. С. Формування компетентностей майбутніх вчителів інформатики та математики у галузі моделювання / Ю. С. Рамський, М. В. Рафальська. // Науковий часопис НПУ імені М. П. Драгоманова. Серія 2. Комп'ютерно-орієнтовані системи навчання. – 2012. – Вип. 12. – С. 117 – 126.
202. Рєзіна О. В. Формування інформаційно-пошукових та дослідницьких умінь учнів старшої школи в процесі навчання інформатики : автореф. дис. … канд. пед. наук : 13.00.02 / О. В. Рєзіна ; Нац. пед. ун-т ім. М. П. Драгоманова. – К., 2005. – 20 с.
203. Роберт И. В. Современные технологии в образовании : дидактические проблемы; перспективы использования / И/ В/ Роберт. – М. : Школа-Пресс, 1994. – 204 с.
204. Роберт И. В. Теория и методика информатизации образования (психолого-педагогический и технологический аспекты) / И. В. Роберт. – М. : ИИО РАО, 2008. – 274 с.
205. Ротаєнко П. А. Мультимедійні засоби навчання / Авт. кол. за ред. В. М. Мадзігона, Ю. О. Дорошенка // Інформатизація середньої освіти : програмні засоби, технології, досвід, перспективи. – К. : Педагогічна думка, 2003. – С. 14 – 48.
206. Руденко В. Д. Курс інформатики (част. 2) Основи алгоритмізації і програмування : навчальний посібник / В. Д. Руденко. – К. : Фенікс, 2002. – 200 с.
207. Рынок систем дистанционного образования / [Электронный ресурс]. – Режим доступа : http://www.cnews.ru/reviews/free/edu/it_russia.
208. Салангина Н. Я. Реализация линии алгоритмизации в курсе «Языки и методы программирования» физико-математических специальностей

педвузов : автореф. дисс. ... канд. пед. наук / Н. Я. Салангина. – М., 1999. – 29 с.

209. Салангина Н. Я. Подготовка будущих учителей информатики к проведению внеурочной деятельности / Н. Я. Салангина // Информатика и образование. – 2011. – № 5. – С. 53 – 58.
210. Самоненко Ю. А. Психология и педагогика : учеб. пособие для вузов / Ю. А. Самоненко. – М. : ЮНИТА-ДАНА, 2001. – 277 с.
211. Самоненко Ю. А. Функции, содержание и дидактические условия формирования научных методологических знаний у школьников : дисс. … док. пед. наук / Ю. А. Самоненко. – М., 2002. – 236 с.
212. Сейдаметова З. С. Мови програмування в навчанні майбутніх програмістів / З. С. Сейдаметова, Л. О. Манжос // Науковий часопис НПУ імені М. П. Драгоманова. Серія 2. Комп'ютерно-орієнтовані системи навчання. – 2010. – Вип. 8 (15). – С. 35 – 41.
213. Семакин И. Г.Основы программирования : учебник / И. Г. Семакин, А. П. Шестаков. – М. : Мастерство, 2002. – 432 с.
214. Семакин И. Г. Информатика-10. Информатика-11 / И. Г. Семакин, Е. К. Хеннер. – М. : Лаборатория Базовых Знаний, 2001–2002. – 120 с.
215. Семакин И. Г. Основы программирования : учебник / И. Г. Семакин, А. П. Шестаков. – М. : Мастерство, 2002. – 432 с.
216. Семеріков С. О. Фундаменталізація навчання інформатичних дисциплін у вищій школі : монографія / С. О. Семеріков ; науковий редактор академік АПН України, д. пед. н., проф. М. І. Жалдак. – Кривий Ріг : Мінерал; К. : НПУ ім. М. П. Драгоманова, 2009. – 340 с.
217. Сергієнко В. П. Засоби і технології продукування навчальних інформаційних ресурсів / В. П. Сергієнко, М. П. Малежик, М. В. Закатнов // Науковий часопис НПУ імені М. П. Драгоманова. Серія 2. Комп'ютерно-орієнтовані системи навчання. – 2010. – Вип. 8 (15). – С. 29 – 35.
218. Сидоров А. В. Теория алгоритмов : машины Тюринга : учеб.-метод. пособие / А. В. Сидоров. – Кострома : КГУ им. Н. А. Некрасова, 2010. – 26 с.

219. Система психолого-педагогічних вимог до засобів інформаційно-комунікаційних технологій навчального призначення : монографія / [О. О. Гриб'юк, В. М. Дем'яненко, М. І. Жалдак, Ю. Г. Запорожченко, Т. І. Коваль, Г. М. Кравцов, Г. П. Лаврентьєва, В. В. Лапінський, С. Г .Литвинова, М. В. Пірко, М. В. Попель, К. І. Скрипка, О. В. Співаковський, А. С. Сухіх, В. П. Татауров, М. П. Шишкіна] ; за ред. М. І. Жалдака. – К. : Атіка, 2014. – 172 с.
220. Сінько Ю. І. Системи комп'ютерної математики та їх роль у математичній освіті / Ю. І. Сінько. – Інформаційні технології в освіті : зб. наук. праць / голов. ред. Співаковський О. В. та ін. – Херсон : Видавництво ХДУ, 2009. – Вип. 3. – С. 274 – 278.
221. Слєдзінський І. Ф. Основи інформатики : посібник для студентів / І. Ф. Слєдзінський, Я. П. Василенко. – Тернопіль : Навчальна книга – Богдан, 2003. – 160 с.
222. Слєдзінський І. Ф. Техніка обчислень і алгоритмізація : навчальний посібник / І. Ф. Слєдзінський, А. М. Ломакович, Ю. С. Рамський, Р. І. Зароський. – К. : Вища школа, 1991. – 199 с.
223. Словак К. І. Мобільні математичні середовища : сучасний стан та перспективи розвитку / К. І. Словак, С. О. Семеріков, Ю. В. Триус // Науковий часопис НПУ імені М. П. Драгоманова. Серія 2. Комп'ютерно-орієнтовані системи навчання. – 2012. – Вип. 12. – С. 102 – 108.
224. Смирнова-Трибульская Е. Н. Основы формирования информатических компетентностей в области дистанционного обучения : монографія / Е. Н. Смирнова-Трибульская. – Херсон : Айлант, 2007. – 704 с.
225. Смирнова-Трибульська Е. М. Деякі результати досліджень в галузі дистанційних форм навчання в підготовці, післядипломній діяльності вчителів на Херсонщині / Е. М. Смирнова-Трибульська // Науковий часопис НПУ ім. М. П. Драгоманова. Серія 2. Комп'ютерно-орієнтовані системи навчання. – 2007. – Вип. 5 (12) – С. 13 – 27.
226. Співаковський О. В. Алгоритмізація та програмування. Енциклопедичне видання : навчально-методичний посібник / О. В. Співаковський. – К. : ТОВ Редакція «Комп'ютер», 2007. – 128с.

227. Співаковський О. В. Педагогические программные средства : объектно-ориентированный подход / О. В. Співаковський // Информатика и образование. – 1990. – № 2. – С. 71 – 73.
228. Співаковський О.В. Управління інформаційними технологіями вищих навчальних закладів : навчальний посібник ; видання третє, доповнене / О. В. Співаковський, Я. Б. Федорова, О. О. Глущенко, Н. А. Кудас. – Херсон : Айлант, 2010. – 302 с.
229. Співаковський О. В. Вступ до об'єктно-орієнтованих технологій : навчальний посібник / О. В. Співаковський, М. С. Львов. – Херсон : Айлант, 2000. – 210 с.
230. Співаковський О. В. Вступ до об'єктно-орієнтованого програмування / О. В. Співаковський, М. С. Львов. – Херсон, Айлант, 2001. – 210 с.
231. Співаковський О. В. Основи алгоритмізації та програмування : навчальний посібник / О. В. Співаковський, М. С. Львов. – Херсон, 1997. – 140 с.
232. Співаковський О. В. Основи алгоритмізації та програмування. Обчислювальний експеримент. Розв'язання проблем ефективності в алгоритмах пошуку та сортування : навчальний посібник / [О. В. Співаковський, Н. В. Осипова, М. С. Львов, К. В. Бакуменко]. – Херсон : Айлант, 2011. – 100 с. : іл.
233. Співаковський О. В. Шляхи удосконалення курсу «Основи алгоритмізації та програмування» у педагогічному вузі / О. В. Співаковський, М. С. Львов. – Комп'ютер у школі та сім'ї. – 2001. – № 4. – С. 22 – 24.
234. Спірін О. М. Інформаційно-комунікаційні та інформатичні компетентності як компоненти системи професійно-спеціалізованих компетентностей вчителя інформатики [Електронний ресурс] / О. М. Спірін // Інформаційні технології і засоби навчання. – 2009. – № 5 (13). – Режим доступу : http://www.ime.edu-ua.net/em.htm.
235. Стариченко Б. Е. Теоретические основы информатики : учеб. пособие для студ. высш. пед. учеб. заведений. 2-е изд., перераб. и доп. / Б. Е. Стариченко. – Екатеринбург : Урал. гос. пед. ун-т, 2003. – 336 с.

236. Стариченко Б. Е. Системно-объектный подход к проектированию и реализации курса информатики в колледже : автореф. дисс. … канд. пед. наук / Б. Е. Стариченко. – Екатеринбург, 2003. – 31 с.
237. Строительство виртуальной образовательной сети / [Электронный ресурс]. – Режим доступа : http://ifets.ieee.org/russian/depository/v8_i4/html/1.html.
238. Талызина Н. Ф. Формирование познавательной деятельности учащихся / Н. Ф. Талызина. – М. : Знание, 1983. – 328 с.
239. Твердохліб І. А. Особливості розвитку логічного мислення школярів під час вивчення фізико-математичних дисциплін / І. А. Твердохліб // Психологічні проблеми сучасності : Тези VII науково-практичної конференції студентів та молодих вчених. Львів, 15–17 квітня 2010 р. – Львів, 2010. – С. 24 – 25.
240. Теория и практика онлайнового обучения : Learning Content Management Systems / [Электронный ресурс]. – Режим доступа : http://www.distance-learning.ru/db/el/74680D276CB4380DC32571D8002E91CE/doc.html.
241. Теплицький О. І. Динамічне графічне об’єктно-орієнтоване моделювання в мультимедіа-середовищі мобільного на- вчання Squeak / О. І. Теплицький, І. О. Теплицький, С. О. Семеріков // Науковий часопис Національного педагогічного університету імені М. П. Драгоманова. Серія 2. Комп’ютерно-орієнтовані системи навчання. – 2009. – Вип. 7 (14). – С. 49 – 54.
242. Теплицький О. І. Об’єктно-орієнтоване моделювання в Alice. Частина 1 / О. І. Теплицький ; за наук. ред. акад. НАПН України М. І. Жалдака. – К. : НПУ імені М. П. Драгоманова, 2011. – 56 с.
243. Тимофеевская М. Изучаем программирование / М. Тимофеевская. – СПб., 2002. – 384 с.
244. Триус Ю. В. Комбіноване навчання як інноваційна освітня технологія у вищій школі / Ю. В. Триус, І. В. Герасименко // Теорія та методика електронного навчання : збірник наукових праць. Вип. ІІІ. – Кривий Ріг : Видавничий відділ НметАУ, 2012. – С. 299 – 308.
245. Триус Ю. В. Комп’ютерно-орієнтовані методичні системи навчання : монографія / Ю. В. Триус. – Черкаси : Брама-Україна, 2005. – 400 с.

246. Триус Ю. В. Комп’ютерно-орієнтовані методичні системи навчання математичних дисциплін у вищих навчальних закладах : дис. ... доктора пед. наук : 13.00.02 – теорія і методика навчання інформатики / Ю. В. Триус ; Черкаський нац. ун-т ім. Богдана Хмельницького. – Черкаси, 2005. – 649 с.
247. Триус Ю. В. Програма-інтерпретатор алгоритмічних систем Маркова, Тюрінга, Поста / Ю. В. Триус, А. Ю. Дяченко // Науковий часопис НПУ імені М. П. Драгоманова. Серія 2. Комп’ютерно-орієнтовані системи навчання. – 2006. – Вип. 4 (11). – С. 28 – 41.
248. Тришина С. В. Информационная компетентность как педагогическая категория [Электронный ресурс] / С. В. Тришина // Журнал «Эйдос». – 2005. – № 10. – Режим доступа : http://www.eidos.ru/ journal/2005/0910-11.htm.
249. Угринович Н. Д. Преподавание курса «Информатика и ИКТ» в основной и старшей школе : методическое пособие / Н. Д. Угринович и др. – 3-е изд. – М. : БИНОМ. Лаборатория знаний, 2006. – 140 с.
250. Угринович Н. Д. Информатика и информационные технологии 10-11 / Н. Д. Угринович. – М. : Лаборатория Базовых Знаний, 2001. – 512 с.
251. Український педагогічний словник / С. У. Гончаренко. – К. : Либідь, 1997. – 375 с.
252. УСЕ : Універсальний словник-енциклопедія / Гол. ред. М. В. Попович. – Львів : Тека, 2006. – 1432 с.
253. Фаліна М. М. Машина Тьюрінга / М. М. Фаліна // Інформатика. – 2005. – № 26. – С. 12 – 15.
254. Філософський енциклопедичний словник / За ред. В. І. Шинкарука. – К. : Абрис, 2002. – 742 с.
255. Фіцула М. М. Педагогіка : навчальний посібник для студентів вищих педагогічних закладів освіти / М. М. Фіцула. – К. : Видавничий центр «Академія», 2000. – 544 с. (Альмаматер).
256. Формуймо навички 21 століття [Електронний ресурс]. – Режим доступу : http://ippo.org.ua/index.php?option=com_content&task=view&id=2468&Itemid=41.

257. Фридман Л. М. О некоторых вопросах использования задач в обучении / Л. М. Фридман, К. К. Джумаев. – М. : Педагогика, 1974. – № 6. – С. 12 – 16.
258. Фути К. Языки программирования и схемотехника СБИС / К. Фути, Н. Судзуки. –М. : Мир, 1988. – 223 с.
259. Харламов И. Ф. Педагогика / И. Ф. Харламов. – М. : Гардарики, 1999. – 520 с.
260. Хомік О. А. Історія розвитку інформаційних технологій в Україні. – [30 листопада 1998]. – Режим доступу : http://www.icfcst.kiev.ua/ SYMPOSIUM/Proceedings/Galdak.doc.
261. Хуторской А. В. Ключевые компетенции и образовательные стандарты : Доклад на Отделении философии образования и теоретической педагогики РАО 23 апреля 2002 г. [Электронный ресурс] / А. В. Хуторской ; Центр «ЭГщос». – Режим доступа : www.eidos.ru/news/coiTmet-dis.htm.
262. Хьюз Дж. Структурный подход к программированию / Дж. Хьюз, Дж. Мичтом. – М. : Мир, 1980. – 280 с.
263. Черников В. В. Формирование системного мышления у учащихся старших классов общеобразовательных учреждений : дисс. … канд. пед. наук / В. В. Черников. – М., 1998. – 149 с.
264. Шафрин Ю. А. Информационные технологии. Ч. 1–2. / Ю. А. Шафрин. – М. : Лаборатория Базовых Знаний, 1999. – 320 с.
265. Шолом Г. І. Роль задач у формуванні критичного мислення / Г. І. Шолом // Науковий часопис НПУ імені М. П. Драгоманова. Серія 2. Комп’ютерно-орієнтовані системи навчання. – 2011. – Вип. 11. – С. 120 – 125.
266. Элиенс А. Принципы объектно-ориентированной разработки программ. – 2-е изд. : пер. с англ. / А. Элиенс. – М. : Издательский дом «Вильямс», 2002. – 496 с.
267. Ягупов В. В. Педагогіка :навч. посібник / В. В. Ягупов. – К. : Либідь, 2002. – 560 с.
268. Якушев А. В. Анализ технологий и систем управления электронным обучением [Электронный ресурс]. – Режим доступа : http://inno.cs.msu.su/implementation/it-university/07/.

269. Яшанов С. М. Сучасні інформаційні технології в освіті : навч.-метод. посіб. / С. М. Яшанов, М. С. Яшанов ; Нац. пед. ун-т ім. М. П. Драгоманова. - Київ : Вид-во НПУ ім. М. П. Драгоманова, 2014. - 157 с.
270. Ящик О. Б. Об’єктно-орієнтований підхід при вивченні алгоритмізації та програмування / О. Б. Ящик // Підготовка фахівців інженерно-педагогічних спеціальностей : досвід, проблеми, перспективи : матеріали регіонального науково-практичного семінару / за ред. Р. М. Горбатюка – Тернопіль : ТНПУ ім. В. Гнатюка, 2013. – С. 124 – 126.
271. Ящик О. Б. Формування системно-логічного мислення у процесі навчання основ алгоритмізації та програмування з допомогою систем комп’ютерної математики / О. Б. Ящик // Інформаційні моделі, системи та технології : матеріали IV науково-технічної конференції Тернопільського національного технічного університету імені Івана Пулюя (м. Тернопіль, 15–16 травня 2014). – Тернопіль : ТНТУ ім. І. Пулюя, 2014. – С. 39.
272. Ящик О. Б. Використання старшокласниками комп’ютерного середовища Maple у процесі навчання основ алгоритмізації та програмування на рівні поглибленого вивчення інформатики / О. Б. Ящик // Засоби навчальної та науково-дослідної роботи : зб. наук. праць. / за заг. ред. проф. В.І. Євдокимова і проф. О. М. Микитюка. – Харків : ХНПУ ім. Г.С. Сковороди, 2013. – Вип. 41. – С. 76 – 88.
273. Ящик О. Б. Компетентнісний підхід у навчанні об'єктно-орієнтованого програмування, як основа підготовки учнів старших класів / О. Б. Ящик // Наукові записки Тернопільського національного педагогічного університету імені Володимира Гнатюка. Серія : Педагогіка. – 2011. – № 1. – С. 103 – 108.
274. Ящик О. Б. Формування системно-логічного мислення старшокласників як міждисциплінарна проблема / О. Б. Ящик // Наукові записки Тернопільського національного педагогічного уініверситету імені Володимира Гнатюка. Серія : Педагогіка, 2011. – № 5. – С. 137 – 145.
275. Ящик О. Б. Машина Тюрінга як універсальний виконавець алгоритмів та її застосування в процесі поглибленого вивчення алгоритмізації і основ програмування старшокласниками [Електронний ресурс] / О. Б. Ящик // Інформаційні технології і засоби навчання. Information Technologies and

Learning Tools. – 2016. – № 2 (52). – Режим доступу : http://journal.iitta.gov.ua/ index.php/itlt/article/view/1365.

276. Ящик О. Б. Принципи створення шкільного сайту методом проектів / О. Б. Ящик // Проблеми та перспективи наук в умовах глобалізації : матеріали всеукраїнської наукової конференції / Тернопіль, 2006. – С. 100 – 111.
277. Ящик О. Б. Система задач як засіб забезпечення розвитку системно-логічного мислення старшокласників в процесі поглибленого вивчення алгоритмізації та основ програмування / О. Б. Ящик // Науковий часопис НПУ імені М. П. Драгоманова. Серія 2. Комп'ютерно-орієнтовані системи навчання. – 2015. – Вип. 17 (24). – С. 75 – 81.
278. Ящик О. Б. Системи комп'ютерної математики в ієрархії засобів розв'язування математичних задач / О. Б. Ящик // Психолого-педагогічні проблеми сільської школи : зб. наук. праць / під. ред. Н. С. Побірченко. – Умань : ФОП Жовтий О.О., 2014. – Вип. 49. – С. 123 – 130.
279. Borsoon T. Harnessing the power of interactivity for instruction / In M. R. Simonson and C. Hargrave (Eds.) // Proceedings of the 1991 Convention of the Association for Educational Communications and Technology. Orlando, FL : Association for Educational Communications and Technology, 1991. – P. 103 – 117.
280. Brinda T. Didaktisches System for objektorientiertes Modellieren im Informatikunterricht der Sekundarstufe II : Doktor der Naturwissenschaften (Dr. rer. nat.) genehmigte Dissertation / Torsten Brinda ; University Siegen. – Siegen, 2004. – 279 p.
281. Fichman R. G. Object-oriented and conventional analysis and design methodologies / R. G. Fichman, C. F. Kemerer. – IEEE Computer, 1992. – № 25(10). – P. 22 – 39.
282. Heller Anne C. Ayn Rand and the World She Made / Anne C. Heller. – New York : Doubleday, 2009.
283. Henderson-Sellers B. A book of object-oriented knowledge / B. Henderson-Sellers. – Prentice-Hall, 1992. – 338 p.

284. Lipsitz J. S. Growing up forgotten : A review of research and programs concerning early adolescence / J. S. Lipsitz. – Toronto : Lexington Books. 1977. – 228 p.
285. LMS and LCMS : В чем разница? / [Электронный ресурс]. – Режим доступа : http://www.distance-learning.ru/db/el/B254358DE85FFE70C325723B0032F739/doc.html.
286. Maxwell J. W. Tracing the Dynabook : A Study of Technocultural Transformations : PhD Dissertation / John W. Maxwell ; The University of British Columbia. – Vancouver, 2006. – VIII+303 p.
287. McGregor J. Object-oriented software development : engineering software for reuse / J. McGregor, D. Sykes. – Van Nostrand Reinhold, 1992. – 415 p.
288. Resnick M. Thinking Like a Tree (and Other Forms of Ecological Thinking) / Mitchel Resnick. – International Journal of Computers for Mathematical Learning. – 2003. – Vol. 8, No. 1. – P. 43 – 62.
289. Resnick M. Turtles, Termites, and Traffic Jams : Explorations in Massively Parallel Microworlds / Mitchel Resnick. – Cambridge : The MIT Press, 1997. – 181 p.
290. Scratch : Programming for All / Mitchel Resnick, John Maloney, Andres Monroy Hernandez, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, Yasmin Kafai // Communications of the ACM. – 2009. – Vol. 52, No. 11. – P. 60 – 67.
291. Why Use MapleNet? [Електронний ресурс]. – Режим доступу : http://www.maplesoft.com/products/maplenet/uses.aspx.
292. Yashchyk O. High-school students' competencies formation in the area of algorithmization and programming by means of computer mathematics / Oleksandr Yashchyk // Australian Journal of Scientific Research, 2014, No. 2 (6) (July-December). Vol. III. «Adelaide University Press». Adelaide, 2014. – P. 343 – 350.

ДОДАТКИ

Додаток А

Анкета для учнів

1. Чи цікавить Вас вивчення системи комп'ютерної математики (СКМ) Maple?

а) так;

б) ні;

в) не дуже.

2. Що нового Ви дізналися у процесі вивчення СКМ Maple?

_____

_____

3. Які розділи (теми) в роботі з СКМ Maple сподобались Вам найбільше?

а) робота з виразами;

б) розв'язування рівнянь та їх систем;

в) розв'язування нерівностей та їх систем;

г) побудова графіків функцій;

д) анімація;

е) знаходження похідної;

ж)знаходження інтегралу;

з) побудова тривимірних графічних зображень.

4. Чи застосовували Ви на практиці (для підготовки до уроків чи самоперевірки) знання, отримані при вивченні Maple?

а) так, часто;

б) кілька разів;

в) ніколи.

5. Які навчальні цілі можна досягти з допомогою Maple?

а) набути вміння та навички розв'язувати практичні задачі з інших предметів за допомогою комп'ютера;

б) розвинути логічне мислення та навчитись створювати власні програмні продукти на основі СКМ Maple;

в) набути практичні вміння застосування інформаційних технологій в подальшій професійній діяльності.

6. Чи сприяє, на Вашу думку, вивчення Maple підвищенню інтересу до предметів фізико-математичного циклу?

а) так;

б) трохи;

в) ні.

7. Які професійні навички, на Вашу думку, Ви отримали при роботі з Maple?

а) реалізації основних алгоритмічних структур з допомогою операторів Maple;

б) розв'язування комбінованих задач;

в) створення графічних об'єктів та їх анімації;

г) розв'язування задач, використовуючи основні типи алгоритмічних структур: слідування, цикл, розгалуження;

д) опис та використання процедур і функцій при розв'язуванні задач.

8. Чи можливе застосування знань і досвіду роботи з використання Maple у Вашій майбутній професії?

а) так;

б) не виключено;

в) ні.

9. Чи хотіли б Ви продовжити вивчення СКМ Maple в майбутньому?

а) так;

б) можливо;

в) ні.

Додаток Б

Тематичне планування. Програмування в середовищі Maple

№	Тема	Кількість год.
	Алгоритмізація	**16**
1.	Поняття моделі. Моделювання.	1
2.	Алгоритми. Властивості алгоритмів. Форми подання алгоритму Файл	1
3.	Алгоритми та їх виконавці.	1
4.	Знайомство з середовищем програмування Scratch.	1
5.	ПРАКТИЧНА РОБОТА 1	1
6.	Знайомство з командами Scratch.	1
7.	ПРАКТИЧНА РОБОТА 2	1
8.	Робота із зображеннями в Scratch.	1
9.	ПРАКТИЧНА РОБОТА 3	1
10.	Побудова інформатичної моделі.	1
11.	ПРАКТИЧНА РОБОТА 4	1
12.	Формальне поняття алгоритму. Машина Тюрінга.	1
13.	ПРАКТИЧНА РОБОТА 5	1
14.	Базові алгоритмічні структури. Типи алгоритмів.	1
15.	ПРАКТИЧНА РОБОТА 6	1
16.	Тест. Алгоритми та їх виконавці.	1
	СКМ Maple	**11**
17.	Середовище Maple. Інтерфейс програми.	1
18.	Структура вікна Maple. Арифметичні операції, числа, константи і стандартні функції. Елементарні перетворення математичних виразів.	1
19.	ПРАКТИЧНА РОБОТА 1	1
20.	Функції в Maple. Операції оцінювання. Розв’язання рівнянь і нерівностей.	1
21.	ПРАКТИЧНА РОБОТА 2	1
22.	Побудова графіків. Двовимірні графіки. Тривимірні графіки. Анімація.	1
23.	ПРАКТИЧНА РОБОТА 3	1

24.	Математичний аналіз: диференціальне числення функції однієї і багатьох змінних (Обчислення границь. Диференціювання. Дослідження функції.).	1
25.	ПРАКТИЧНА РОБОТА 4	1
26.	Математичний аналіз: інтегральне числення функції однієї і багатьох змінних.	1
27.	ПРАКТИЧНА РОБОТА 5	1
	Елементи програмування в Maple	**20**
28.	Програмування лінійних алгоритмів. Команди введення-виведення. Оператор присвоєння. Програмування розгалужених алгоритмів. Умовний оператор if	1
29.	Програмування циклічних алгоритмів. Оператори for, while, in.	1
30.	Застосування циклічних структур для розв'язання задач.	1
31.	Векторизація циклів. Вкладені цикли. Оператор пропуску елементу next. Оператор переривання циклу break.	1
32.	ПРАКТИЧНА РОБОТА 6	1
33.	Лінійна алгебра (Векторна алгебра. Дії з матрицями. Системи лінійних рівнянь. Матричні рівняння.)	1
34.	ПРАКТИЧНА РОБОТА 7	1
35.	Масиви, вкладені цикли, візуалізація масивів даних.	1
36.	ПРАКТИЧНА РОБОТА 8	1
37.	Вивчення функцій для опрацювання матриць. Метод Жордана-Гауса.	1
38.	ПРАКТИЧНА РОБОТА 9	1
39.	Функції користувача. Процедури.	1
40.	Програмування символьних операцій.	1
41.	ПРАКТИЧНА РОБОТА 10	1
42.	Створення користувацьких бібліотек програмних процедур	1
43.	Елементи об'єктно-орієнтованого програмування.	1
44.	Проектна діяльність	3
	Підсумкове тестування	**1**
Всього		***48***

Додаток В
ПРАКТИЧНІ РОБОТИ
ПРАКТИЧНА РОБОТА 1.
І. Структура вікна Maple. Арифметичні операції, числа, константи і стандартні функції. Елементарні перетворення математичних виразів

1. Структура вікна Maple.
2. Арифметичні операції, цілі і раціональні числа і константи Maple.
3. Синтаксис команд. Стандартні функції.
4. Перетворення математичних виразів.

§1. Структура вікна Maple

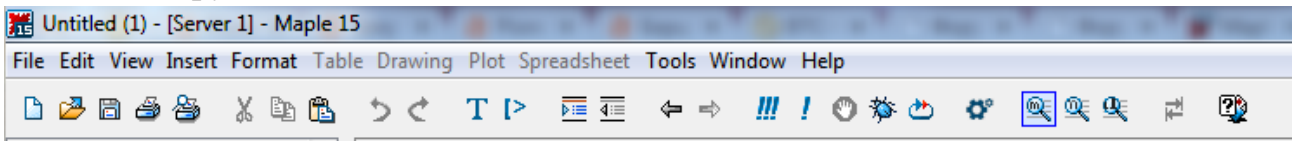
Система комп'ютерної алгебри — це комп'ютерна програма чи пакет програм, що допомагає виконувати найрізноманітніші математичні операції та перетворення алгебраїчних виразів заданих в чисельній та символьній (змінні, функції, поліноми, матриці тощо) формах. Сучасні системи містять функції практично з усіх розділів сучасної математики, підтримують інтерактивну візуалізацію, одну чи кілька мов програмування, і часто дозволяють комбінувати алгоритми, математичні формули, текст, графіку, діаграми чи анімацію зі звуком, а також результати обчислення в одному файлі.

Maple – комерційна система комп'ютерної алгебри від компанії Waterloo Maple Inc. Першу версію було розроблено та оприлюднено в 1980-му році групою Symbolic Computation Group з університету Ватерлоо, місто Ватерлоо, Онтаріо, Канада. Остання версія містить понад 5000 функцій для більшості розділів сучасної математики, моделювання та інтерактивної візуалізації, підтримує мову програмування Maple, і дозволяє комбінувати алгоритми, результати обчислення, математичні формули, текст, графіку, діаграми та анімацію зі звуком в електронному документі.

Для того, щоб запустити Maple, необхідно в Головному меню Windows вибрати в групі «Програми» назву даного застосування: Maple.

Maple є типовим вікном Windows, яке складається з Рядка назви, Основного меню, Панелі інструментів, Робочого поля і Рядка стану, а також Лінійки і Смуг прокрутки.

Вигляд фрагмента вікна Maple 15, що містить Рядок назви, Основне меню, Панель інструментів:



Пункти Основного меню:

File (Файл) містить стандартний набір команд для роботи з файлами, наприклад: зберегти файл, відкрити файл, створити новий файл і т. д.

Edit (Правка) містить стандартний набір команд для редагування тексту, наприклад: копіювання, видалення виокремленого тексту в буфер обміну, відміна команди і т. д.

View (Вигляд) – містить стандартний набір команд, керівників структурою вікна Maple.

Insert (Вставка) – служить для вставки полів різних типів: математичних текстових рядків, графічних два і тривимірних зображень.

Format (Формат) – містить команди оформлення документа, наприклад: установка типу, розміру і стилю шрифту.

Tools (Інструменти) – служить для доповнення сокупності інструментів новими та їх налаштування.

Windows (Вікно) – служить для переходу з одного робочого аркуша в іншій.

Help (Довідка) – містить детальну довідкові відомості про Maple.

Робота в Maple проходить в режимі сесії – користувач вводить пропозиції (команди, вирази, процедури), які сприймаються умовно і опрацьовуються Maple. Робоче поле поділяється на три частини:

1) ***межа введення*** – складається з командних рядків. Кожен командний рядок починається з символу >;
2) ***межа виводу*** – містить результати опрацювання введених команд у вигляді аналітичних виразів, графічних об'єктів або повідомлень про помилку;
3) ***місце текстових коментарів*** – містить текстові дані, шо можуть пояснити виконувані процедури. Текстові рядки не сприймаються Maple і ніяк не обробляються.

Для того, щоб перемкнути командний рядок в текстовий, потрібно на Панелі інструментів натискувати мишею на кнопку <Text>.

Вступ

1. Запустіть Maple.
2. Після запуску Maple перший рядок є командним. Переведіть його в текстовий. Наберіть в цьому рядку: «Практична робота №1» і назва теми. Перейдіть на наступний рядок, натискуючи Enter.
3. У новому рядку наберіть «Виконав учень» і своє прізвище. Натискуйте Enter.
4. На наступному рядку наберіть «Завдання №1».
5. Збережіть свій файл на диску. Для цього в меню Fail виберіть пункт Save і наберіть ім'я вашого файлу у вигляді: Фамілія_1, де вказується ваше прізвище і 1 – номер практичної роботи.

6. Після цього в наступному рядку наберіть текст: «Файл із завданнями практичної роботи №1 збережений під ім'ям: Фамілія_n».

Надалі виконання кожної практичної роботи повинне оформлятися в такий спосіб. На початку кожної практичної роботи потрібно набирати текст: «Практична робота N», N – номер теми. Виконання кожного завдання потрібно починати з текстового коментаря: «Завдання N». Для правильності обчислень перед виконанням кожного пункту завдання потрібно виконувати команду restart. Перед виконанням контрольних завдань потрібно набирати в текстовому режимі «Контрольні завдання». Після закінчення виконання роботи необхідно зберегти файл зі всіма виконаними завданнями на диск. Ім'я вашого файлу набирається у вигляді: Фамілія_n, де вказується ваше прізвище і N – номер теми.

§2. Арифметичні операції. Цілі і раціональні числа, константи в Maple

Математичні константи і арифметичні операції.

Основні математичні константи:

Pi – число 3,14…;

I – уявна одиниця і;

infinity – нескінченність;

Gamma – константа Ейлера ($\gamma \approx 0.57721566490153286060$.);

true, false – логічні константи, означаючі істинність і помилковість вислову.

Знаки арифметичних операцій:

+ - додавання; – - віднімання;

***** - множення; / - ділення;

^ - піднесення до степеня; ! – факторіал.

знаки порівняння: <, >, >=,<=, <> =.

Комплексні, цілі і раціональні числа.

Числа в Maple бувають дійсні (real) і комплексні (complex). Комплексне число записується у формі z=x+iy, і в командному рядку такий запис повинен виглядати так:

> z:=x+I*y;

Дійсні раціональні числа поділяються на цілі і дробові. Ціле число (integer) визначається своїми десятковими знаками, що утворюють десятковий запис цього числа. Раціональне число може бути заданим в 3-х видах:

1) раціональним дробом з використанням оператора ділення, наприклад: 28/70;
2) з плаваючою комою (**float**), наприклад: 2.3;
3) у показниковій формі, наприклад: 1,602*10^(-19) означає 1,602(10^{-19}).

Для того, щоб отримати наближене значення раціонального дробу у вигляді десяткового числа з плаваючою комою, потрібно знаменник цього дробу записати у вигляді числа з плаваючою комою. Приклад:

> **75/4;**

$$\frac{75}{4}$$

> 75/4.0;

18.75000000

У Maple можна записати букви грецького алфавіту в поліграфічному вигляді. Для цього в командному рядку набирається назва грецької букви. Наприклад, буква α вийде, якщо набрати alpha

β **- beta, - epsilon, - eta, - theta, - lambda, -xi, - chi.**

Великі грецькі букви можна записати, якщо набирати назву грецької букви із великої, наприклад, аби отримати Ω, потрібно набрати Omega. Грецькі букви також можна набирати за допомогою спеціального меню.

Приклади обчислень

1. Перейдіть в текстовий режим і наберіть «Завдання №2». Після не забудьте перейти в режим командного рядка.
2. Обчисліть значення . Для цього в командному рядку наберіть:

> (sqrt(6+2*sqrt(5)) -sqrt(6-2*sqrt(5)))/sqrt(3);

і натисніть Enter. В результаті вийде точне значення $\frac{2}{3}\sqrt{3}$.

3. Наберіть формули. Для цього в командному рядку наберіть:

> **omega=theta/t; abs(f(x) -delta) <epsilon;**

натисніть Enter:

$$\omega = \frac{\theta}{t}$$

$$|-f(x) + \delta| < \varepsilon$$

§3. Синтаксис команд. Стандартні функції

Синтаксис команд.

Стандартна команда Maple складається з імені команди та її параметрів, вказаних в круглих дужках: command(p1, p2). В кінці кожної команди має бути знак (;) або (:). Роздільника (;) означає, що в місці виводу після виконання цієї

команди буде одразу видано результат. Роздільник (:) використовується для відміни виводу, тобто коли команда виконується, але її результат на екран не виводиться.

Символ відсотка (%) служить для виклику попередньої команди. Цей символ відіграє роль короткострокової заміни попередньої команди з метою скорочення запису. Приклад використання (%):

> **a+b;**

$$a+b$$

> **%+c;**

$$a+b+c.$$

Для присвоювання змінній заданого значення використовується знак (:=).

Коли програма Maple запускається, вона не має жодної команди, повністю завантаженої в пам'ять. Велика частина команд мають покажчики їх знаходження, і при виклику вони завантажуються автоматично. Інші команди знаходяться в стандартній бібліотеці і перед виконанням обов'язково мають бути викликані командою readlib(command), де command – ім'я команди, що викликається. Остання частина процедур Maple міститься в спеціальних бібліотеках підпрограм, званих пакетами. Пакети необхідно підвантажувати при кожному запуску файлу з командами з цих бібліотек. Є два способи виклику команди з пакету:

1) можна завантажити весь пакет командою with(package) де package – ім'я пакету;
2) виклик який-небудь однієї команди command з будь-якого пакету package можна здійснити, якщо набрати команду в спеціальному форматі:
   > **package[command](options);**

де спочатку записується назва пакету package, з якого треба викликати команду, а потім в квадратних дужках набирається ім'я самої команди command, і після чого в круглих дужках вказуються параметри options даної команди.

До бібліотек підпрограм Maple належать, наприклад, наступні пакети: linalg – містить операції лінійної алгебри; geometry – розв'язування завдань планіметрії; geom3d – розв'язування завдань стереометрії; student – містить команди, завдання, що дозволяють провести поетапне розв'язання, в аналітичному вигляді з проміжними обчисленнями.

Maple містить величезну кількість спеціальних функцій, таких, як Бессельови функції, Ейлерови бета- і гамма – функції, інтеграл помилок, еліптичні інтеграли, різні ортогональні поліноми.

За допомогою функції exp(x) визначається число e=2.718281828, записавши exp(1).

Стандартні функції.

Стандартні функції Maple	
Математичний запис	Запис в Maple
e^x	**exp(x)**
$\ln x$	**ln(x)**
$\lg x$	**log10(x)**
$\log_a x$	**log[a](x)**
$\sqrt{x}$	**sqrt(x)**
$\lvert x \rvert$	**abs(x)**
$\sin x$	**sin(x)**
$\cos x$	**cos(x)**
$\mathrm{tg}\, x$	**tan(x)**
$\mathrm{ctg} x$	**cot(x)**
$\sec x$	**sec(x)**
$\mathrm{cosec} x$	**csc(x)**
$\arcsin x$	**arcsin(x)**
$\arccos x$	**arccos(x)**
$\mathrm{arctg} x$	**arctan(x)**
$\mathrm{arcctg} x$	**arccot(x)**
$\mathrm{sh}\, x$	**sinh(x)**
$\mathrm{ch} x$	**cosh(x)**
$\mathrm{th} x$	**tanh(x)**
$\mathrm{cth} x$	**coth(x)**
$\delta(x)$ - функція Дирака	**Dirac(x)**
$\theta(x)$ - функція Хевіссайда	**Heaviside(x)**

Приклади обчислень

1. Перейдіть в текстовий режим і наберіть «Завдання №3». Після не забудьте перейти в режим командного рядка.
2. Обчисліт ctg (π/3) + tg14(π/3). Для цього наберіть в командному рядку:

   > cot(Pi/3)+tan(14*Pi/3);

   Натисніть Enter. В результаті в місці виводу повинне з'явитися число:

$$-\frac{2}{3}\sqrt{3}\ .$$

3. Обчисліть. Для цього наберіть в командному рядку:

   > combine((sin(Pi/8))^4+(cos(3*Pi/8))^4+(sin(5*Pi/8))^4+ (cos(7*Pi/8))^4);

   Натискуйте Enter. (значення команди combine – перетворювати вирази, наприклад, з мірами). В результаті в місці виводу повинне з'явитися число:

$$\frac{3}{2}\ .$$

§4. Перетворення математичних виразів

Maple включає широкий набір команд для проведення аналітичних перетворень математичних формул. До них належать такі операції, як зведення подібних, розкладання на множники, розкриття дужок, зведення раціонального дробу до нормального вигляду і багато інших.

Виокремлення частин виразів.

Математична формула, над якою проводитимуться перетворення, записується в наступній формі: > **eq:=exp1=exp2**; де **eq** – довільне ім'я виразу, **exp1** – умовне позначення лівої частини формули, **exp2** – умовне позначення правої частини формули.

Виокремлення в правій частині виразу здійснюється командою **rhs(eq)**, виокремлення в лівій частині виразу – командою **lhs(eq)**. приклад:

> **eq:=a^2-b^2=c;**

$$a^2 - b^2 = c$$

> **lhs(eq);**

$$a^2 - b^2$$

> **rhs(eq);**

$$c$$

Якщо заданий раціональний дріб вигляду a/b, то можна виокремити її чисельник і знаменник за допомогою команд numer і denom, відповідно. Приклад:

> f:=(a^2+b)/(2*a-b);

$$f := \frac{a^2 + b}{2a - b}$$

> **numer(f);**

$$a^2 + b$$

> **denom(f);**

$$2a - b$$

<u>Тотожні перетворення виразів.</u>

Розкриття дужок виразу eq здійснюється командою **expand(eq)**. Приклад:

> **eq:=(x+1)*(x-1)*(x^2-x+1)*(x^2+x+1);**

$$eq := (x+1)(x-1)(x^2 - x + 1)(x^2 + x + 1)$$

> **expand(eq);**

$$x^6 - 1$$

Розкладання многочлена на множники здійснюється командою **factor(eq)**.

Приклад:

> **p:=x^5-x^4-7*x^3+x^2+6*x;**

$$p := x^5 - x^4 - 7x^3 + x^2 + 6x$$

> **factor(p);**

$$x(x-1)(x-3)(x+2)(x+1)$$

Команда expand може мати додатковий параметр, що дозволяє при розкритті дужок залишати певний вираз без змін. Наприклад, хай потрібний кожен доданок виразу помножити на вираз (x+a). Тоді в командному рядку потрібно написати:

> **expand((x+a)*(ln(x)+exp(x)-y^2), (x+a));**

$$(x+a)\ln x + (x+a)e^x - (x+a)y^2$$

Дріб можна привести до нормального вигляду за допомогою команди normal(eq). Наприклад:

> **f:=(a^4-b^4)/((a^2+b^2)*a*b);**

$$f := \frac{a^4 - b^4}{(a^2 + b^2)ab}$$

> normal(f);

$$\frac{a^2 - b^2}{ab}$$

Спрощення виразів здійснюється командою simplify(eq).

Приклад:

> eq:=(cos(x) -sin(x))*(cos(x)+sin(x)):

> simplify(eq);

$$2\cos(x)^2 - 1$$

Приведення подібних членів у розв'язанні здійснюється командою collect(exp,var), де exp – вираз, var – ім'я змінної, відносно якої потрібно збирати подібні. У команді simplify як параметри можна вказати, які вирази перетворювати. Наприклад, при вказівці simplify(eq,trig) проводитиметься спрощення при використанні великого числа тригонометричних співвідношень. Стандартні параметри мають назви: power – для степиневих перетворень; radical або sqrt – для перетворення кореня; exp – перетворення експонент; ln – перетворення логарифмів. Використання параметрів набагато збільшує ефективність команди simplify.

Об'єднати показники степеневих функцій або спростити тригонометричні функції можна за допомогою команди combine(eq, param), де eq – вираз, param – параметри, вказуючі, в який тип функцію перетворити, наприклад, trig – для тригонометричних, power – для степиневих. Приклад:

> combine(4*sin(x)^3, trig);

$$-\sin(3x) + 3\sin(x)$$

Для спрощення виразів, що містять не лише квадратні корені, але і корені інших степенів, краще використовувати команду radnormal(eq). Приклад:

> sqrt(3+sqrt(3)+(10+6*sqrt(3))^(1/3))=

radnormal(sqrt(3+sqrt(3)+(10+6*sqrt(3))^(1/3)));

$$\sqrt{3 + \sqrt{3} + (10 + 6\sqrt{3})^{1/3}} = 1 + \sqrt{3}$$

За допомогою команди convert(exp, param), де exp – вираз, який буде перетворено у вираз вказаного типу param. Зокрема, можна перетворити вираз, що містить sin(x), і cos(x), у вираз, що містить лише tg(x), якщо вказати як параметр tg, або, навпаки, вираз, який містить tg(x), ctg(x) можна перевести в sin(x) і cos(x), якщо в параметрах вказати sin cos.

Взагалі, команда convert має ширше призначення. Вона здійснює перетворення виразу одного типу в іншій. Наприклад: **convert(list, vector)** – перетворення деякого списку **list** у вектор з тими ж елементами; **convert(expr, string)** – перетворення математичного виразу в його текстовий запис. Для виклику детальних даних про призначення параметрів команди convert потрібно звернутися до довідкової системи, набравши convert[termin].

Якщо ви забули параметри якої-небудь команди, то можна скористатися довідковою системою Maple. Для виклику довідки по конкретній команді, потрібно виокремити набране ім'я цієї команди і натискувати клавішу F1. Якщо команда набрана правильно, то з'явиться опис цієї команди (у більшості версій Maple допомога англійською мовою).

Приклади перетворень виразів

1. Перейдіть в текстовий режим і наберіть «Завдання №4». Після цього перейдіть в режим командного рядка. Перед виконанням кожного пункту цього завдання обов'язково набирайте команду оновлення restart;
2. Розкладання полінома x^3+4x^2+2x-4 на множники. Для цього наберіть в командному рядку:

   > factor(x^3+4*x^2+2*x-4);

Після натиснення клавіші Enter повинне вийти:

$$(x+2)\left(x^2+2x-2\right)$$

3. Спростити вираз $\frac{1+\sin(2x)+\cos(2x)}{1+\sin(2x)-\cos(2x)}$. Наберіть:

> eq:=(1+sin(2*x)+cos(2*x))/(1+sin(2*x) -cos(2*x)):
> convert(eq, tan):
> eq=normal(%);

$$\frac{1+\sin(2x)+\cos(2x)}{1+\sin(2x)-\cos(2x)}=\frac{1}{\tan(x)}.$$

4. Спростити вираз. Для цього наберіть:

> eq:=3*(sin(x)^4+cos(x)^4)-2*(sin(x)^6+cos(x)^6):
> eq=combine(eq, trig);

Хід роботи:

1. Виконаєте всі контрольні завдання. Перед їх виконанням вкажіть в текстовому режимі «Контрольні завдання». Результати виконання завдань покажіть вчителю.
2. Збережіть файл зі всіма виконаними завданнями на диск.
3. Дайте відповідь на контрольні запитання.

Контрольні завдання:

1. Обчислити: $\frac{1}{2}e^x - \frac{1}{2}\frac{1}{e^x}$.
2. Обчислити: $\frac{(n+1)^3}{3} - \frac{(n+1)^2}{2} + \frac{n}{6} + \frac{1}{6}$. n=1..6
3. Обчислити точне і наближене значення виразу: $1 + \frac{x}{1!} + \frac{x^2}{2!} + \cdots + \frac{x^n}{n!}$ $(n = 30)$
4. Записати формули: $f := \frac{e^{\ln(xe^x \sin^2(y)) - x}}{x} + \cos^2(y)$; $f := y^3\left(\frac{y^2 + 2y + 1 + 8y^3}{y^3(y+1)} - \frac{8}{y+1}\right) - 1 - y$.
5. Розкласти на множники поліном $x^4 - 10x^3 + 35x^2 - 50x + 24$.
6. Спростити вираз 1/x+x/(x+1); sin(x)+1/sin(x)^2.

Контрольні запитання:

1. Що таке Maple і для чого він призначений?
2. Якими є основні елементи вікна Maple?
3. На які умовні частини ділиться робоче поле Maple і що в цих частинах відображується?
4. Як перевести командний рядок в текстовий і навпаки?
5. У якому режимі проходить сеанс роботи в Maple?
6. Перерахуєте пункти основного меню Maple і їх призначення.
7. Яке стандартне розширення привласнюється файлу робочого аркуша Maple?
8. Як представляються в Maple основні математичні константи?
9. Опишіть види представлення раціонального числа в Maple.
10. Як набути наближеного значення раціонального числа?
11. Якими розділовими знаками закінчуються команди в Maple і чим вони відрізняються?
12. Якою командою здійснюється виклик бібліотеки підпрограм?
13. Яким є призначення команд factor, expand, normal, simplify, combine, convert?

ПРАКТИЧНА РОБОТА 2.

II. Функції в Maple. Операції оцінювання. Розв'язання рівнянь і нерівностей

1. Способи задання функцій. Заміна змінних.
2. Операції оцінювання.
3. Розв'язання рівнянь.
4. Розв'язання нерівностей.

§1. Способи задання функцій. Заміна змінних

У Maple є декілька способів задання функції.

Спосіб 1. Визначення функції за допомогою оператора присвоєння (:=), наприклад:

> f:=sin(x)+cos(x);

$$f := \sin(x) + \cos(x)$$

Якщо задати конкретне значення змінної x, то вийде значення функції f для цього x. Наприклад, якщо продовжити попередній приклад і обчислити значення f, то потрібно записати:

> x:=Pi/4;

$$x := \frac{\pi}{4}$$

> f;

$$\sqrt{2}$$

Після виконання цих команд змінна x має задане значення .

Аби назовсім не присвоювати змінної конкретного значення, зручніше використовувати команду підстановки subs({x1=a1, x2=a2., },f), де у фігурних дужках вказуються змінні x_i і їх нові значення a_i (i=1,2.), які потрібно підставити у функцію *f*. Наприклад:

> f:=x*exp(-t);

$$f := xe^{(-t)}$$

> subs({x=2,t=1},f);

$$2e^{(-1)}$$

Всі обчислення в Maple за замовчуванням проводяться символьно, тобто результат міститиме в явному вигляді ірраціональні константи, такі як ***e,*** та інші. Аби набути наближеного значення у вигляді числа з плаваючою комою, потрібно використовувати команду **evalf(expr,t)**, де **expr** – вираз, **t** – точність, тобто кількість знаків після коми. Наприклад, продовжуючи попередній приклад, обчислимо набутого значення функції приблизно:

> evalf(%);

0.7357588824

Тут використаний символ (%) для виклику результату попередньої команди.

Спосіб 2. Визначення функції за допомогою функціонального оператора, який ставить у відповідність набору змінних (*x1*, *x2*) одне або декілька виразів

(*f1, f2*). Наприклад, визначення функції двох змінних за допомогою функціонального оператора виглядає таким чином:

> f:=(x,y)->sin(x+y);

$$f := \sin(x + y)$$

Звернення до цієї функції здійснюється найбільш звичним в математиці способом, коли в дужках замість аргументів функції вказуються конкретні значення змінних. Продовжуючи попередній приклад обчислимо значення функції:

> f(Pi/2,0);

1

Спосіб 3. За допомогою команди **unapply(expr,x1,x2)**, де **expr** – вираз, *x1, x2* – набір змінних, від яких воно залежить, можна перетворити вираз **expr** у функціонального оператора. Наприклад:

> f:=unapply(x^2+y^2,x,y);

$$f := (x, y) -> x^2 + y^2$$

> f(-7,5);

74

У Maple є можливість визначення неелементарних функцій вигляду

$$f(x) = \begin{cases} f_1(x),\ x < a_1 \\ f_2(x),\ a_1 < x < a_2 \\ \text{..... ....} \\ f_n(x),\ x > a_n \end{cases}$$

за допомогою команди

> piecewise(cond_1,f1, cond_2, f2).

Наприклад, функція

$$f(x) = \begin{cases} 0,\ x < 0 \\ x,\ 0 \le x < 1 \\ \sin x,\ x \ge 1 \end{cases}$$

записується таким чином:

> f:=piecewise(x<0, 0, 0<=x and x<1, x, x>=1, sin(x));

$$f := \begin{cases} 0 & x < 0 \\ x & -x \le 0 \text{ and } x - 1 < 0 \\ \sin x & 1 \le x \end{cases}$$

Приклади здійснення операцій

Не забувайте, що виконання всіх подальших завдань повинно починатися з текстового рядка, що містить «Завдання №», де № – номер завдання. Також пам'ятаєте, що для правильності обчислень перед виконанням кожного пункту завдання потрібно виконувати команду restart. Перед виконанням контрольних завдань потрібно набирати в текстовому режимі «Контрольні завдання». Ці правила оформлення стосується всіх практичних робіт.

1. Запустіть Maple. Переведіть перший рядок в текстову і наберіть в ній: «Практична робота №2». Натисніть Enter. Рядком нижче наберіть: «Виконав учень ...» і своє прізвище, а на наступному рядку наберіть: «Завдання №1».
2. Приклад переходу до полярних координат і спрощення отриманого виразу. Наберіть:

> f:=sqrt(1-x^2-y^2);

$$f = \sqrt{1 - x^2 - y^2}$$

> f:=subs({x=rho*cos(phi),y=rho*sin(phi)},f);

$$f = \sqrt{1 - \rho^2 \cos(\phi)^2 - \rho^2 \sin(\phi)^2}$$

> f:=simplify(%);

$$f = \sqrt{1 - \rho^2}$$

3. Приклад операції додавання функцій. Наберіть:

> f:=piecewise(x<-1, x -1<=x and x<1 -x^2, x>=1, -x);

$$f := \begin{cases} x & x < -1 \\ -x^2 & -1 - x \le 0 \text{ and } x - 1 < 0 \\ -x & 1 \le x \end{cases}$$

> %+x: simplify(%);

$$\begin{cases} 2x & x < -1 \\ x - x^2 & x \le 1 \\ 0 & 1 < x \end{cases}$$

§2. Команди обчислення числових виразів

Обчислення дробових виразів.

У Maple є наступні команди обчислення дробових виразів:

frac(expr) – обчислення дробової частини виразу **expr;**

trunc(expr) – обчислення цілої частини виразу **expr;**

round(expr) – округлення виразу **expr;**

Обчислення комплексних виразів.

Дійсну і уявну частини комплексного виразу $z=x+iy$ можна знайти за допомогою команд **Re(z)** і **Im(z).** Наприклад:

> z:=3+I*2:

> Re(z); Im(z);

$$3,$$

$$2$$

Якщо $z=x+iy$, то комплексно спряжений йому вираз $w=z^*=x-iy$ можна знайти за допомогою команди **conjugate(z)**. Продовження попереднього прикладу:

w:=conjugate(z);

$$w:=3-2\,I$$

Модуль і аргумент комплексного виразу z можна знайти за допомогою команди **polar(z)**, яку необхідно заздалегідь викликати із стандартної бібліотеки командою **readlib**. Наприклад:

> readlib(polar): polar(I);

$$\mathrm{polar}\left(1, \frac{1}{2}\pi\right)$$

У рядку виводу в дужках через кому вказані модуль числа і, рівний одиниці і його аргумент, рівний 1.

Якщо комплексний вираз дуже складний або містить параметри, то команди **Re(z)** і **Im(z)** не дають необхідного результату. Отримати дійсну і уявну частини комплексного виразу **z** можна, якщо використовувати команду перетворення комплексних виразів **evalc(z).** Наприклад:

> z:=ln(1-I*sqrt(3))^2;

$$z := \ln(1 - I\sqrt{3})^2$$

> evalc(Re(z)); evalc(Im(z));

$$\ln(2)^2 - \frac{1}{9}\pi^2$$

$$-\frac{2}{3}\ln(2)\pi$$

Приклад обчислень

1. Дано число *a=57/13*. Щоб знайти його цілу частину *x*, дробову частину *y* і переконатися, що *a=x+y*. Наберіть:

   > a:=57/13;

   > y:=frac(a);

$$\frac{5}{13}$$

   > x:=trunc(a);

$$4$$

   > x+y;

$$\frac{57}{13}$$

2. Приклади знаходження дійсної і уявної частини комплексного числа, та комплексно спряженого до нього числа w і перевірка того, що: *w+z=2Re(z).* У командному рядку наберіть:

   > z:=(2-3*I)/(1+4*I)+I^6:

   > Re(z); Im(z);

$$-\frac{27}{17}$$

$$-\frac{11}{17}$$

   > w:=conjugate(z);

$$w := -\frac{27}{17} + \frac{11}{17} I$$

   > z+w;

$$-\frac{54}{17}$$

3. Приклад знаходження модуля і аргумента комплексного числа *z* і обчислення z^4. Наберіть:

   > z:=-1-I*sqrt(3);

$$-1 - I\sqrt{3}$$

   > readlib(polar): polar(z);

$$\mathrm{polar}\left(2, -\frac{2}{3}\pi\right)$$

   Чому дорівнює модуль і аргумент цього числа?

   > evalc(z^4);

$$-8-8I\sqrt{3}$$

§3. Розв'язання рівнянь

Розв'язання звичайних рівнянь.

Для розв'язання рівнянь в Maple існує універсальна команда **solve(eq,x)**, де **eq** – рівняння, **x** – змінна, відносно якої рівняння треба розв'язати. В результаті виконання цієї команди в рядку виводу з'явиться вираз, який є розв'язанням даного рівняння. Наприклад:

> solve(a*x+b=c,x);

$$-\frac{b-c}{a}$$

Якщо рівняння має декілька розв'язків, які вам знадобляться для подальших розрахунків, то команді **solve** потрібно привласнити яке-небудь ім'я **name**. Звернення до якого-небудь *k-го* розв'язку даного рівняння проводиться вказівкою його імені з номером розв'язку в квадратних дужках: **name[*k*]**. Наприклад:

> x:=solve(x^2-a=0,x);

$$x := -\sqrt{a}, \sqrt{a}$$

> x[1];

$$-\sqrt{a}$$

> x[2];

$$\sqrt{a}$$

> x[1]+x[2];

$$0$$

Розв'язання системи рівнянь.

Системи рівнянь розв'язуються за допомогою аналогічної команди **solve({eq1,eq2.},{x1,x2.})**, лише тепер в параметрах команди потрібно вказувати у перших фігурних дужках через кому рівняння, а в других фігурних дужках перераховуються через кому змінні, відносно яких потрібно розв'язати систему. Якщо вам буде необхідно для подальших обчислень використовувати отримані розв'язки рівнянь, то команді **solve** потрібно привласнити яке-небудь ім'я **name**. Потім виконується привласнення команда **assign(name)**. Після цього над розв'язками можна буде проводити математичні операції. Наприклад:

> s:=solve({a*x-y=1,5*x+a*y=1},{x,y});

$$s := \left\{x = \frac{a+1}{a^2+5}, y = \frac{a-5}{a^2+5}\right\}$$

> assign(s); simplify(x-y);

$$\_\frac{a^2 y - a + 5y - 1}{a^2 + 5}$$

Чисельне розв'язання рівнянь.

Для чисельного розв'язання рівнянь, в тих випадках, коли трансцендентні рівняння не мають аналітичних розв'язків, використовується спеціальна команда **fsolve(eq,x)**, параметри якої такі ж, як і команди **solve**. Наприклад:

> x:=fsolve(cos(x)=x,x);

$$x:=.7390851332$$

Розв'язання рекурентних і функціональних рівнянь.

Команда **rsolve(eq,f)** дозволяє розв'язати рекурентне рівняння **eq** для цілої функції ***f***. Можна задати деяку початкову умову для функції **f(n)**, тоді отримаємо розв'язок даного рекурентного рівняння. Наприклад:

> eq:=2*f(n)=3*f(n-1)-f(n-2);

$$eq := 2f(n) = 3f(n-1) - f(n-2)$$

> rsolve({eq,f(1)=0,f(2)=1},f);

$$2 - 4 \cdot \left(\frac{1}{2}\right)^n$$

Універсальна команда solve дозволяє розв'язувати функціональні рівняння, наприклад:

> F:=solve(f(x)^2-3*f(x)+2*x,f);

$$F:= proc(x)\ RootOf(\_Z^2 - 3*\_Z + 2*x)\ end$$

В результаті виходить розв'язок в неявному вигляді. Проте Maple може працювати з такими розв'язками. Неявне розв'язання функціонального рівняння можна спробувати перетворити в яку-небудь елементарну функцію за допомогою команди **convert**. Продовжуючи наведений вище приклад, можна отримати розв'язок в явному вигляді:

> f:=convert(F(x),radical);

$$f := \frac{3}{2} + \frac{1}{2}\sqrt{9 - 8x}$$

Розв'язання тригонометричних рівнянь.

Команда **solve**, застосована для розв'язання тригонометричного рівняння, видає лише головні розв'язки, тобто розв'язки в інтервалі [0,2]. Для того, щоб отримати всі розв'язки, потрібно заздалегідь ввести додаткову команду _EnvAllSolutions:=true. Наприклад:

> _EnvAllSolutions:=true:

> solve(sin(x)=cos(x),x);

$$\frac{1}{4}\pi + \pi \_ZI\sim$$

У Maple символ _ZI~ позначає константу цілого типу, тому розв'язання даного рівняння в звичній формі має вигляд $\frac{1}{4}\pi$.

Розв'язання трансцендентних рівнянь.

При розв'язанні трансцендентних рівнянь для знаходження розв'язку в явному вигляді перед командою **solve** потрібно ввести додаткову команду _EnvExplicit:=true. Приклад розв'язання складної системи трансцендентних рівнянь і спрощення вигляду розв'язків:

> eq:={ 7*3^x-3*2^(z+y-x+2)=15, 2*3^(x+1)+

3*2^(z+y-x)=66, ln(x+y+z)-3*ln(x)-ln(y*z)=-ln(4) }:

> _EnvExplicit:=true:

> s:=solve(eq{x,y,z}):

> simplify(s[1]);simplify(s[2]);

{x=2, y=3, z=1}, {x=2, y=1, z=3}

Приклади розв'язування рівнянь та систем рівнянь

1. Для знаходження розв'язків системи рівнянь $\begin{cases} x^2 - y^2 = 1 \\ x^2 + xy = 2 \end{cases}$, наберіть:

   > eq:={x^2-y^2=1,x^2+x*y=2};
   > _EnvExplicit:=true:
   > s:=solve(eq{x,y});

$$s := \{x = \frac{2}{3}\sqrt{3}, y = \frac{1}{3}\sqrt{3}\}, \{x = -\frac{2}{3}\sqrt{3}, y = -\frac{1}{3}\sqrt{3}\}$$

Тепер знайдіть суму наступних виразів. Наберіть:

   > x1:=subs(s[1],x): y1:=subs(s[1],y):
   x2:=subs(s[2],x): y2:=subs(s[2],y):
   > x1+x2; y1+y2;

Чому дорівнюють розв'язки цих сум?

2. Для розв'язання рівняння $x^2 = \cos(x)$ чисельно. Наберіть:

   > x=fsolve(x^2=cos(x),x);

x=0.8241323123

3. Щоб знайти функцію *f(x)*, що задовільняє рівняння $f(x)^2 - 2f(x) = x$, наберіть:

   > F:=solve(f(x)^2-2*f(x)=x,f);

*F:= proc(x) RootOf(_Z^2(2*_Z(x))) end*

> f:=convert(F(x), radical);

$$f := 1 + \sqrt{1 + x}$$

Щоб знайти всі розв'язки рівняння $5\sin(x) + 12\cos(x) = 13$, наберіть:

> _EnvAllSolutions:=true:

> solve(5*sin(x)+12*cos(x)=13,x);

$$\arctan\left(\frac{5}{12}\right) + 2\pi\_Z \sim$$

§4. Розв'язання нерівностей

Розв'язання простих нерівностей.

Команда **solve** застосовується також для розв'язання нерівностей. Розв'язання нерівності видається у вигляді інтервалу зміних. В випадку, якщо розв'язком нерівності є пів вісь, то в полі виводу з'являється конструкція вигляду RealRange(–(, Open(a)), яка означає, що $x \in [a;\infty]$, а – деяке число. Слово Open означає, що інтервал з відкритою границею. Якщо цього слова немає, то відповідна границя інтервалу включена в безліч розв'язків. Наприклад:

> s:=solve(sqrt(x+3) <sqrt(x-1)+sqrt(x-2),x):

> convert(s,radical);

RealRange

Якщо ви хочете отримати розв'язок нерівності не у вигляді інтервалу $x \in [a;b]$, а у вигляді обмежень для шуканої змінної типу a<x, x< b, то змінну, відносно якої потрібно розв'язати нерівність, потрібно вказувати у фігурних дужках. Наприклад:

> solve(1-1/2*ln(x) >2,{x});

$$\{0 < x, x < e^{(-2)}\}$$

Розв'язання систем нерівностей.

За допомогою команди **solve** можна також розв'язати систему нерівностей. Наприклад:

> solve({x+y>=2,x-2*y<=1,x-y>=0,x-2*y>=1},{x,y});

$$\{x = 1 + 2y, \frac{1}{3} \le y\}$$

Приклади розв'язань нерівностей

1. Щоб розв'язати нерівність $0 < -x^4 + 13x^3 - 25x^2 - 129x + 270$, наберіть:

   > solve(13*x^3-25*x^2-x^4-129*x+270>0,x);

   RealRange(Open(-3), Open(2)), RealRange(Open(5), Open(9))

   Запишіть цей результат в аналітичному вигляді. Отримаєте розв'язок цієї нерівності у вигляді обмежень для шуканої змінної. Виконайте це самостійно.

2. Розв'яжіть нерівність . Наберіть:

   > solve(exp(2*x+3) <1,x);

   $$RealRange\left(-\infty, Open\left(-\frac{3}{2}\right)\right)$$

   Тепер отримаєте самостійно розв'язання цієї нерівності у вигляді обмежень для шуканої змінної.

Контрольні завдання.

Виконаєте всі контрольні завдання. Перед їх виконанням наберіть в текстовому режимі «Контрольні завдання». Результати виконання завдань покажіть викладачеві. Збережіть файл зі всіма виконаними завданнями на диск. Відповідайте на всі контрольні питання.

При виконанні контрольних завдань учневі необхідно підставити замість буквених параметрів індивідуальні анкетні характеристики:

a - число букв в прізвищі учня

b - число букв в повному імені учня

c - число букв в по батькові учня.

У звіті на титульному аркуші необхідно обов'язково вказати, які анкетні дані використовувалися при виконанні контрольних завдань (ім'я, по батькові, прізвище).

Завдання.

1. Дано комплексне число . Знайти його дійсну і уявні частини, модуль і аргумент.
2. Записати функцію у вигляді функціонального оператора і обчислите її значення при x=1, y=0 і при .
3. Записати функцію за допомогою оператора привласнення і обчислите її значення при x=d, y=1/d, використовуючи команду підстановки subs.
4. Знайти всі точні розв'язки системи в аналітичному вигляді.
5. Знайти всі розв'язання тригонометричного рівняння .
6. Знайти чисельні розв'язки рівняння.
7. Розв'яжіть нерівність.

Контрольні питання.

1. Як описати способи завдання функцій в Maple?
2. Які операції перетворення проводяться в Maple з дійсними виразами?
3. Для чого призначена команда evalf?
4. За допомогою яких команд можна знайти дійсну і уявну частини комплексного числа, а також його модуль і аргумент, і комплексно зв'язане йому число? Яку роль виконує команда evalc?
5. Для чого призначена команда solve?
6. Які команди використовуються для чисельного розв'язування рівнянь і для розв'язання рекурентних рівнянь?
7. Які додаткові команди потрібно ввести для того, щоб отримати точне розв'язання рівняння, всі розв'язання рівняння?
8. У яких форматах виводяться розв'язки нерівностей? Як відрізнити в рядку виводу замкнений інтервал від відкритого?

ПРАКТИЧНА РОБОТА 3.

ІІІ. Побудова кутових графіків та поверхонь

1. Двовимірні графіки.
2. Тривимірні графіки. Анімація.

§1. Двовимірні графіки

Команда plot та її параметри.

Для побудови графіків функції *y=f(x)* однієї змінної на площині *X0Y* використовується команда **plot**(*f(x), x=a..b, y=c..d, parameters*), де **parameters** – параметри управління зображенням. Якщо їх не вказувати, то будуть використані установки за умовчанням. Налаштування зображення також може здійснюватися з панелі інструментів.

Основні параметри команди **plot**:

1) **title=«text»,** де text-заголовок малюнка (текст можна залишати без лапок, якщо він містить лише латинські букви без пропусків).

2) **coords=polar –** установка полярних координат (за умовчанням встановлені декартові).

3) **axes –** установка типу координатних осей: axes=NORMAL – звичайні осі; axes=BOXED – графік в рамці з шкалою; axes=FRAME – осі з центром в лівому нижньому кутку малюнка; axes=NONE – без осей.

4) **scaling –** установка масштабу малюнка: scaling=CONSTRAINED – однаковий масштаб по осях; **scaling=UNCONSTRAINED** – графік масштабується по розмірах вікна.

5) **style=LINE(POINT) –** вивід лініями (або крапками).

6) **numpoints=n –** число обчислюваних точок графіка (за умовчанням n=49).

7) **color –** установка кольору лінії: англійська назва кольору, наприклад, **yellow –** жовтий і так далі

8) **xtickmarks=nx i ytickmarks=ny –** число міток по осі Ox і осі Oy, відповідно.

9) **thickness=n, де n=1,2,3. –** товщина лінії (за умовчанням n=1).

10) **linestyle=n –** тип лінії: безперервна, пунктирна і так далі (n=1 – безперервна, встановлено за умовчанням).

11) **symbol=s –** тип символу, яким позначають крапки: **BOX, CROSS, CIRCLE, POINT, DIAMOND.**

12) **font=[f,style,size] –** установка типу шрифту для виведення тексту: f задає назву шрифтів: **TIMES, COURIER, HELVETICA, SYMBOL; style** задає стиль шрифту: **BOLD, ITALIC, UNDERLINE; size –** розмір шрифту в **pt.**

13) **labels=[tx,ty]** – написи по осях координат: **tx –** по осі **Ox i ty –** по осі **Oy.**

14) **discont=true –** вказівка для побудови безкінечних розривів.

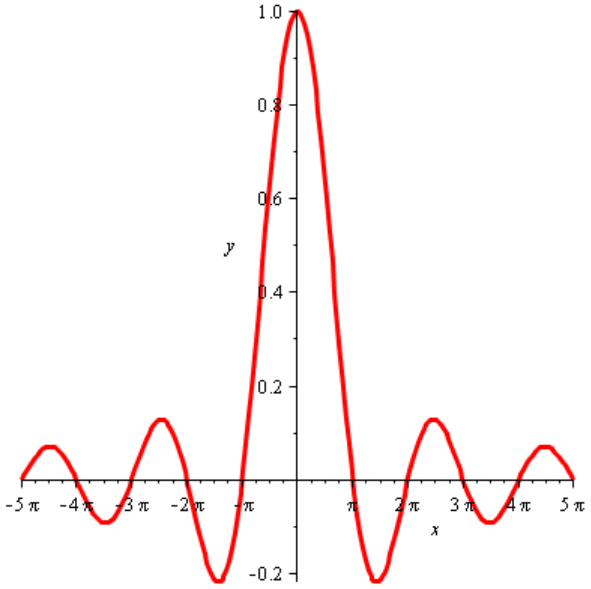
За допомогою команди **plot** можна будувати окрім графіків функцій *y=f(x),* заданою явно, також криві, задані параметрично *y=y(t), x=x(t),* якщо записати команду **plot**(*[y=y(t), x=x(t), t=a..b], parameters*).

Приклади побудови графіків

1. Приклад побудови графіка $\frac{\sin(x)}{x}$ жирною лінією в інтервалі від -4 до 4. Наберіть:

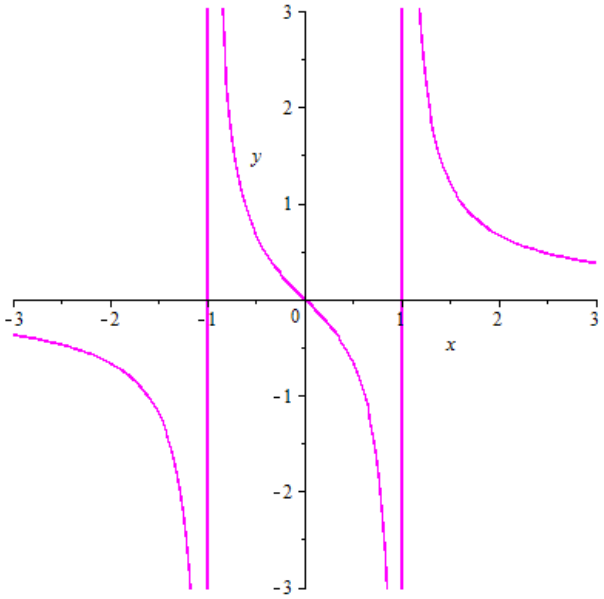
> plot(sin(x) /x, x=-5*Pi..5*Pi, labels=[x,y] labelfont=[TIMES,ITALIC,12], thickness=3);

$$plot\left(\frac{\sin(x)}{x},\ x=-5\cdot Pi..5\cdot Pi,\ labels=[x,\ y],\ thickness=3\right);$$



1. Приклад побудуви графіка розривної функції $\frac{x}{x^2-1}$, наберіть:

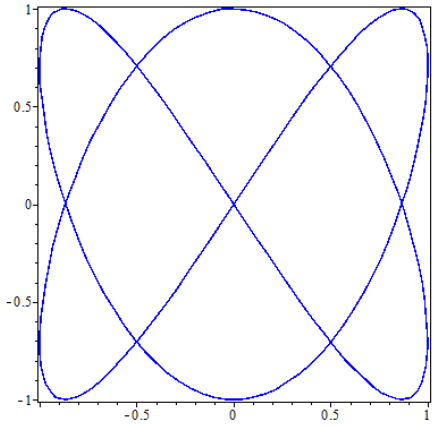
> plot(x/(x^2-1),x=-3..3,y=-3..3,color=magenta);



Зауваження: на малюнку автоматично з'являються вертикальні асимптоти.

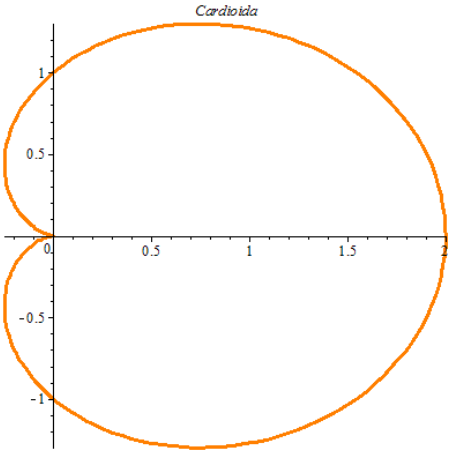
2. Для побудови параметрично заданої кривої $\sin(2t), \cos(3t), t = 0..2\pi$ в рамці, наберіть:

> plot([sin(2*t),cos(3*t),t=0..2*Pi], axes=BOXED, color=blue);



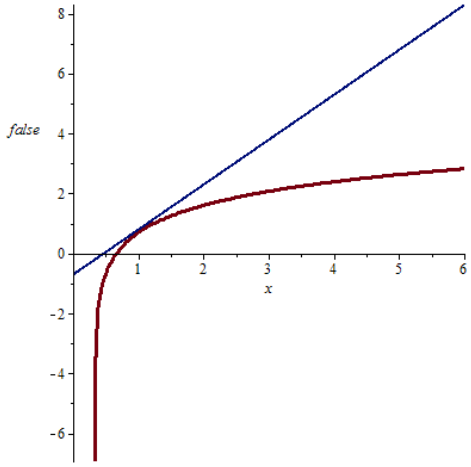
Приклад побудови заданої в полярних координатах кардіоїди $1+\cos(x), x=0..2\pi$ з підписом, наберіть:

> plot(1+cos(x), x=0..2*Pi, title=Cardioida, coords=polar, color=coral, thickness=3);



Приклад побудови на одному малюнку графік функції $\ln(3x-1)$ і дотичну до нього $3x/2-\ln(2)$, для $x=0..6$, наберіть:

> plot([ln(3*x-1), 3*x/2-ln(2)], x=0..6, color=[violet,gold] linestyle=[1,2], thickness=[3,2]);



Побудова графіка функції, заданої неявно.

Для побудови графіка неявної функції використовується команда **implicitplot** з графічного пакету:

plots: implicitplot(F(x,y)=0, x=x1..x2, y=y1..y2).

Виведення текстових коментарів на малюнок.

У пакеті **plots** є команда **textplot** для виведення текстових коментарів на малюнок: **textplot**([xo,yo,'text'], options), де xo, yo – координати точки, з якою починається виведення тексту 'text'.

Виведення декількох графічних об'єктів на один малюнок.

Часто буває необхідно поєднати на одному малюнку декілька графічних об'єктів, отриманих за допомогою різних команд, наприклад, додати графіку, намальованому командою **plot**, текстові написи, отримані командою textplot. Для цього результат дії команди привласнюється деякій змінній:

> **p:=plot(.): t:=textplot(.):**

При цьому на екрані елемент не появляється. Для виведення графічних зображень необхідно виконати команду з пакету plots:

> **with(plots): display([p,t], options).**

Побудова двовимірної множини, заданої нерівностями.

Якщо необхідно побудувати двовимірну множину, задану системою нерівностей, то для цього можна використовувати команду **inequal** з пакету **plots**. У команді **inequals({f1(x,y) >c1.,fn(x,y)>cn}, x=x1.x2, y=y1..y2, options)** у фігурних дужках вказується система нерівностей, що визначають множину, потім розміри координатних осей і параметри. Параметри регулюють кольори відкритих і закритих границь, кольори зовнішньої і внутрішньої меж, а також товщину ліній границь:

- **optionsfeasible=(color=red) –** установка кольору внутрішньої межі;
- **optionsexcluded=(color=yellow) –** установка кольору зовнішньої межі;
- **optionsopen(color=blue, thickness=2) –** установка кольору і товщини лінії відкритої границі;
- **optionsclosed(color=green,thickness=3) –** установка кольору і товщини лінії закритої границі.

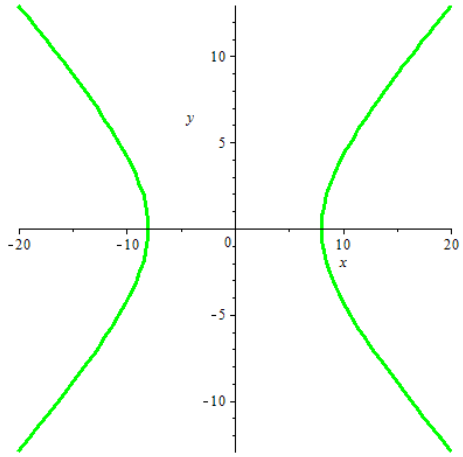
Приклади побудов

1. Приклад побудови неявної функції (гіперболи)

$\frac{1}{4}x^2 - \frac{1}{2}y^2 = 16, x = -20..20, y = -16..16$

> **with(plots):**

> **implicitplot(x^2/4-y^2/2=16, x=-20..20, y=-16..16, color=green, thickness=3);**



2. Приклад побудови на одному малюнку еліпса $\frac{1}{16}x^2 + \frac{1}{4}y^2 = 1$ та вписану у нього астроїду $4\cos(t)^3, 2\sin(t)^3, t = 0..2\pi$. Для цього наберіть наступні рядки:

> **with(plots):**

> **eq:=x^2/16+y^2/4=1:**

> **el:=implicitplot(eq, x=-4..4, y=-2..2, scaling=CONSTRAINED, color=green, thickness=3):**

> **as:=plot([4*cos(t)^3,2*sin(t)^3, t=0..2*Pi], color=blue, scaling=CONSTRAINED, thickness=2):**

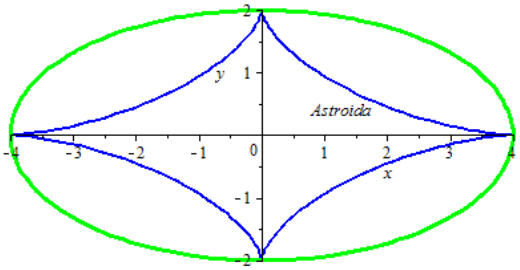
> **eq1:=convert(eq,string):**

> **t1:=textplot([1.5,2.5,eq1], font=[TIMES ITALIC, 10], align=RIGHT):**

> **t2:=textplot([0.2,2.5, Ellips], font=[TIMES BOLD,10], align=RIGHT):**

> **t3:=textplot([1.8,0.4,Astroida], font=[TIMES BOLD,10], align=LEFT):**

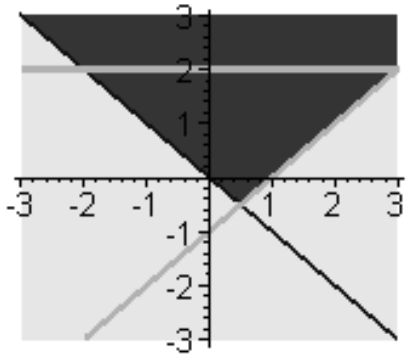
> **display([as,el,t1,t2,t3]);**



3. Приклад побудови фігуру множини, обмежену заданими лініями:

> **with(plots):**

> **inequal({x+y>0, x-y<=1, y=2}, x=-3..3, y=-3..3, optionsfeasible=(color=red), optionsopen=(color=blue,thickness=2), optionsclosed=(color=green, thickness=3), optionsexcluded=(color=yellow));**



§2. Тривимірні графіки. Анімація

Графік поверзні, заданої явною функцією.

Графік функції *z=f(x,y)* можна намалювати, використовуючи команду **plot3d(f(x,y)**, **x=x1.x2, y=y1.y2, options)**. Параметри цієї команди частково збігаються з параметрами команди **plot**. До часто використовуваних параметрів команди **plot3d** відноситься **light=[angl1, angl2, c1, c2, c3]** – підсвічування поверхні, що задається джерелом світла з крапки із сферичними координатами (angl1, angl2). Колір визначається долями червоного (c1), зеленого (c2) і синього (c3) кольорів, які знаходяться в інтервалі [0,1]. Параметр **style=opt** задає стиль малюнка: **POINT** –точки, **LINE** – лінії, **HIDDEN** – сітка з видаленням невидимих ліній, **PATCH** – заповнювач (встановлений за умовчанням), **WIREFRAME** – сітка з виведенням

невидимих ліній, **CONTOUR** – лінії рівня, **PATCHCONTOUR** – заповнювач і лінії рівня. Параметр **shading=opt** задає функцію інтенсивності заповнювача, його значення рівне **xyz** – за умовчанням, **NONE** – без розфарбовування.

Зображення поверхні, заданої параметрично.

Якщо потрібно побудувати поверхню, задану параметрично: x=x(u,v), y=y(u,v), z=z(u,v), то ці функції перераховуються в квадратних дужках в команді: **plot3d([x(u,v), в(u,v), z(u,v)], u=u1..u2, v=v1..v2).**

Зображення поверхні, заданої рівнянням *F(x,y,z)=0*.

Тривимірна поверхня, задана рівнянням *F(x,y,z)=0*, будується за допомогою команди пакету **plot: implicitplot3d(F(x,y,z)=c, x=x1..x2, y=y1..y2, z=z1..z2)**, де вказується рівняння поверхні і розміри малюнка по координатних осях.

Зображення просторових кривих.

У пакеті **plot** є команда **spacecurve** для побудови просторової кривої, заданої параметрично:

> **spacecurve([x(t),y(t),z(t)],t=t1..t2)**

де змінна t змінюється від t1 до t2.

Анімація.

*Maple дозволяє виводити на екран рухомі зображення за допомогою команд **animate** (двовимірні) і **animate3d** (тривимірні) з пакету **plot**. Серед параметрів команди **animate3d** є **frames** – число кадрів анімації (за умовчанням **frames=8**).*

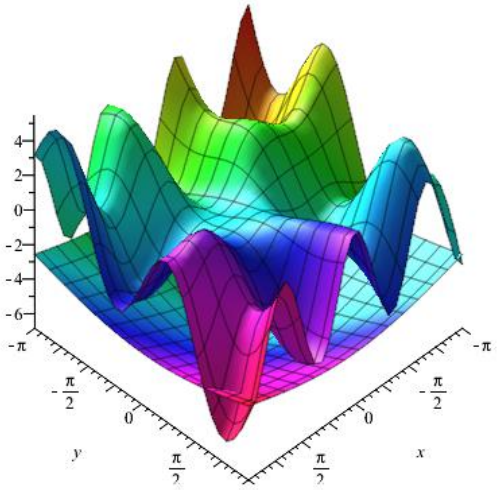
Тривимірні зображення зручніше набудовувати не за допомогою опцій команди **plot3d**, а використовуючи контекстне меню програми. Для цього потрібно клацнути правою кнопкою миші по зображенню. Тоді з’явиться контекстне меню налаштування зображення. Команди цього меню дозволяють змінювати колір зображення, режими підсвічування, встановлювати потрібного типу осей, тип ліній і керувати рухомим зображенням.

Контекстне меню налаштування зображення:

Приклади побудов поверхонь

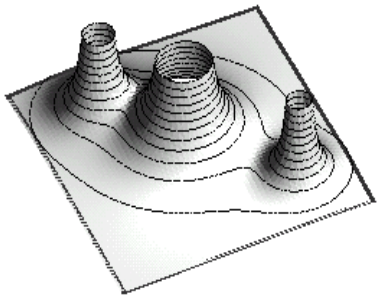
1. Приклад побудови двох поверхонь та встановлення змінного кольору поверхонь:

> **plot3d({x*sin(2*y)+y*cos(3*x), sqrt(x^2+y^2)-7}, x=-Pi..Pi, y=-Pi..Pi, grid=[30,30], axes=FRAMED, color=x+y);**



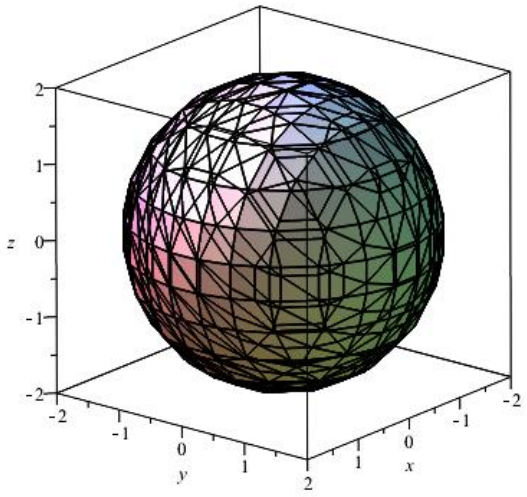
2. Приклад побудови поверхні разом з лініями рівня, наберіть:

> **plot3d(1/(x^2+y^2) +0.2/((x+1.2)^2+(y-1.5)^2)+ 0.3/((x-0.9)^2+(y+1.1)^2), x=-2..2, y=-2..2.5, view=[-2..2, -2..2.5, 0..6], grid=[60,60], shading=NONE, light=[100,30,1,1,1], axes=NONE, orientation=[65,20], style=PATCHCONTOUR);**



3. Приклад побудуви кулі:

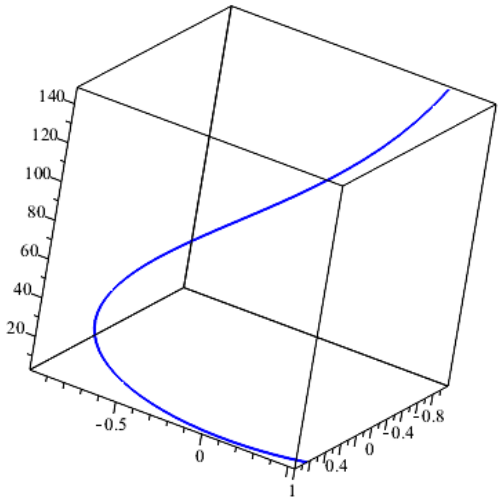
> **with(plots): implicitplot3d(x^2+y^2+z^2=4, x=-2..2, y=-2..2, z=-2..2, scaling=CONSTRAINED);**



4. Приклад побудуви просторової кривої:

> **with(plots):**

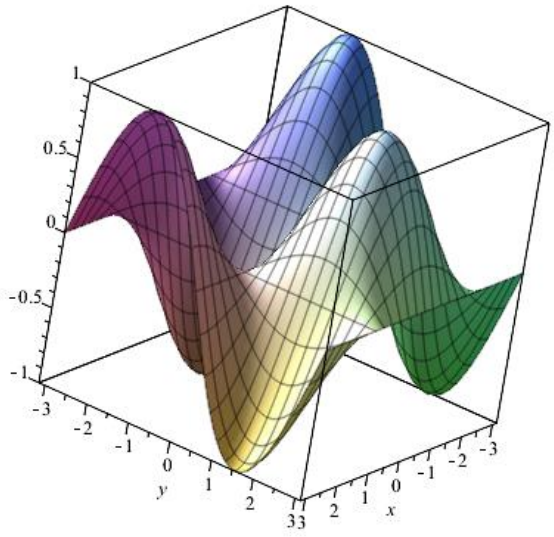
> **spacecurve([sin(t),cos(t),exp(t)], t=1..5, color=blue, thickness=2, axes=BOXED);**

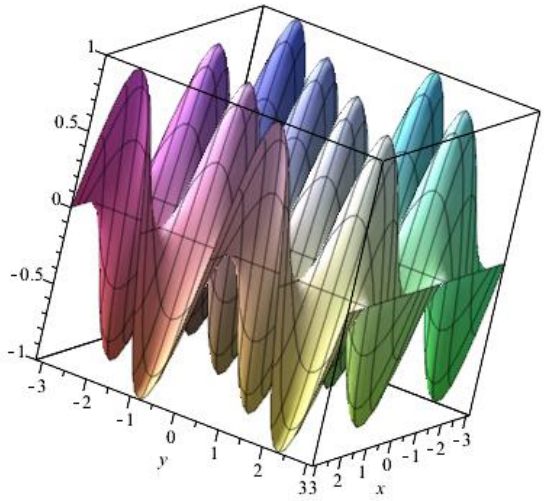


5. Приклад зображення рухомого об’єкту. Спочатку наберіть в командному рядку:

> **with(plots):**

> **animate3d(cos(t*x)*sin(t*y), x=-Pi..Pi, y=-Pi..Pi, t=1..2);**





Клацніть по зображенню правою кнопкою миші, що з'явилося. У контекстному меню, що з'явилося, виконаєте команду **Animation(Continuous)**. Потім знову викличте контекстне меню і виконаєте команду **Animation(Play**). Для того, щоб зупинити рух, виконаєте команду **Animation(Stop**). Потім оберніть малюнок під іншим кутом і зробіть його знов рухомим.

6. Виконаєте всі контрольні завдання. Результати виконання завдань покажіть вчителеві. Збережіть файл зі всіма виконаними завданнями на диск. Відповідайте на всі контрольні питання.

Контрольні завдання.

При виконанні контрольних завдань необхідно підставити замість буквених параметрів індивідуальні анкетні характеристики:

a - число букв в повному імені учня

b - число букв в по батькові учня

c - число букв в прізвищі учня.

У звіті на титульному аркуші необхідно обов'язково вказати, які анкетні дані використовувалися при виконанні контрольних завдань (ім'я, по батькові, прізвище).

Завдання.

1. Побудувати трикутник, сторони якого утворюють задані прямі (відповідно свого варіанта).

2. Побудувати на одному малюнку графіки функції і її асимптоти. Встановити наступні параметри: колір основної лінії – блакитний, асимптот – червоний (встановлений за умовчанням, тому його можна не змінювати); товщина основної лінії – 3, асимптоти – звичайною; масштаб по координатних осях – однаковий. Зробити підписи: яка функція відноситься до якої лінії. Вказівка:

використовувати для перетворення в текст формул команду convert, а для побудови графіків і написів команди textplot і display з пакету plots (див. Завдання 1.2, п.2)

3. Побудувати графік неявної функції (відповідно свого варіанта).
4. Побудувати графік функції (відповідно свого варіанта).
5. Побудувати графік поверхні і визначити її вигляд .
6. Побудувати в одній системі координат кілька поверхонь і визначити їх вигляд. Користуючись можливостями Maple, отримати оптимальне зображення.
7. Побудувати графік функції на множені визначення.

Вказівка. Результат виконання кожного завдання необхідно оптимізувати, використовуючи можливості Maple, зробити всі необхідні підписи (заголовки, осі і т. д.).

Контрольні запитання.

1. За допомогою яких команд будуються графіки функцій на площині і в просторі? Які аргументи мають ці команди?
2. Як називається пакет додаткових графічних команд?
3. За допомогою якої команди можна побудувати графік неявної функції? Опишіть її параметри.
4. Для чого призначена команда display?
5. Яка команда дозволяє побудувати двовимірну множену, задану системою нерівностей?
6. За допомогою якої команди можна побудувати просторову криву?
7. Які можливості надають команди animate і animate3d?

ПРАКТИЧНА РОБОТА 4.

IV. Вступ до математичного аналізу та диференціальне числення функцій.

1. Обчислення границь.
2. Диференціювання.
3. Дослідження функції.

§1. Обчислення границь

У Maple для деяких математичних операцій існує по дві команди: одна прямого, а інша – відкладеного виконання. Імена команд складаються з однакових букв за винятком першої: команди прямого виконання починаються з рядкової букви, а команди відкладеного виконання – із

заголовною. Після звернення до команди відкладеної дії математичні операції (інтеграл, границя, похідна і так далі) виводяться на екран у вигляді стандартного аналітичного запису цієї операції. Обчислення в цьому випадку відразу не проводиться. Команда прямого виконання видає результат відразу.

Для обчислення границь є дві команди:

1) прямого виконання – **limit(expr,x=a,par)**, де **expr** – вираз, границю якого потрібно знайти, **a** – значення точки, для якої обчислюється границя, **par** – необов'язковий параметр для пошуку односторонних границь (left – зліва, right – справа) або вказівка типу змінній (real – дійсна, complex – комплексна).
2) відкладеного виконання – **Limit(expr,x=a,par)**, де параметри команди такі ж, як і у попередньому випадку. Приклад дій цих команд:

> **Limit(sin(2*x) /x,x=0);**

$$\lim_{x\to 0}\frac{\sin(2x)}{x}$$

> **limit(sin(2*x) /x,x=0);**

2

За допомогою цих двох команд прийнято подавати математичні записи в стандартному аналітичному вигляді, наприклад:

> **Limit(x*(Pi/2+arctan(x)),x=-infinity)=**

limit(x*(Pi/2+arctan(x)), x=-infinity);

$$\lim_{x\to\infty} x\left(\frac{\pi}{2}+\arctan(x)\right)=-1$$

Односторонні границі обчислюються з вказівкою параметрів: left – для знаходження границі зліва і righ – справа. Наприклад:

> **Limit(1/(1+exp(1/x)),x=0,left)=**

limit(1/(1+exp(1/x)),x=0,left);

$$\lim_{x\to 0-}\frac{1}{1+e^{\frac{1}{x}}}=1$$

> **Limit(1/(1+exp(1/x)),x=0,right)=**

limit(1/(1+exp(1/x)), x=0,right);

$$\lim_{x\to 0+}\frac{1}{1+e^{\frac{1}{x}}}=0$$

Приклади обчислення границь

1. Наберіть:

   > **Limit((1-x)*tan(Pi*x/2),x=1)=**
   limit((1-x)*tan(Pi*x/2),x=1);

$$\lim_{x\to 1}(1-x)\tan\left(\frac{1}{2}\pi x\right)=2\frac{1}{\pi}$$

2. Для знаходження односторонньї границі. Наберіть:

   > **Limit(arctan(1/(1-x)),x=1,left)=**
   limit(arctan(1/(1-x)), x=1, left);

$$\lim_{x\to 1-}\arctan\left(\frac{1}{1-x}\right)=\pi\frac{1}{2}$$

   > **Limit(arctan(1/(1-x)),x=1,right)=**
   limit(arctan(1/(1-x)),x=1, right);

$$\lim_{x\to 1+}\arctan\left(\frac{1}{1-x}\right)=-\pi\frac{1}{2}\,.$$

§2. Диференціювання

Обчислення похідних.

Для обчислення похідних в Maple є дві команди:

1) прямого виконання – diff(f,x), де f – функція, яку потрібно продиференціювати, x – ім'я змінної, по якій проводиться диференціювання.

2) відкладеного виконання – Diff(f,x), де параметри команди такі ж, як і в попередній. Дія цієї команди зводиться до аналітичного запису похідної у вигляді . Після виконання диференціювання, отриманий вираз бажано спростити. Для цього можна використовувати команди simplify, factor, expand, залежно від того, в якому вигляді вам потрібний результат.

Приклад:

> **Diff(sin(x^2),x)=diff(sin(x^2),x);**

$$\frac{\partial}{\partial x}\sin(x^2)=2\cos(x^2)x$$

Для обчислення похідних старших порядків потрібно вказати в параметрах x$n, де n – порядок похідної; наприклад:

> **Diff(cos(2*x)^2,x$4)=diff(cos(2*x)^2,x$4);**

$$\frac{\partial^4}{\partial x^4}\cos(2x)^2=-128\sin(2x)^2+128\cos(2x)^2$$

Отриманий вираз можна спростити наприклад так:

> **simplify(%);**

$$\frac{\partial^4}{\partial x^4}\cos(2x)^2 = 256\cos(2x)^2 - 128$$

або

> **combine(%);**

$$\frac{\partial^4}{\partial x^4}\left(\frac{1}{2}\cos(4x)+\frac{1}{2}\right)^2 = 128\cos(4x)$$

Диференціальний оператор.

Для визначення диференціального оператора використовується команда *D(f)*, де *f*-функция. Наприклад:

> **D(sin);**

cos

Обчислення похідної в точці:

> **D(sin)(Pi):eval(%);**

-1

Оператор диференціювання застосовується до функціональних операторів

> **f:=x-> ln(x^2)+exp(3*x):**

> **D(f);**

$$x \to 2\frac{1}{x} + 3e^{(3x)}$$

Приклади обчислення похідної

1. Для обчислення похідної, наберіть:
   > **Diff(sin(2*x)^3-cos(2*x)^3,x)=**
   diff(sin(2*x)^3-cos(2*x)^3,x);

$$\frac{\partial}{\partial x}(\sin(2x)^3 - \cos(2x)^3) = 6\sin(2x)^2\cos(2x) + 6\cos(2x)^2\sin(2x)$$

2. Наберіть:
   > **Diff(exp(x)*(x^2-1),x$24)=**
   diff(exp(x)*(x^2-1),x$24):
   > **collect(%,exp(x));**

$$\frac{\partial^{24}}{\partial x^{24}}e^x(x^2-1) = e^x(x^2+48x+551)$$

3. Обчислити другу похідну функції в точках x=π, x= π/2.
   > **y:=sin(x)^2/(2+sin(x)): d2:=diff(y,x$2):**

> **x:=Pi; d2y(x)=d2;**

$$\pi$$
$$d2y(\pi) = 0$$

> **x:=Pi/2;d2y(x)=d2;**

$$\frac{1}{2}\pi$$
$$d2y\left(\frac{1}{2}\pi\right) = 0$$

§3. Дослідження функції

Дослідження функції необхідно починати із знаходження її множини визначення, але, на жаль, це операція, що важко автоматизується. Тому при розгляді цього питання доводиться розв'язувати нерівності (див. тему II). Проте, відповісти на питання, чи визначена функція на всій числовій осі, чи ні, можна дослідивши її на неперервність.

Неперервність функції і точки розриву.

Перевірити неперервність функції f(x) на заданому проміжку [x1,x2] можна за допомогою команди iscont(f,x=x1..x2). Якщо функція f неперервна на цьому інтервалі, то в полі виводу з'явиться відповідь true – (істина); якщо функція f не є неперервною на цьому інтервалі, то в полі виводу з'явиться відповідь false – (брехня). Зокрема, якщо задати інтервал x=-infinity..+infinity, то функція f перевірятиметься на всій числовій осі. В цьому випадку, якщо буде отримана відповідь true, то можна сказати, що функція визначена і неперервна на всій числовій осі. Інакше потрібно шукати точки розриву. Це можна зробити двома способами:

1) за допомогою команди discont(f,x), де f – функція, досліджувана на неперервність, x – змінна. Ця команда придатна для знаходження точки розриву першого і другого роду.
2) за допомогою команди singular(f,x), де f – функція, x – змінна. Ця команда годиться для знаходження точок розриву другого роду як для дійсних значень змінної, так і для комплексних.

Перед використанням цих команд їх потрібно обов'язково завантажити із стандартної бібліотеки readlib(name), де name – ім'я якоїсь з вказаних вище команд.

Обидві ці команди видають результати у вигляді перерахування точок розриву у фігурних дужках. Тип такого запису називається set. Для того, щоб надалі можна було використовувати набутих значень точок розриву, виходить з типу set за допомогою команди convert перевести їх в звичайного числового типу.

Знаходження точок розриву

1. Для дослідження неперервності функції $e^{\frac{1}{\frac{1}{2}\pi+3}}$, наберіть:

> **readlib(iscont): readlib(discont):**

> **iscont(exp(1/(x+3)),x=-infinity..+infinity);**

false

Це означає, що функція не є неперервною. Тому, щоб знайти точки розриву, наберіть:

> **discont(exp(1/(x+3)),x);**

{-3}

Відповідь можна вивести в текстовому режимі в новому рядку:

«Точка розриву x=(-3).»

2. Для знаходження точок розриву функції $\tan\left(\frac{1}{2}\,\frac{\pi}{2-x}\right)$, наберіть:

> **readlib(singular):**

> **iscont(tan(x/(2-x)),x=-infinity..infinity);**

False

Функція не є неперервною на (-∞;+∞):

> **singular(tan(x/(2-x)),x);**

$$\left\{x=\frac{1}{2}\,\pi\,(1+2\_ZI\sim)\,(-2+x)\right\}$$

Тут _Zl – цілі числа. Відповідь наберіть в текстовому режимі в новому рядку: «Точки розриву: $\left\{x=\frac{1}{2}\,\pi\,(1+2\_ZI\sim)\,(-2+x)\right\}$ ».

Екстремуми. Найбільше і найменше значення функції.

У Maple для дослідження функції на глобальний екстремум є команда extrema(f,{cond},x,’s’), де *f* – функція, екстремуми якої шукаються. У фігурних дужках {cond} вказуються обмеження для змінної, x – ім’я змінної, по якій шукається екстремум, в апострофах ‘s’ – вказується ім’я змінної, якою буде привласнена координата точки екстремуму. Якщо залишити порожніми фігурні дужки {}, то пошук екстремумів проводитиметься на всій числовій осі. Результат дії цієї команди належить до типу set. Приклад:

> **extrema(arctan(x) -ln(1+x^2) /2,{},x,’x0’);x0;**

$$\left\{\frac{\pi}{4}-\frac{1}{2}\ln(2)\right\}$$

{{x=1}}

У першому рядку виводу наводиться екстремум функції, а в другому рядку виводу – точка цього екстремуму.

На жаль, ця команда не може дати відповідь на питання, яка з точок екстремуму є максимум, а яка – мінімум. Для знаходження максимуму функції

f(x) по змінній x на інтервалі використовується команда maximize(f,x,x=x1..x2), а для знаходження мінімуму функції f(x) по змінній x на інтервалі використовується команда minimize(f, x, x=x1..x2). Якщо після змінної вказати ‘infinity’ або інтервал

x=-infinity..+infinity, то команди maximize і minimize шукатимуть, відповідно, максимуми і мінімуми на всій числовій осі як в дійсних числах, так і комплексних. Якщо такі параметри не вказувати, то пошук максимумів і мінімумів проводитиметься лише на множені дійсних чисел.

Приклад:

> **maximize(exp(-x^2),x);**

$$1$$

Недолік цих команд в тому, що вони видають лише значення функції в точках максимуму і мінімуму, відповідно.

У версії пакету аналітичних обчислень Maple 16 цей недолік команд maximize і minimize усунений. Координати точок максимуму або мінімуму можна отримати, якщо в параметрах цих команд після змінної записати через кому нову опцію location. В результаті в рядку виводу після самого максимуму (мінімуму) функції будуть у фігурних дужках вказані координати точок максимуму (мінімуму). Наприклад:

> **minimize(x^4-x^2, x, location);**

$$-\frac{1}{4}, \left\{\left[\left\{x=-\frac{1}{2}\sqrt{2}\right\}, -\frac{1}{4}\right], \left[\left\{x=\frac{1}{2}\sqrt{2}\right\}, -\frac{1}{4}\right]\right\}$$

У рядку виводу вийшли координати мінімумів і значення функції в цих точках.

Приклади знаходження max і min функцій

1. Приклад знаходження max і min функції

$$\frac{1}{2}\left(x^2-\frac{1}{2}\right)\arcsin(x)+\frac{1}{4}x\sqrt{-x^2+1}-\frac{1}{12}\pi x^2:$$

> **y:=(x^2-1/2)*arcsin(x)/2+x*sqrt(1-x^2) /4-Pi*x^2/12:**

> **extrema(y,{},x,’s’);s;**

$$\left\{0,-\frac{1}{24}\pi+\frac{1}{16}\sqrt{3}\right\}$$

$$\left\{\{x=0\},\{x=\frac{1}{2}\}\right\}$$

Після виконання цих команд знайдені екстремуми функції та відповідні їм точки екстремумів. Таким чином, знайдені екстремуми в точках (0,0) і (1/2, $-\frac{1}{24}\pi+\frac{1}{16}\sqrt{3}$). Залишилося з’ясувати, яка з них є максимумом, а яка – мінімумом. Для цього застосуємо команди maximize і minimize.

> **ymin:= minimize(y,x=0..1/2);**

$$ymin := -\frac{1}{24}\pi + \frac{1}{32}\sqrt{3}\sqrt{4}$$

> **ymax:= maximize(y,x=0..1/2);**

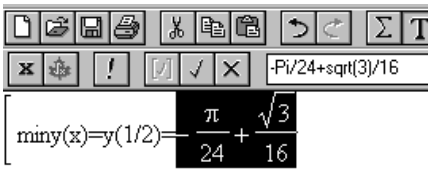
$$ymax := 0$$

Відповідь можна наберати в текстовому режимі в новому рядку:

«Екстремуми: $ymin := -\frac{1}{24}\pi + \frac{1}{32}\sqrt{3}\sqrt{4}$, $ymax := 0$ »

Для набору математичних символів і грецьких букв в текстовому режимі потрібно натискувати кнопку із значком суми на Панелі інструментів. У рядку введення формул, що з'явиться, нижче за Панель інструментів потрібно набирати звичайні команди Maple, після чого натискувати Enter. Наприклад, для відображення формули потрібно набрати в рядку введення формул sqrt(3). Для повернення в текстовий режим потрібно натискувати на кнопку з буквою «T». Тому порядок набору другої формули відповідає такий:

- знаходячись в текстовому режимі, набрати: miny(x)=y(1/2)= ;
- натискувати на кнопку «Math»;
- у рядку введення формул набрати: -Pi/24+sqrt(3)/16
- натискувати Enter;



- повернутися в текстовий режим.

2. Для знаходження найбільшого і найменшого значення функції $x^2\ln(x)$ на проміжку [1;2], наберіть:

> **f:=x^2*ln(x): maximize(f,x=1..2);**

$$4\ln(2)$$

> **minimize(f,x=1..2);**

$$0$$

Відповідь наберіть в текстовому режимі в новому рядку:

«Найбільше значення: min f(x)= $4\ln(2)$, найменше значення min f(x)=0».

Контрольні завдання.

При виконанні контрольних завдань учневі необхідно підставити замість буквених параметрів індивідуальні анкетні характеристики:

a - число букв в повному імені учня

b - число букв в по батькові учня

c - число букв в прізвищі учня.

У звіті на титульному аркуші необхідно обов'язково вказати, які анкетні дані використовувалися при виконанні контрольних завдань (ім'я, по батькові, прізвище).

Завдання.

1. Обчисліть границі відповідно до варіанта.
2. Знайти границі функції.
4. Знайти всі часткові похідні 2 – ого порядку функції.
5. Знайти точки розриву функції.
6. Знайти екстремуми функції.

ПРАКТИЧНА РОБОТА 5.

V. Інтегральне числення функції однієї і багатьох змінних.

1. Інтегрування.
2. Інтегральне числення функцій багатьох змінних.

§1 Інтегрування функції однієї змінної.

Аналітичне і чисельне інтегрування.

Невизначений інтеграл обчислюється за допомогою 2-х команд:

1) прямого виконання – **int(f, x)**, де **f** – підінтегральна функція, **x** – змінна інтегрування;
2) відкладеного виконання – **Int(f, x)** – де параметри команди такі ж, як і в команді прямого виконання **int.** Команда **Int** видає на екран інтеграл в аналітичному вигляді математичної формули.

Для обчислення визначеного інтеграла в командах **int** і **Int** вказуються межі інтегрування, наприклад:

```
> Int((1+cos(x))^2, x=0..Pi)=int((1+cos(x))^2,
x=0..Pi);
```

$$\int_0^{\pi} (1+\cos(x))^2 dx = \frac{3}{2}\pi$$

Якщо в команді інтегрування додати опцію **continuous: int(f, x, continuous)**, то Maple ігноруватиме будь-які можливі розриви підінтегральної функції в діапазоні інтегрування. Це дозволяє обчислювати невласні інтеграли від необмежених функцій. Невласні інтеграли з безконечними межами інтегрування обчислюються, якщо в параметрах команди **int** вказувати, наприклад, **x=0..+infinity**.

Чисельна інтегрування виконується командою **evalf(int(f, x=x1..x2), e)**, де **e** – точність обчислень (число знаків після коми).

Основні методи інтегрування.

У Maple є пакет **student**, призначений для навчання математиці. Він містить набір підпрограм, призначених для виконання розрахунків крок за кроком, так, щоб була зрозуміла послідовність дій, що веде до результату. До таких команд відносяться інтегрування по частинах **intparts** і заміна змінною **changevar**.

Формула інтегрування по частинах:

$$\int u(x)v'(x)dx = u(x)v(x) - \int u'(x)v(x)dx$$

Якщо позначити підінтегральну функцію **f=u(x) v'(x)**, то параметри команди інтегрування по частинах такі: **intparts(Int(f, x), u)**, де **u** – саме та функція **u(x)**, похідну від якої належить обчислити за формулою інтегрування по частинах. Якщо в інтегралі потрібно зробити заміну змінних **x=g(t)** або **t=h(x)**, то параметри команди заміни змінних такі: **changevar(h(x)=t, Int(f, x), t)**, де **t** нова змінна.

Обидві команди **intparts** і **changevar** не обчислюють остаточно інтеграл, а лише проводять проміжне обчислення. Для того, щоб отримати остаточну відповідь, слід, після виконання цих команд ввести команду **value(%)**; де **%** – позначають попередній рядок.

Напочатку використання описаних тут команд слід обов'язково завантажити пакет **student** командою **with(student)**.

Приклади обчислення інтегралів

1. Щоб знайти невизначений інтеграл $\int \cos(x)\cos(2x)\cos(3x)\,dx$, наберіть:

```
> Int(cos(x)*cos(2*x)*cos(3*x),x)=
  int(cos(x)*cos(2*x)*cos(3*x), x);
```

$$\int \cos(x)\cos(2x)\cos(3x)dx = \frac{1}{8}\sin(2x) + \frac{1}{16}\sin(4x) + \frac{1}{24}\sin(6x) + \frac{1}{4}x$$

```
> Int((3*x^4+4)/(x^2*(x^2+1)^3),x)=
  int((3*x^4+4)/(x^2*(x^2+1)^3),x);
```

$$\int \frac{3x^4+4}{x^2(x^2+1)^3}dx = -4\frac{1}{x} - \frac{57}{8}\arctan(x) - \frac{25}{8}\frac{x}{x^2+1} - \frac{7}{4}\frac{x}{(x^2+1)^2}$$

2. Щоб знайти інтеграл $\int_0^{\frac{1}{2}\pi} \frac{\sin(x)\cos(x)}{a^2\cos(x)^2 + b^2\sin(x)^2}\,dx$, де a>0, b>0, наберіть:

```
> assume (a>0); assume (b>0);
> Int(sin(x)*cos(x)/(a^2*cos(x)^2+b^2*sin(x)^2),
```

```
x=0..Pi/2)=int(sin(x)*cos(x)/(a^2*cos(x)^2+b^2*
sin(x)^2),x=0..Pi/2);
```

$$\int_0^{\frac{1}{2}\pi} \frac{\sin(x)\cos(x)}{a^2\cos(x)^2+b^2\sin(x)^2}\,dx = -\frac{1}{2}\,\frac{-\ln(a^2)+\ln(b^2)}{a^2-b^2}$$

3. Щоб знайти невласний $\int_0^{+\infty}\frac{1-e^{(-a\sim x^2)}}{xe^{x^2}}dx$ інтеграл, де a>-1, наберіть:

```
> restart; assume(a>-1);
> Int((1-exp(-a*x^2))/(x*exp(x^2)),
 x=0..+infinity)=int((1-exp(-a*x^2))/(x*exp(x^2)),
 x=0..+infinity);
```

$$\int_0^{+\infty}\frac{1-e^{(-a\sim x^2)}}{xe^{x^2}}dx = \frac{1}{2}\ln(a\sim+1)$$

4. Щоб чисельно знайти інтеграл $\int_{\pi/6}^{\pi/4}\frac{\cos(x)}{x}dx$, наберіть:

```
> Int(cos(x) /x, x=Pi/6..Pi/4)=evalf(int(cos(x) /x
 x=Pi/6..Pi/4), 15);
```

$$\int_{\pi/6}^{\pi/4}\frac{\cos(x)}{x}dx = 0.322922981113732$$

5. Щоб повністю виконати всі етапи обчислення невизначеного інтеграла по частинах, наберіть:

```
> restart; with(student): J=Int(x^3*sin(x),x);
```

$$J = \int x^3\sin(x)dx$$

```
> J=intparts(Int(x^3*sin(x),x),x^3);
```

$$J = -x^3\cos(x) - \int -3x^2\cos(x)dx$$

```
> intparts(%,x^2);
```

$$J = -x^3\cos(x) + 3x^2\sin(x) + \int -6x\sin(x)dx$$

```
> intparts(%,x);
```

$$J = -x^3\cos(x) + 3x^2\sin(x) + 6x\cos(x) - \int 6\cos(x)dx$$

```
> value(%);
```

$$J = -x^3\cos(x) + 3x^2\sin(x) + 6x\cos(x) - 6\sin(x)$$

6. Щоб обчислити визначений інтеграл $\int_{-\pi/2}^{\pi/2} \frac{1}{1+\cos(x)}dx$ за допомогою універсальної підстановки.

```
> J=Int(1/(1+cos(x)), x=-Pi/2..Pi/2);
```

$$J = \int_{-\pi/2}^{\pi/2} \frac{1}{1+\cos(x)}dx$$

```
> J=changevar(tan(x/2)=t,Int(1/(1+cos(x))
x=-Pi/2..Pi/2), t);
```

$$J = \int_{-1}^{1} \frac{1}{(1+\cos(2\arctan(t)))(1+t^2)}dt$$

```
> value(%);
```

$$J=2$$

Контрольні завдання.

При виконанні контрольних завдань учням необхідно підставити замість символьних параметрів індивідуальні анкетні характеристики:

a - число символів в повному імені учня

b - число символів в по батькові учня

c - число символів в прізвищі учня.

У звіті на титульному аркуші необхідно обов'язково вказати, які анкетні дані використовувалися при виконанні контрольних завдань (ім'я, по батькові, прізвище).

Завдання.

1. Обчислити невизначені інтеграли відповідно до варіанта.
2. Обчислити невласні інтеграли.
3. Чисельно знайти інтеграли.
4. Повністю виконати всі етапи обчислення інтеграла по частинах.
5. Обчислити інтеграл за допомогою універсальної підстановки .
6. Обчислити подвійний інтеграл.

Контрольні запитання.

1. Що таке команди прямого і відкладеного виконання? Опишіть їх дії.
2. Які команди проводять аналітичну і чисельну інтеграцію? Опишіть їх параметри.
3. За допомогою яких команд вводяться обмеження на параметри для обчислення інтегралів, залежних від параметрів?
4. Для чого призначений пакет student?
5. Як описати команду інтегрування по частинах?
6. Як описати команду інтегрування методом заміни змінних?

ПРАКТИЧНА РОБОТА 6

Базові алгоритмічні структури. Типи алгоритмів

Мета.

Навчальна. Закріпити навички відтворення за блок-схемою алгоритму умови задачі; складання простіших блок-схем алгоритмів розв'язування задач.

Розвиваюча. Розвивати логічне та алгоритмічне мислення.

Виховна. Виховувати культуру оформлення блок-схем, алгоритмів

Запустіть Maple.

7. Після запуску Maple перший рядок є командним. Переведіть його в текстовий. Наберіть в цьому рядку: «Практична робота №1» і назва теми. Перейдіть на наступний рядок, натискуючи Enter.
8. У новому рядку наберіть «Виконав учень» і своє прізвище. Натискуйте Enter.
9. На наступному рядку наберіть «Завдання №1».
10. Збережіть свій файл на диску. Для цього в меню Fail виберіть пункт Save і наберіть ім'я вашого файлу у вигляді: Фамілія_1, де вказується ваше прізвище і 1 – номер практичної роботи.
11. Після цього в наступному рядку наберіть текст: «Файл із завданнями практичної роботи №1 збережений під ім'ям: Фамілія_n».

Актуалізація опорних знань

Що таке алгоритм?

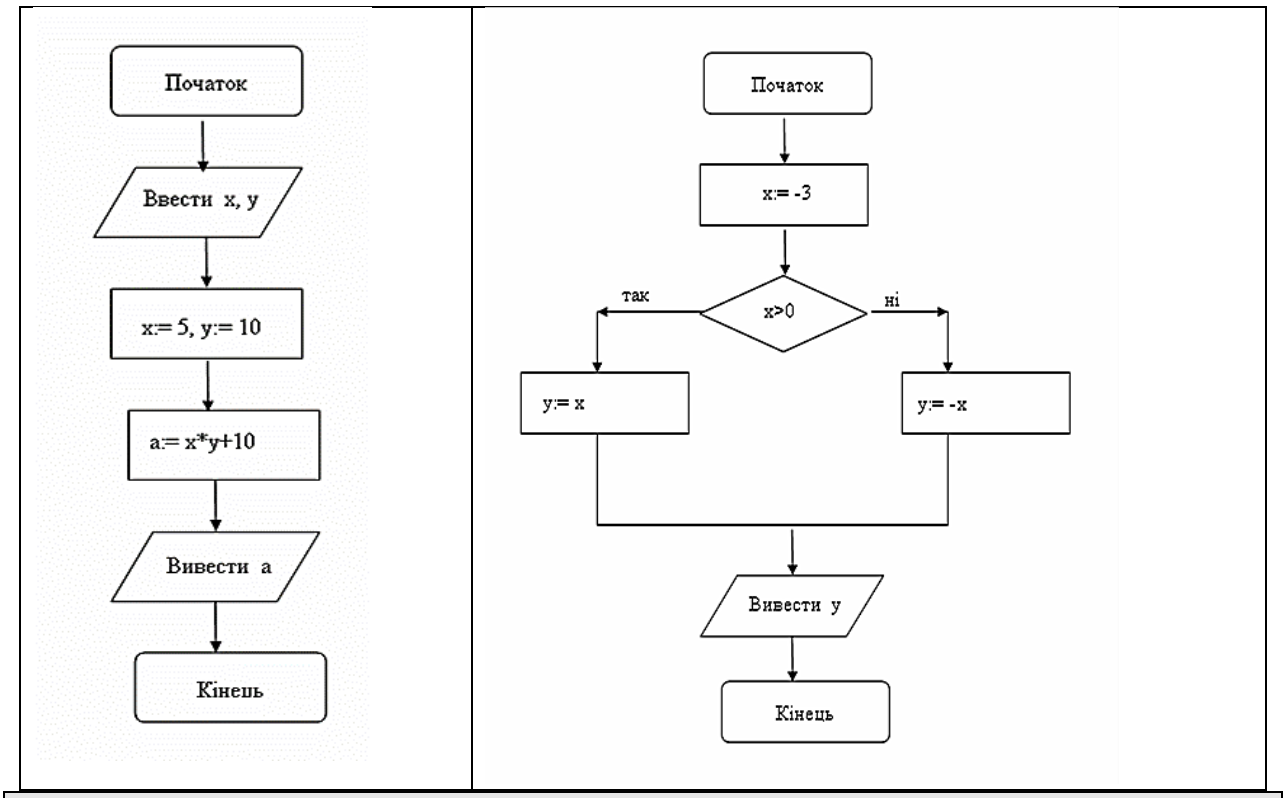
Які властивості повинен мати алгоритм?

Які способи представлення алгоритмів ви знаєте?

Назвіть базові алгоритмічні структури.

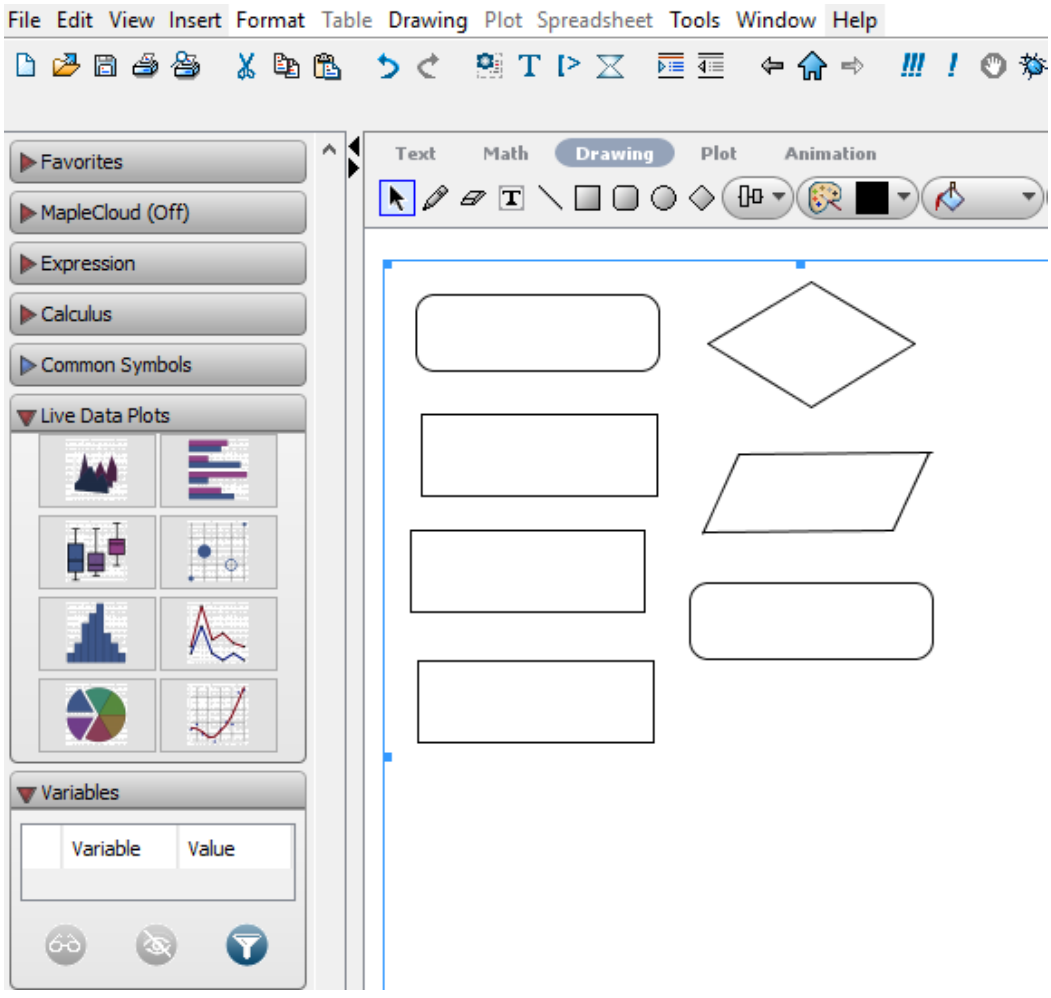
Виконання вправ

Вказати тип алгоритму та відтворити за блок-схемами алгоритмів умови задач та записати результат обчислення за алгоритмом:



Хід роботи

За наведеним сценарієм в СКМ Maple виконайте завдання по створенню алгоритмів.



1) Побудувати інформаційну модель відповідно до умови задачі.
2) Визначити аргументи, результати та проміжні величини алгоритму.
3) Розробити словесний опис алгоритму відповідно до умови задачі.
4) Побудувати схему алгоритму відповідно до умови задачі.
5) Записати алгоритм мовою псевдокодів відповідно до умови задачі.
6) Визначити наявність лінійних елементів в побудованому алгоритмі і вказати їх.
7) Визначити наявність розгалужених елементів в алгоритмі і вказати їх.
8) Визначити наявність циклічних елементів в побудованому алгоритмі і вказати їх.
9) Визначити загальний тип алгоритму або окремих його фрагментів.
10) Покроково виконати побудований алгоритм.

Скласти простіші блок-схеми алгоритмів розв'язування задач та назвати базові структури алгоритму, які були використані для написання алгоритму:

І варіант

1. Пончик за сніданком з'їв 5 тістечок, за обідом – у 2 рази більше, ввечері – стільки ж, скільки в обід та ще 4. Скільки всього тістечок з'їв Пончик за день.
2. Записати у вигляді схеми алгоритму послідовність виконання дій для обчислення такого виразу: (x+y)-x/y.
3. Обчислити площу поверхні та об'єм куба за заданою стороною.

ІІ варіант

1. У Вінні-Пуха було 10 грн. Він купив 2 шоколадки по 2 грн та 1 тістечко за 3 гривні. Скільки грошей залишилося у Вінні-Пуха?
2. Записати у вигляді схеми алгоритму послідовність виконання дій для обчислення такого виразу: x-y/(x+2).
3. Знайти довжину кола та площу круга за заданим радіусом.

Контрольні завдання

1. Повторити вивчений матеріал на попередніх уроках.
2. Оформіть словесним, графічним способами, алгоритм обчислення коренів квадратного рівняння.

ПРАКТИЧНА РОБОТА 7

Умовні вирази, оператори циклу, векторизація циклів.

1. МЕТА І ЗАВДАННЯ РОБОТИ

Мета роботи – засвоєння синтаксису основних операторів програмування в середовищі **Maple.**

Задача роботи – практичне вивчення роботи вказаних операторів **Maple.**

2. ТЕОРЕТИЧНІ ВІДОМОСТІ

1. Умовні вирази.

2. Цикли.

2.1. Цикл типу ***for***

2.2. Цикл типу ***while***

2.3. Цикл типу ***in***

3. Векторизація циклів

4. Вкладені цикли

5. Оператор пропуску елементу *next*

6. Оператор переривання циклу *break*

§1. Умовні вирази

Для підготовки програм, що розгалужуються, у Maple-мову програмування включений оператор **if,** можливі наступні варіанти конструкцій (тобто синтаксис команди):

1. **if** <*Умовний вираз*> **then** <*Послідовність тверджень*> **end if**
2. **if** <*Умовний вираз*> **then** <*Послідовність тверджень*> **else** <*Інша послідовність тверджень*> **end if**

або

if (Умовний вираз, Твердження1, Твердження2)

3. **if** <*Умовний вираз*> **then** <*Послідовність тверджень*> **elif** <*Умовний вираз*> **then** <*Послідовність тверджень*> **else** <*Послідовність тверджень*> **end if**

Замість оператора закінчення умовної конструкції **end if** можна також записати більш *короткий* оператор **fi.**

Інше застосування оператора **if** (зверніть увагу на використання кутових апострофів ‘ ‘!) вони фактично є функцією.

Приклади застосування різних конструкцій.

Приклад 1.1. Нехай необхідно проводити обчислення тільки з дійсними числами, а при наявності комплексних – тільки з їх модулями. Тобто необхідно перевірити тип даного та, у разі потреби, знайти його модуль. Це можна реалізувати наступним чином.

```
> restart:
a:=-sqrt(3):
b:=-3:
c:=3+3*I:
if Im(a)<>0 then a:=abs(a) end if:
if Im(b)<>0 then b:=abs(b)end if:
if Im(c)<>0 then c:=abs(c) fi:
'a'=a;
'b'=b;
'c'=c;
>
```

$$a = -\sqrt{3}$$

$$b = -3$$

$$c = 3\sqrt{2}$$

Приклад 1.2. Дуже часто необхідно робити якісь дії в залежності від значення вхідного аргументу. Наприклад, необхідно округлити дані до сотих. Звичайно можна використати параметр **Digits:**

```
> restart:
a:=7.245:
b:=2.436:
c:=-3.721:
if abs(frac(a*100))<0.5 then a:=(a*100-frac(a*100))/100 else a:=(a*100-
frac(a*100)+1)/100 fi:
if abs(frac(b*100))<0.5 then b:=(b*100-frac(b*100))/100 else b:=(b*100-
frac(b*100)+1)/100 fi:
if abs(frac(c*100))<0.5 then c:=(c*100-frac(c*100))/100 else c:=(c*100-
frac(c*100)+1)/100 fi:
```

```
a;
b;
c;
```

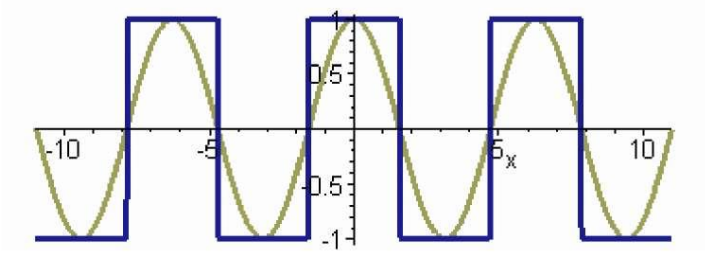
7.250000000

2.440000000

-3.720000000

Використання конструкції ***if*** можна наочно продемонструвати наступним чином

```
> restart:
plot([`if`(cos(x)>0,1,-1),cos(x)], x=-3.5*Pi..3.5*Pi,color=[navy,khaki],thickness
=3);
```




```
> plot([if cos(x)>0 then 1 else -1 fi,cos(x)], x=-3.5*
Pi..3.5*Pi,color=[navy,khaki],thickness=3);
```

Error, reserved word `if` unexpected

Тобто службове слово **if** програма сприймає як змінну.

Приклад 1.3. Використаємо правило округлення, коли точне значення 0,5 не змінюється

```
> restart:
a:=7.247:
b:=2.455:
c:=-3.721:
if abs(frac(a*100))<0.5 then a:=(a*100-frac(a*100))/100 elif
```

```
abs(frac(a*100))>0.5 then   a:=(a*100-frac(a*100)+1)/100 fi:
  if      abs(frac(b*100))<0.5      then      b:=(b*100-frac(b*100))/100      elif
abs(frac(b*100))>0.5 then   a:=(b*100-frac(b*100)+1)/100 fi:
  if      abs(frac(c*100))<0.5      then      c:=(c*100-frac(c*100))/100      elif
abs(frac(c*100))>0.5 then   a:=(c*100-frac(c*100)+1)/100 fi:
  a;
  b;
  c;
                                    7.250000000
                                      2.455
                                    -3.720000000
```

Зверніть увагу на дві речі. По-перше, необхідними ключовими словами оператора **if** є **if, then, end if (fi).** Усе інше є опційним. По-друге, після проведення розрахунків програма приводить точність даних до системної точності (перший та третій приклад).

Інколи послідовність тверджень необхідно (або зручно) записати декількома рядками. Тоді синтаксис наступний (**Приклад 1.4**):

```
> restart:
  a:=7.247:
  b:=2.451:
  c:=-3.721:
  "Обробка a";
  if
  abs(frac(a*100))<0.5
  then
  k:=(a*100-frac(a*100))/100:   #   необхідний   розділовий   знак   двох
послідовних тверджень
  a:=k:
```

```
elif abs(frac(a*100))>0.5
then
k:=(a*100-frac(a*100)+1)/100:
a:=k:
fi;  # кінець блока if
"Обробка b";
if abs(frac(b*100))<0.5 then
  k:=(b*100-frac(b*100))/100; # розділ тверджень
  b:=k
elif abs(frac(b*100))>0.5 then
  a:=(b*100-frac(b*100)+1)/100:
fi:
"Обробка c";
if abs(frac(c*100))<0.5 then k:=(c*100-frac(c*100))/100; c:=k elif
abs(frac(c*100))>0.5 then  a:=(c*100-frac(c*100)+1)/100 fi;
"Зверніть увагу на виводи результатів!";
a;
b;
c;
```

"Обробка a"

$k := 7.250000000$

$a := 7.250000000$

"Обробка b"

"Обробка c"

$k := -3.720000000$

$c := -3.720000000$

"Зверніть увагу на виводи результатів!"

7.250000000

2.450000000

-3.720000000

На виведення результатів розрахунків впливає **тільки** вказівка у кінці блоку **if.**

§2. Цикли

Часто буває необхідним циклічне повторення виконання виразу задану кількість разів або доти, доки виконується певна умова. Це здійснюється циклами двох типів: **for** та **while**.

2.1. Цикл типу ***for***

Maple має узагальнену конструкцію циклу **for,** яка може задаватися таким чином:

for <ім'я> **from** <вираз> **by** <вираз> **to** <вираз> **do** <послідовність тверджень> **end do;**

Інструкції, які треба виконати для об'єкту <ім'я>, починаючи з **(from)** певного значення, з кроком **(by),** до певного кінцевого значення **(to),** які стоять після ключового слова **do,** після чого зупинитися **(end do,** або **od).**

За замовченням (тобто, якщо спеціально не вказується у програмі) крок дорівнює 1.

Розглянемо приклад простого циклу.

Приклад 2.1:

```
> for n from 0 by 3 to 15 do a[n]:=n^2-n+1 od;
```

$$a_0 := 1$$

$$a_3 := 7$$

$$a_6 := 31$$

$$a_9 := 73$$

$$a_{12} := 133$$

$$a_{15} := 211$$

У цьому прикладі сформований цикл розпочав з $n = 0$ з кроком 3 (тобто були обрані значення n = 0, 3, 6, 9 ,12, 15) знаходити послідовні значення індексованої величини (a_n) за формулою $a_n = n\text{^}2 - n + 1$.

Керування показом результатів здійснюється завданням відповідного фіксатора виразу після ключового слова **end do (od).**

У циклі може використовуватися змінна, що має привласнене значення, однак воно змінюється, про що свідчить приклад:

```
> restart:
x:=1:
for x from 1 to 5 do x end do:
x;
```

$$6$$

Розглянемо використання циклу для більш алгоритмічної реалізації прикладу з конструкції оператора if.

Приклад 2.2:

```
> restart:
a1:=7.247:
a2:=2.455:
a3:=-3.721:
for i from 1 to 3 do
if abs(frac(a||i*100))<0.5 then a||i:=(a||i*100-frac(a||i*100))/100 elif
abs(frac(a||i*100))>0.5 then a||i:=(a||i*100-frac(a||i*100)+1)/100 fi:
od:
for i from 1 to 3 do
a||i:=a||i;
od;
```

$$a1 := 7.2500000$$

$$a2 := 2.455$$

$$a3 := -3.720000000$$

Нагадаємо, що оператор || конструює змінні поєднуючи ліву (не обчислюється) та праву (обчислюється) частини.

2.2. Цикл типу ***while***

Можлива наступна спрощена конструкція циклу типу **while:**

while <*Умовний вираз*> **do** <*Послідовність тверджень*> **od;**

Тут <*Послідовність тверджень*> виконуються, поки виконується логічний <*Умовний вираз*>.

Приклад 2.3 такого циклу:

```
> n:=1:
  while n<16 do n:=2*n od;
```

$$n := 2$$
$$n := 4$$
$$n := 8$$
$$n := 16$$

В даному прикладі йде подвоєння числа n з початковим значенням n=1 доти, доки воно є меньшим за 16.

2.3. Цикл типу ***in***

Є більш специфічна конструкція циклу. Вона використовує оператор in. Усередині цього оператора повинен бути заданий список значень, які набуватиме змінна:

```
> S:=[1,2,8,4];
  for i in S do s:=2*i od;
```

$$S := [1, 2, 8, 4]$$
$$s := 2$$
$$s := 4$$
$$s := 16$$
$$s := 8$$

§ 3. Векторизація циклів

Стандартні методи формування циклів за допомогою операторів **for** або **while** вимагають достатньо великого машинного часу. Іноді є можливість векторизувати цикли. Для цього використовується конструкція, що використовує відомий символьний оператор **$:**

expr$var=var1..var2

де **expr** – послідовність операторів, які необхідно виконати;

var – ім’я параметра циклу;

var1 – вираз, що задає початкове значення параметра циклу;

var2 – вираз, що задає кінцеве значення параметра циклу.

Приклад 3.1:

```
> i^2$i=1..5;
```

$$1, 4, 9, 16, 25$$

При цьому потрібно стежити, щоб ім’я умовної змінної було вільне від якого-небудь значення.

§ 4. Вкладені цикли

Оператори розгалуження часто використовуються усередині циклів. Це показано на наступному прикладі:

```
> S:=1:
  for i to 10 do
   if S>100 then S:=2*S: print(S)
    else S:=2*S
   fi
  od:
                                    256
                                    512
                                    1024
```

У даному прикладі йде подвоєння числа **S** з початковим значенням **S=1** без виведення результатів обчислень на екран доти, доки воно не перевищить числа 100. Після того обчислення продовжуються вже з виведенням результатів на екран.

§ 5. Оператор **next** *пропуску елементу*

Якщо в циклі необхідно пропустити одну ітерацію, то для цієї мети можна використовувати оператор **next.** Цей оператор використовується тільки в циклах усередині оператора розгалуження. Дія цього оператора полягає у тому, що на поточній ітерації не буде виконано ніяких обчислень, а почнеться виконання наступної ітерації.

```
> for i to 5 do if i=3 then next else print(i) fi od;
                                    1
                                    2
                                    4
                                    5
```

§ 6. Оператор **break** *переривання циклу*

Для переривання виконання циклу використовується оператор **break;** що використовується тільки в циклах усередині оператора розгалуження:

```
> for i to 5 do if i>3 then break else print(i) fi od;
                                    1
                                    2
                                    3
```

3. ХІД ВИКОНАННЯ

3.1. Запустити Maple (див. л.р. №1) і перезберегти порожній файл під власним ім’ям в певну папку за допомогою команди **Save As** в меню **File** (пп. *1.2.7*, л.р. №1).

3.2. Записати умовний оператор, за допомогою якого можна задати функцію:

$$f(x)=\begin{cases}0, при\, x<1\\ x, при\, 1\le x<2\\ 2, при\, x\ge 2\end{cases}$$

3.3. Записати умовний оператор, який в залежності від значення змінної K виводить на екран різні графіки – в разі, якщо K=SIN то малюється синусоїда у діапазоні від –π до π, в разі, якщо K=EXP, то малюється експонента від 0 до 1, в разі, якщо K=COS – малюється косинусоїда від –π до π, якщо ж значення K не відповідає жодному з перерахованих – на екран повинно виводитися повідомлення ERROR, EMPTY PLOT.

3.4. Побудувати цикл, що виводить значення функції $f=\sin(i^2)/i!$ для відрізка натурального ряду i=1..10.

3.5. Знайти суму всіх парних чисел від 12 до 100.

3.6. Знайти добуток усіх непарних чисел від 1 до 100.

3.7. За допомогою циклу while обчислити значення виразу $\sqrt{1+..\sqrt{1+\sqrt{1+\sqrt{2}}}}$, цикл зупинити після здійснення 15 ітерацій.

3.8. Скласти програму, що обчислює функцію для значень x=(i–1)/3 при i=1..10 та виводить результати на екран.

3.9. Задати цикл, що обчислює вираз за формулою $S=n^2$ для n=1,2,3,…, але по досягненні значення S величини більшої за 2000 вираз надалі обчислюється за формулою S=sin(n)*n . Підрахунки проводити до n<50.

ПРАКТИЧНА РОБОТА 8.

V. Лінійна алгебра

1. Векторна алгебра.
2. Дії з матрицями.
3. Системи лінійних рівнянь. Матричні рівняння.

§1. Векторна алгебра

Основна частина команд для розв’язання завдань лінійної алгебри міститься в бібліотеці linalg. Тому перед розв’язанням завдань з матрицями і векторами потрібно завантажити цю бібліотеку командою with(linalg).

Способи задання векторів.

Для визначення вектора в Maple використовується команда vector([x1,x2.,xn]), де в квадратних дужках через кому вказуються координати вектора. Наприклад:

> **x:=vector([1,0,0]);**

$$x:=[1, 0, 0]$$

Певну координату вже заданого вектора x можна отримати в рядку виводу, якщо ввести команду x[i], де i (номер координати. Наприклад, першу координату заданого в попередньому прикладі вектора можна вивести так:

> **x[1];**

$$1$$

Вектор можна перетворити в список i, навпаки, за допомогою команди convert(vector, list) або convert(list, vector).

Додавання векторів.

Додати два вектори a і b можна за допомогою двох команд:

1) **evalm(a+b);**

2) **matadd(a,b).**

$a := vector([1, 0, 0])$

$$\begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

$b := vector([1, 1, 1])$

$$\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

$evalm(a + b)$

$$\begin{bmatrix} 2 & 1 & 1 \end{bmatrix}$$

$matadd(\mathrm{a}, b);$

$$matadd(a, b)$$

Команда add дозволяє обчислювати лінійну комбінацію векторів a і b:, де (скалярні величини, якщо використовувати формат: matadd(a,b,alpha,beta).

Скалярний та векторний добуток векторів і кут між векторами.

Скалярний добуток двох векторів обчислюється командою dotprod(a,b).

Векторний добуток двох векторів обчислюється командою crossprod(a,b).

Кут між двома векторами a і b обчислюється за допомогою команди angle(a,b).

Нормування вектора.

Норму (довжину) вектора a можна обчислити за допомогою команди norm(a,2).

Можна нормувати вектор a за допомогою команди normalize(a), в результаті виконання якої буде отриманий вектор одиничної довжини.

Знаходження базису системи векторів. Ортогоналізація системи векторів за процедурою Грамма-Шмідта.

Якщо є система n векторів, то за допомогою команди basis([a1,a2.,an]) можна знайти базис цієї системи.

За допомогою команди GramSchmidt([a1,a2.,an]) можна ортогонализовать систему незалежних векторів .

Приклади операцій над векторами

1. Щоб знайти і кут між даними векторами a і b, наберіть:

> **with(linalg):**

> **a:=([2,1,3,2]); b:=([1,2,-2,1]);**

$$a:=[2,1,3,2]$$
$$b:=[1,2,-2,1]$$

> **dotprod(a,b);**

$$0$$

> **phi=angle(a,b);**

$$\phi = \frac{1}{2}\pi$$

2. Щоб знайти векторний добуток, а потім скалярний добуток, наберіть:

> **restart; with(linalg):**

> **a:=([2,-2,1]); b:=([2,3,6]);**

$$a:=[2,2,1]$$
$$b:=[2,3,6]$$

> **c:=crossprod(a,b);**

$$c:=[15,10,10]$$

> **dotprod(a, з);**

$$0$$

3. Щоб знайти норму вектора, наберіть:

> **restart; with(linalg):**

> **a:=vector([1,2,3,4,5,6]): norm(a,2);**

$$\sqrt{91}$$

4. З системи векторів: виокремити базис і ортогоналізувати його за процедурою Грамма-Шмідта:

> **restart; with(linalg):**

> **a1:=vector([1,2,2,-1]): a2:=vector([1,1,-5,3]): a3:=vector([3,2,8,7]): a4:=vector([0,1,7,-4]): a5:=vector([2,1,12,-10]):**

> **g:=basis([a1,a2,a3,a4,a5]);**

$$g := [a1, a2, a3, a5]$$

> **GramSchmidt(g);**

$$\left[[1, 2, 2, -1], [2, 3, -3, 2], \left[\frac{81}{65}, -\frac{93}{65}, \frac{327}{65}, \frac{549}{65}\right], \left[\frac{1633}{724}, -\frac{923}{724}, -\frac{71}{724}, -\frac{355}{724}\right]\right]$$

§2. Дії з матрицями

Визначення матриці.

Для визначення матриці в Maple можна використовувати команду matrix(n, m [[a11,a12.,a1n], [a21,a22.,a2m],., [an1,an2.,anm]]), де n (число рядків, m – число стовпців в матриці. Ці числа задавати необов'язково, а досить перерахувати елементи матриці відрядковий в квадратних дужках через кому. Наприклад:

> **A:=matrix([[1,2,3],[-3,-2,-1]]);**

$$A := \begin{bmatrix} 1 & 2 & 3 \\ -3 & -2 & -1 \end{bmatrix}$$

У Maple матриці спеціального вигляду можна генерувати за допомогою додаткових команд. Зокрема діагональну матрицю можна отримати командою diag. Наприклад:

> **J:=diag(1,2,3);**

$$J := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$$

Генерувати матрицю можна за допомогою функції f(i, j) від змінних i, j – індексів матриці: matrix(n, m, f), де де n – число рядків, m – число стовпців. Наприклад:

> **f:=(i, j)->x^i*y^j;**

$$f := (i, j) \to x^i y^j$$

> **A:=matrix(2,3,f);**

$$A := \begin{bmatrix} xy & xy^2 & xy^3 \\ x^2y & x^2y^2 & x^2y^3 \end{bmatrix}$$

Число рядків в матриці A можна визначити за допомогою команди rowdim(A), а число стовпців – за допомогою команди coldim(A).

Арифметичні операції з матрицями.

Додавання двох матриць однакової розмірності здійснюється тими ж командами, що і додавання векторів: evalm(A+B) або matadd(A,B). Добуток двох матриць A і B, може бути знайдений за допомогою двох команд:

1) **evalm(A&*B);**
2) **multiply(A,B).**

Як другий аргумент в командах, що обчислюють добуток, можна вказувати вектор, наприклад:

> **A:=matrix([[1,0],[0,-1]]);**

> **B:=matrix([[-5,1], [7,4]]);**

$$A := \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad B := \begin{bmatrix} -5 & 1 \\ 7 & 4 \end{bmatrix}$$

> **v:=vector([2,4]);**

$$v := [2,4]$$

> **multiply(A,v);**

$$[2,-4]$$

> **multiply(A,B);**

$$\begin{bmatrix} -5 & 1 \\ -7 & -4 \end{bmatrix}$$

> **matadd(A,B);**

$$\begin{bmatrix} -4 & 1 \\ 7 & 3 \end{bmatrix}$$

Команда **evalm** дозволяє також додавати до кожного елемента матриці число і умножати матрицю на число. Наприклад:

> **C:=matrix([[1,1],[2,3]]):**

> **evalm(2+3*C);**

$$\begin{bmatrix} 5 & 3 \\ 6 & 11 \end{bmatrix}$$

Визначники, мінори і доповнення. Ранг і слід матриці.

Визначник матриці A обчислюється командою det(A). Команда minor(A,i,j) повертає матрицю, отриману з вихідної матриці A викреслюванням i-ой рядки і j-ого стовпця. Мінор Mij елементу a_{ij} матриці A можна обчислити командою det(minor(A,i,j)). Ранг матриці A обчислюється командою rank(A). Слід матриці A, рівний сумі її діагональних елементів, обчислюється командою trace(A).

> **K:=matrix([[4,0,5],[0,1,-6],[3,0,4]]);**

$$A := \begin{bmatrix} 4 & 0 & 5 \\ 0 & 1 & -6 \\ 3 & 0 & 4 \end{bmatrix}$$

> **det(K);**

$$1$$

> **minor(K,3,2);**

$$\begin{bmatrix} 4 & 5 \\ 0 & -6 \end{bmatrix}$$

> **det(%);**

$$-24$$

> **trace(K);**

$$9$$

Обернена і транспонована матриці.

Обернену матрицю можна обчислити двома способами:

1) **evalm(1/A);**
2) **inverse(A).**

Транспонування матриці А – це зміна місцями рядків і стовпців. Отримана в результаті цього матриця називається транспонованою і позначається А’. Транспоновану матрицю А’ можна обчислити командою transpose(A).

Наприклад, використовуючи задану в попередньому пункті матрицю K, знайдемо її обернену і транспоновану:

> **inverse(K);**

$$\begin{bmatrix} 4 & 0 & -5 \\ -18 & 1 & 24 \\ -3 & 0 & 4 \end{bmatrix}$$

> **multiply(K,%);**

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

> **transpose(K);**

$$\begin{bmatrix} 4 & 0 & 3 \\ 0 & 1 & 0 \\ 5 & -6 & 4 \end{bmatrix}$$

Функції від матриць.

Зведення матриці А в ранг n проводиться командою evalm(A^n). Обчислення матричної експоненти можливе за допомогою команди exponential(A). Наприклад:

> **with(linalg, exponential):**

T:=matrix([[5*a,2*b],[-2*b,5*a]]);

$$T := \begin{bmatrix} 5a & 2b \\ -2b & 5a \end{bmatrix}$$

> **exponential(T);**

$$\begin{bmatrix} e^{(5a)}\cos(2b) & e^{(5a)}\sin(2b) \\ -e^{(5a)}\sin(2b) & e^{(5a)}\cos(2b) \end{bmatrix}$$

> **evalm(T^2);**

$$\begin{bmatrix} 25a^2-4b^2 & 20ab \\ -20ab & 25a^2-4b^2 \end{bmatrix}$$

Приклади операцій над матрицями

1. Дані матриці: $A=\begin{bmatrix}4 & 3\\ 7 & 5\end{bmatrix}$; $B=\begin{bmatrix}-28 & 93\\ 38 & -126\end{bmatrix}$; $C=\begin{bmatrix}7 & 3\\ 2 & 1\end{bmatrix}$

Щоб знайти: (AB)C, detA, detB, detC, det[(AB)C], наберіть:

> **restart;**

with(linalg): A:=matrix([[4,3],[7,5]]):

> **B:=matrix([[-28,93],[38,-126]]):**

> **C:=matrix([[7,3],[2,1]]):**

> **F:=evalm(A&*B&*C);**

$$F=\begin{bmatrix}2 & 0\\ 0 & 3\end{bmatrix}$$

> **Det(A)=det(A); Det(B)=det(B); Det(C)=det(C);**

Det(F)=det(F);

Det(A)=(1

Det(B)=(6

Det(C)=1

Det(F)=6

2. Щоб для даної матриці, знайти: det A, A,, det(M22), наберіть:

> **A:=matrix ([[2,5,7],[6,3,4],[5,-2,-3]]);**

$$A:=\begin{bmatrix}2 & 5 & 7\\ 6 & 3 & 4\\ 5 & -2 & -3\end{bmatrix}$$

> **Det(A)=det(A);**

Det(A)=(1

> **transpose(A);**

$$\begin{bmatrix}2 & 6 & 5\\ 5 & 3 & -2\\ 7 & 4 & -3\end{bmatrix}$$

> **inverse(A);**

$$\begin{bmatrix} 1 & -1 & 1 \\ -38 & 41 & -34 \\ 27 & -29 & 24 \end{bmatrix}$$

> **det(minor(A,2,2));**

$$41$$

3. Щоб знайти ранг даної матриці, наберіть:

> **A:=matrix([[8,-4,5,5,9], [1,-3,-5,0,-7], [7,-5,1,4,1], [3,-1,3,2,5]]):**

> **r(A)=rank(A);**

$$r(A)=3$$

4. Щоб обчислити det, наберіть:

> **with(linalg, exponential); exponential([[3,-1],[1,1]]);**

$$\begin{bmatrix} 2e^2 & -e^2 \\ e^2 & 0 \end{bmatrix}$$

5. Щоб для даної матриці знайти значення многочлена, наберіть:

> **A:=matrix([[5,1,4],[3,3,2],[6,2,10]]):**

> **P(A)=evalm(A^3-18*A^2+64*A);**

$$P(A)=\begin{bmatrix} 64 & 0 & 0 \\ 0 & 64 & 0 \\ 0 & 0 & 64 \end{bmatrix}.$$

§3. Системи лінійних рівнянь. Матричні рівняння.

Системи лінійних рівнянь і матричні рівняння.

Система лінійних рівнянь може бути розв'язана двома способами.

Спосіб 1: стандартна команда solve знаходить майбутній розв'язок системи лінійних рівнянь, записаних в розгорнутому вигляді:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n = b_1 \\ .... \\ a_{m1}x_m + a_{m2}x_2 + ... + a_{mn}x_n = b_m \end{cases}.$$

Спосіб 2: команда linsolve(A,b) з пакету linalg знаходить розв'язання рівняння. Аргументи цієї команди: A – матриця, b – вектор.

За допомогою команди linsolve(A,b) можна знайти розв'язання матричного рівняння AX=B, якщо як аргументи цієї команди вказати, відповідно, матриці A і B.

Ядро матриці.

Ядро матриці A – це безліч векторів x таких, добуток матриці A на яких рівно нульовому вектору. Пошук ядра матриці A еквівалентний розв'язанню

системи лінійних однорідних рівнянь. Знайти ядро матриці A можна командою kernel(A).

Завдання 3.

1. Щоб знайти загальний і одиничний розв'язок системи, наберіть:

> **eq:={2*x-3*y+5*z+7*t=1, 4*x-6*y+2*z+3*t=2, 2*x-3*y-11*z-15*t=1}:**

> **s:=solve(eq{x,y,z});**

$$s:=\{, y=y\ \}$$

Для знаходження числового розв'язку потрібно виконати підстановку конкретного значення однієї із змінних за допомогою команди subs:

> **subs({y=1,t=1},s);**

{,, 1=1}

2. Щоб розв'язати матричне рівняння: Ax=B; наберіть:

> **A:=matrix([[1,2],[3,4]]):**

> **B:=matrix([[3,5],[5,9]]):**

> **X:=linsolve(A,B);**

$$X := \begin{bmatrix} -1 & -1 \\ 2 & 3 \end{bmatrix}$$

3. Дана матриця $A := \begin{bmatrix} 1 & 1 & 0 \\ 0 & 2 & -1 \\ 1 & 3 & -1 \end{bmatrix}$. Щоб знайти її ранг **r**, дефект: **d(A)=n–r(A)**, де **n** – розмір квадратної матриці, а також ядро A, наберіть:

> **A:=matrix([[1,1,0],[0,2,-1],[1,3,-1]]):**

> **r(A):=rank(A);**

r(A):=2

> **d(A):=rowdim(A)-r(A);**

d(A):=1

> **d(A):=kernel(A);**

d(A):={[(1,1,2]}

Контрольні завдання.

При виконанні контрольних завдань учневі необхідно підставити замість буквених параметрів індивідуальні анкетні характеристики:

a - число букв в повному імені учня

b - число букв по батькові учня

c - число букв в прізвищі учня.

У звіті на титульному аркуші необхідно обов'язково вказати, які дані використовувалися при виконанні контрольних завдань (ім'я, по батькові, прізвище).

Завдання.

1. Дані вектори. Виконати наступні завдання відповідно до варіанта:

а) знайти суму;
б) знайти добуток;
в) знайти кут між векторами.
2. Дані матриці, Обчислити: ранг, дефект, ядро.
3. Обчислити визначники для заданих матриць.
4. Знайти обернені для заданих матриць.
5. Дана матриця:
а) Привести матрицю до трикутного вигляду.
б) Обчислити M23.
в) Знайти ранг матриці.
6. Знайти власні значення і власні вектори матриці.
7.Розв’язати матричні рівняння.

Контрольні питання.

1. Який пакет потрібно завантажити перед розв’язанням завдань лінійної алгебри в Maple?
2. За допомогою яких команд можна ввести вектор, матрицю?
3. Якими двома командами можна скласти два вектори однакової розмірності (2 матриці)?
4. Які види дій над векторами обчислюються Maple і які команди для цього використовуються?
5. Як обчислити норму вектора?
6. Як обчислити кут між двома векторами?
7. Як знайти базис системи векторів і побудувати ортогональний базис системи векторів.
8. Якими двома командами можна обчислити добуток двох матриць (або матриці на вектор)?
9. Які команди використовуються для знаходження визначника, мінору, доповнення алгебри, сліду матриці?
10. Що таке дефект матриці? Як знайти дефект квадратної матриці. Які команди при цьому використовуються?
11. Яка матриця називається оберненою до даної матриці і якими способами вона обчислюється в Maple?
12. Перерахуєте спеціальні види матриць і команди, що наводять матриці до цих форм.
13. Що називається ядром матриці, і яка команда використовується для його знаходження?
14. Яка команда дозволяє розв’язувати матричні рівняння?

ПРАКТИЧНА РОБОТА 9

Масиви, вкладені цикли, візуалізація масивів даних

МЕТА І ЗАВДАННЯ РОБОТИ

Мета роботи – засвоєння синтаксису основних структурних даних в середовищі **Maple**.

Задача роботи – практичне вивчення роботи вказаних операторів **Maple.**

2. ТЕОРЕТИЧНІ ВІДОМОСТІ

2.1. Масиви. Формування одновимірних масивів. Перетворення одновимірних масивів в матрицю-стовпчик

Для створення масивів служить оператор **array:**

array(a1..b1, a2..b2, … an..bn)

Даний оператор створює масив розмірності **(b1-a1+1)x(b2-a2+1)x…x(bn-an+1)**.

Наприклад:

```
> A:=array(1..2,4..5);
  print(A);
```

$$A := \text{array}(1 .. 2, 4 .. 5, [\ ])$$

```
array(1 .. 2, 4 .. 5, [
  (1, 4) = A[1, 4]
  (1, 5) = A[1, 5]
  (2, 4) = A[2, 4]
  (2, 5) = A[2, 5]
  ])
```

Для заповнення елементів одновимірного масиву використаємо цикл:

```
> A:=array(1..3);
  for i to 3 do
  A[i]:=i^3
  od:
  print(A);
```

$$A := \text{array}(1 .. 3, [\ ])$$

$$[1, 8, 27]$$

Для перетворення отриманого масиву в матрицю-стовпчик використовуємо оператор **convert:**

```
> Am:=convert(A,matrix);
```

$$Am := \begin{bmatrix} 1 \\ 8 \\ 27 \end{bmatrix}$$

Вказані операції можна векторизувати. Це робиться таким чином:

```
> [i^3$i=1..3];
  A:=convert(%,array):
  Am:=convert(%,matrix):
  print(Am);
```

$$[1, 8, 27]$$

$$\begin{bmatrix} 1 \\ 8 \\ 27 \end{bmatrix}$$

2.2. Вкладені цикли. Формування багатовимірних масивів. Перетворення двовимірних масивів в матрицю.

Цикли можуть бути *вкладеними*. Це ілюструє наступний приклад, що створює матрицю $E = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$ на базі заданого масиву:

```
> A:=array(1..3,1..3):
  for i to 3 do
   for j to 3 do
    if i=j then
    A[i,j]:=1
    else A[i,j]:=0
    fi
   od
  od:
  E:=evalm(A);
```

$$E := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Тут використана функція **evalm** – перетворення масиву в еквівалентну матрицю.

2.3. Векторизація вкладених циклів

Вкладені цикли також можна векторизувати. Переваги векторизації циклів демонструє наступний приклад формування двовимірного списку даних вигляду $1.2(i+j)^2$ з одночасним обчисленням часу, що витрачається на формування вказаних списків:

```
> restart:
  T1:=time():
  for i to 500 do
   for j to 500 do
    A[i,j]:=1.2*(i+j)^2
   od
  od:
  T2:=time():
  T:=T2-T1;
  A[1,1],A[100,222],A[500,500];
```

$$T := 8.473$$

$$4.8,\ 124420.8,\ 0.12000000\ 10^{7}$$

```
> restart:
  T1:=time():
  A:=[[1.2*(i+j)^2$i=1..500]$j=1..500]:
  T2:=time():
  T:=T2-T1;
  A[1,1],A[100,222],A[500,500];
```

$$T := 3.265$$

$$4.8,\ 124420.8,\ 0.12000000\ 10^{7}$$

Як видно з прикладу, векторизація істотно прискорює обчислення.

2.4. Візуалізація масивів даних

Масиви даних можуть бути візуалізовані. Для цього заздалегідь необхідно завантажити графічну бібліотеку за допомогою команди:

with(plots)

Безпосередньо візуалізація масивів проводиться оператором matrixplot з наступним синтаксисом:

matrixplot(A, options)

де **A** – масив даних (матриця);

options – параметри, аналогічні параметрам оператора **plot3d** (функція для побудови тривимірних графіків (3d-типу)).

E:\f\diss 14\lab 3.mw* - [Server 5] - Maple 15

File Edit View Insert Format Table Drawing Plot Spreadsheet Tools Window Help

Favorites

MapleCloud

Search

public

mconessa@maple...	13.05.2015
Equations Parabole	
physicsleninac@h...	25.04.2015
Fuerza-Masa y Aceleracion	
physicsleninac@h...	16.04.2015
Relaciones y Funciones	
jmay@maplesoft....	16.04.2015
Bike Tires vs. Ideal Gas Law ...	
physicsleninac@h...	08.04.2015
Inversa y Composicion de F...	
jfriebe@maplesof...	23.03.2015
Definition Ableitung	
jfriebe@maplesof...	23.03.2015
Volumen einer Kugel	
jfriebe@maplesof...	23.03.2015
Radioaktiver Zerfall	
jfriebe@maplesof...	23.03.2015
Brechungsgesetz	
physicsleninac@h...	23.03.2015
Movimiento Libre No Amortig...	

powered by Google™

Variables

Variable	Value

Handwriting

Expression

Units (SI)

Units (FPS)

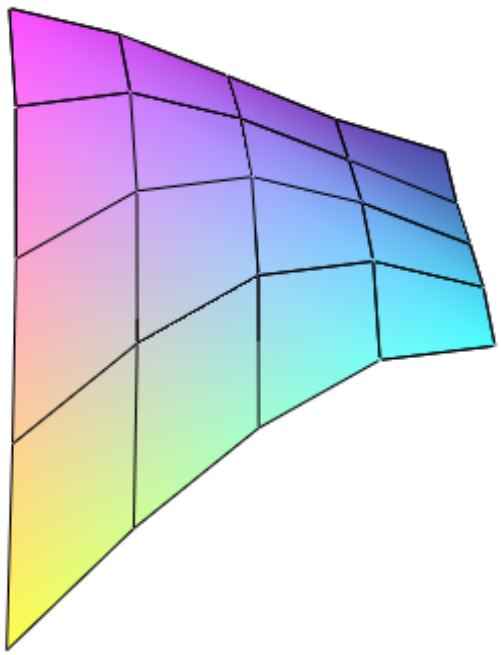
Text Math Drawing Plot Animation Hide

90 42 6 1:1

```
A := array(1..5, 1..5);
for i to 5 do
 for j to 5 do
A[i,j] := evalf(sin((i+j)^2/25), 2)
od od:
print(A);
with(plots):
matrixplot(A);
```

$$array(1..5, 1..5, [\,])$$

$$\begin{bmatrix} 0.16 & 0.35 & 0.60 & 0.84 & 0.99 \\ 0.35 & 0.60 & 0.84 & 0.99 & 0.91 \\ 0.60 & 0.84 & 0.99 & 0.91 & 0.52 \\ 0.84 & 0.99 & 0.91 & 0.52 & -0.058 \\ 0.99 & 0.91 & 0.52 & -0.058 & -0.76 \end{bmatrix}$$



3. ХІД РОБОТИ

3.1. Запустити Maple і перезберегти порожній файл під власним ім'ям за допомогою команди **Save As** в меню **File**.

3.2. Задати масив з 10 елементів, елементами якого будуть квадрати чисел натурального ряду.

3.3. Створити масив, що задає діагональну матрицю $\begin{bmatrix} 0 & 0 & 0 & 4 \\ 0 & 0 & 3 & 0 \\ 0 & 2 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$ і перетворити його безпосередньо на матрицю.

3.4. Візуалізувати двомірний масив з деяких ваших чисел.

ПРАКТИЧНА РОБОТА 10

Вивчення функцій для опрацювання матриць. Метод Жордана-Гауса

1. МЕТА І ЗАДАЧА РОБОТИ

Мета роботи – засвоєння синтаксису основних функцій для опрацювання матриць програмування в середовищі **Maple.**

Задача роботи – практичне вивчення роботи вказаних функцій **Maple.**

ХІД РОБОТИ

1.1 Постановка задачі

Нехай дано систему n лінійних алгебраїчних рівнянь з n змінними

$\sum_{j=1}^{n} a_{ij} x_j = b_i$, (I=1.2…..n) (1)

Систему (1) можна записати у вигляді одного матричного рівняння

AX=B, (2)

де

$$A = (a_{ij}) = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

матриця коефіцієнтів a_{ij} (індекс «*i*» вказує номер рівняння, якому належить коефіцієнт, а індекс «*j*» – змінну номерації, при якій він стоїть),

$$B = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}, \quad X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix},$$

відповідно стовпчик вільних членів і стовпчик змінних.

Упорядкована сукупність n чисел $c_1, c_2, \ldots, c_n$, яка, будучи підставленою в систему (1) замість $x_1, x_2, \ldots, x_n$, перетворює всі рівняння в правильні числові рівності, називається розв'язком системи (1)

$$\Delta = \det A = \begin{vmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{vmatrix} \neq 0$$

Методи розв'язування систем лінійних рівнянь можна поділити на дві групи: точні й ітераційні.

Точними називають такі методи, які дають змогу знайти точний розв'язок системи (1) за допомогою виконання скінченої кількості арифметичних операцій у припущенні, що всі обчислення виконуються точно (без округлень), а коефіцієнти системи і вільні члени – точні числа. Але на практиці всі обчислення виконуються з обмеженою кількістю десяткових розрядів, а ірраціональні коефіцієнти і вільні члени, якщо такі є, замінюються раціональними числами. Тому в процесі обчислення вдаються до округлень, а це означає, що розв'язки, які обчислюються за точними методами, фактично є наближеними числами з певними похибками (похибками округлень). До точних належать метод Гаусса, метод квадратних коренів, правило Крамера, сюди ж належить метод Жордана-Гаусса.

Інтераційними називають такі методи, які дають змогу знайти наближений розв'язок системи (1) із заздалегідь вказаною точністю шляхом виконання скінченої кількості арифметичних операцій, хоч самі обчислення можуть проводитись і без округлень, а коефіцієнти і вільні члени системи бути точними числами.

У процесі вивчення різних питань економіки, природознавства, техніки тощо доводиться розв'язувати системи алгебраїчних рівнянь. Зокрема, до таких систем зводиться чисельне розв'язування лінійних, диференціальних та інтегральних рівнянь. У таких системах коефіцієнти і вільні члени рівнянь – числа наближені. А це веде до появи додаткових (так званих неусувних) похибок.

Якщо систему рівнянь у пам'яті машини записати навіть точно, то в процесі її розв'язування ЕОМ обов'язково виникнуть похибки округлень, які не можуть не вплинути на точність розв'язку. Проте, якщо матриця А системи (2) майже вироджена, то невиключено, що малі зміни в коефіцієнтах і (або) вільних членах призведуть до значних змін у її розв'язку.

Якщо не великі зміни коефіцієнтів і (або) вільних членів системи (1) дуже ускладнюють її розв'язок, то таку систему рівнянь називають погано обумовленою. Якщо ж не значні зміни коефіцієнтів і (або) вільних членів системи (1) мало збурюють її розв'язок, то таку систему називають добре обумовленою. Прикладом погано обумовленої є, наприклад, система вигляду:

$$\begin{cases} 6{,}1x_1 + 3{,}4x_2 = 6{,}1 \\ 14{,}7x_1 + 8{,}2x_2 = 14{,}7 \end{cases} \quad (3)$$

розв'язком якої є пара (1;0). Якщо число 6,1 у правій частині першого рівняння системи (3) змінити на 0,02, то система

$$\begin{cases} 6{,}1x_1 + 3{,}4x_2 = 6{,}12 \\ 14{,}7x_1 + 8{,}2x_2 = 14{,}7 \end{cases}$$

матиме розв'язком пару (5,1;-7,35). Отже, не велике відхилення (меньше 0,33%) одного з вільних членів системи (3) зовсім змінило розв'язок системи.

На щастя, на практиці системи рівнянь, погано обумовлені, зустрічаються дуже рідко.

2. ТЕОРЕТИЧНІ ВІДОМОСТІ

2. Методи розв'язування задачі

Метод Жордана-Гаусса був розроблений двома вченими Жорданом та Гаусом (від яких і пішла назва методу). Цей метод вони помітили після довгої практики роботи з системами рівнянь. Це можна пояснити складністю розв'язку цим методом.

Суть методу полягає в тому, щоб послідовно за певним правилом вилучати один, за одним стовпці елементів квадратної матриці, які стоять біля відповідних невідомих.

3. ХІД РОБОТИ

3.1 Алгоритм розв'язку задачі

Розглянемо систему m лінійних рівнянь з n невідомими. Для методу Жордана-Гауса її зручно зобразити у вигляді такої таблиці:

	x_1	x_2	$\cdots$	x_j	$\cdots$	x_n	b_i
y_1	a_{11}	a_{12}	$\cdots$	a_{1j}	$\cdots$	a_{1n}	b_1

y_2	a_{21}	a_{22}	$\cdots$	a_{2j}	$\cdots$	a_{2n}	b_2
…	…	…	…	…	…	…	…
y_i	a_{i1}	a_{i2}	$\cdots$	a_{ij}	$\cdots$	a_{in}	b_i
y_m	a_{m1}	a_{m2}	$\cdots$	a_{mj}	$\cdots$	a_{mn}	b_m

Знайдемо змінну x_j з і-го рівняння системи, та підставимо знайдений вираз замість x_j в усі інші рівняння системи. Таке перетворення системи називають кроком Жорданових виключень з основним елементом a_{ij}.

Такі перетворення зручно виконувати користуючись таблицею (1), яка перейде в іншу таблицю за наступними правилами:

Усі вільні елементи реально заданої системи лінійних алгебраїчних рівнянь, тобто стовпець b_i замінюють на протилежні;

Основний елемент a_{ij} замінюють на одиницю. Над основним стовпчиком записують y_i, а біля рядка x_y;

Інші елементи основного стовпчика j залишають без змін;

Інші елементи основного рядка і-го змінюють лише свої знаки;

Елементи, які не належать розв'язуючому рядку або стовпчику обчислюють наступним чином. Створюється двовимірний визначник, який складається з таких елементів попередньої таблиці:

а) елемента з цими ж індексами, що й в обчислювальному елементі;

б) основний елемент;

в) елемент, який є спільним для стовпця з елементом (а) і стовпця з елементом (б);

г) елемент, який є спільним для стовпця з елементом (б) і рядка з елементом (а).

Шуканий елемент обчислюється як добуток елементів (а) та (б) мінус добуток залишившихся елементів визначника.

Цю дію можна зобразити у вигляді формули:

$$a_{ij} = \begin{vmatrix} a_{kh} & a_{kj} \\ a_{ih} & a_{ij} \end{vmatrix} = a_{ij} * a_{kh} - a_{ih} * a_{kj}.$$

Всі елементи нової таблиці ділять на елемент a_{ij}. Тим самим створюють ще одну таблицю, але вже без стовпця з елементом a_{ij}, а рядок з елементом a_{ij} позначають так, як позначили вилучений стовпець.

Виконавши всі описані операції новостворена таблиця матиме вигляд:

	x_1	x_2	⋯	x_n	b_i
y_1	a_{11}	a_{12}	⋯	a_{1n}	b_1
y_2	a_{21}	a_{22}	⋯	a_{2n}	b_2
…	…	…	…	…	…
x_j	a_{i1}	a_{i2}	⋯	a_{in}	b_i
…	…	…	…	…	…
y_m	a_{m1}	a_{m2}	⋯	a_{mn}	b_m

Шукаючи невідомі $x_1 \cdots x_n$ системи лінійних алгебраїчних рівнянь продовжують виконувати операції 2...6, причому основним елементом вже не можна вибрати елемент ji рядка, в якому вже був при попередніх Жорданових виключеннях використаний елемент. Операції 2...6 продовжують мати, поки усі позначення рядків не будуть замінені позначеннями стовпців, тобто поки всі стовпці крім b_i -го не будуть вилучені.

Шуканими елементами будуть елементи, які залишаться після всіх обчислень в рядках навпроти нових позначень даних рядків. Оскільки нові позначення рядків відповідають відповідним невідомим та елементи навпроти, будуть відповідати розв'язкам заданої системи. З обчислення випливає, що шукані невідомі опиняються в стовпці під позначенням стовпця вільних елементів.

З даного методу обчислення помітно, що для розв'язку система повинна мати одинакову кількість рядків і невідомих, бо в протилежному випадку невідомі буде важко чи навіть неможливо знайти (коли невідових більше ніж рядків) даним методом.

Розв'язуючи систему вручну методом Жордана-Гауса, основним елементом зручно вибирати число на яке найменьше ділити і, зрозуміло, що неможливо вибирати 0. для полегшення розв'язку при обчисленні кожен рядок зокрема можна скорочувати.

Оскільки комп'ютер не має логічного мислення, то йому важко задати вибирати зручніший елемент. Тому йому в програмі можна задати, щоб він вибрав перший можливий елемент для основного елемента.

Розв'яжемо систему:

$$\begin{cases} x_1 - x_2 + x_3 + x_4 + x_5 = -1, \\ -x_1 - x_2 + 2x_3 + x_4 = -4, \\ -2x_3 - 2x_4 - x_5 = 2, \\ -3x_1 + x_2 - x_4 - 2x_5 = -2, \end{cases}$$

Якій відповідає матриця:

$$A = \begin{pmatrix} 1 & -1 & 1 & 1 & 1 \\ -1 & -1 & 2 & 1 & 0 \\ 0 & 0 & -2 & -2 & -1 \\ -3 & 1 & 0 & -1 & -2 \end{pmatrix}$$

та вектор правої частини:

$b = (-1, -4, 2, -2).$

Наведемо текст програми повністю, опускаючи не суттєві обрахунки:

$$A := \begin{bmatrix} 1 & -1 & 1 & 1 & 1 \\ -1 & -1 & 2 & 1 & 0 \\ 0 & 0 & -2 & -2 & -1 \\ -3 & 1 & 0 & -1 & -2 \end{bmatrix}$$

```
> b:= vector([-1, -4, 2, -2]);
```

$$b := [-1, -4, 2, -2]$$

```
> restart;with(linalg):
> m:=4:n:=5:
> A:=matrix(m,n,[[1, -1, 1, 1, 1], [-1, -1,
> 2, 1, 0], [0, 0, -2, -2, -1],
> [-3, 1, 0, -1, -2]]);
```

```
> Ae:=concat(A,b):
> Hstr:=vector(n+2):
>  for i from 1 to n do
>   Hstr[i+1]:=x[i];
>  end do:
>  Hstr[1]:=‘ ‘:
>  Hstr[n+2]:=1:
> Lcol:=vector(m): for k from 1 to m do
>  Lcol[k]:=‘ ‘:
> end do:
> Maketab:=proc() local At;
> global Ae,Lcol,Hstr;
>  At:=Ae; At:=concat(Lcol,At):
>  At:=stackmatrix(Hstr,At);
>  print(At);
> end proc:
> Maketab();
```

$$\begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & 1 \\ 1 & -1 & 1 & 1 & 1 & -1 \\ -1 & -1 & 2 & 1 & 0 & -4 \\ 0 & 0 & -2 & -2 & -1 & 2 \\ -3 & 1 & 0 & -1 & -2 & -2 \end{bmatrix}$$

```
> Gauss_Jordan:=proc(A::evaln,k,r)
> local i,j,z,zag;
> global m,n,Lcol,Hstr;
> z:=A[k,r]:
> for j to n+1 do
>   A[k,j]:=A[k,j]/z;
> end do;
> for i from 1 to m do
>  if i <> k then
>    z:=A[i,r];
>    for j from 1 to n+1 do
>      A[i,j]:=A[i,j]-A[k,j]*z;
>    end do;
>  end if;
> end do:
> A[k,r]:=1:
> print(k,r);
> end:
```

```
> SYS_solve:=proc(m,n)
> local k,r,i,zag,p,fin,num;
> global Lcol,Hstr,Ae;
> k:=1; r:=1; fin:=true;
> while fin do
>  p:=Ae[k,r];num:=k;
>   for i from k+1 to m do
>    if abs(abs(Ae[i,r]-1))<abs(abs(p)-1) then
>      p:=Ae[i,r]; num:=i;
>    end if;
>   end do;
>   if abs(p)>1e-7 then
>     if num<>k then
>       swaprow(Ae,num,k);
>     end if;
>     Gauss_Jordan(Ae,k,r);
>     Maketab();
>     k:=k+1; r:=r+1;
>   else
>    r:=r+1;
>   end if;
>   if (k>m) or (r>n) then
>     fin:=false;
>   end if;
> end do;
> end proc:
> SYS_solve(m,n):
```

$$\begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & 1 \\ 1 & 0 & 0 & \frac{1}{2} & \frac{3}{4} & 1 \\ 0 & 1 & 0 & \frac{1}{2} & \frac{1}{4} & 1 \\ 0 & 0 & 1 & 1 & \frac{1}{2} & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Нагадаємо, що вільні коефіцієнти записуються в таблицю з вихідними знаками. Опрацьована матриця передається в процедуру Gauss_Jordan і повертається як результат, тому відповідний формальний параметр A використовується для передавання даних в процедуру. Наступна процедура SYS_solve по черзі виключає змінні з рівнянь системи, вибираючи на стовпці елемент, найближчий до одиниці. При виклику процедури SYS_solve виконуються три кроки виключення (рівні рангу основної матриці системи), в результаті чого отримуємо остаточну таблицю. Система сумісна, оскільки вільний коефіцієнт в залишився в рядку = 0. Виписавши рівняння системи, відповідні таблиці, отримаємо наступні рівності

$$\begin{aligned} x_1 &+ \frac{1}{2}x_4 + \frac{3}{4}x_5 = 1, \\ x_2 &+ \frac{1}{2}x_4 + \frac{1}{4}x_5 = 1, \\ x_3 &+ x_4 + \frac{1}{2}x_5 = -1, \end{aligned}$$

які легко розв'язуються щодо змінних x1, x2, x3 і отримати розв'язок системи в параметричному вигляді з вільними змінними x4, x5.

Завдання

Розв'язати систему рівнянь відповідно до свого варіанта.

ПРАКТИЧНА РОБОТА 11

Функції користувача. Процедури.
Програмування символьних операцій.

МЕТА І ЗАВДАННЯ РОБОТИ

Мета роботи – засвоєння синтаксису функції користувача, процедури, програмування символьних операцій в середовищі Maple.

Задача роботи – практичне вивчення роботи вказаних операторів Maple.

ТЕОРЕТИЧНІ ВІДОМОСТІ

1. Функції користувача.

Існує значна відмінність між привласненням змінній виразу та функції Вираз є подібний до комбінації змінних і може виглядати приблизно так:

```
> Fn1:=cosh(x^3)/(1+x^2); # Тут cosh(z) - гіперболічний косінус змінної z, який є однією з функцій ядра Maple
```

$$Fn1 := \frac{\cosh(x^3)}{1+x^2}$$

Значення змінної F_j є саме виразом, але не є функцією. Maple не зрозуміє звертання до змінної Fj як до функції у вигляді $Fn_j(x)$, наприклад:

```
> Fn1(1.5);
```

$$\frac{\cosh(x^3)(1.5)}{1+\mathrm{x}(1.5)^2}$$

Як можна побачити, обчислення не відбулося. Для отримання бажаного результату необхідно привласнити x значення 1.5 подібно до того зроблено нижче у командному рядку, а далі запустити програму обчислення значення виразу Fn1 за умови, що x =1.5

```
> x:=1.5; Fn1;
```

$$x := 1.5$$

$$4.501307985$$

Однак це не завжди зручно.

Функція користувача у Maple може задаватися *спрощеним* синтаксисом:

```
> Fn2:=x- >cosh(x^3)/(1+x^2);
```

$$Fn2 := x \to \frac{\cosh(x^3)}{1+x^2}$$

Ця конструкція, створена за допомогою оператора-стрілки (яка, у свою чергу є комбінацією двох знаків: мінус « - « та знаку « > « (->). Вона означає, що значення x необхідно замінити на cosh(x^3)/(1+x^2). І це перетворення привласнене змінній (ідентифікатору) Fn2, яка і стає ім’ям цієї функції.

```
> Fn2(1.5); Fn2(1); Fn2(1/3);
```

$$4.501307985$$

$$\frac{1}{2}\cosh(1)$$

$$\frac{9}{10}\cosh\left(\frac{1}{27}\right)$$

Зверніть увагу, що не можна простим привласненням за допомогою оператора «:=« створити функцію користувача

```
> restart:
Fn1(x):=cosh(x^3)/(1+x^2);
Fn1(3);
```

$$Fn1(x) := \frac{\cosh(x^3)}{1+x^2}$$

$$Fn1(3)$$

Також можна визначити й функцію декількох аргументів у спрощеному синтаксисі (за допомогою оператора-стрілки):

```
> Fn3:=(x,y,z)->x*y^2*z^3;
```

$$Fn3 := (x, y, z) \rightarrow x\, y^2\, z^3$$

```
> Fn3(1,2,3);
```

$$108$$

Ще один спосіб завдання функції користувача базується на застосуванні змінній (ідентифікатору) Fn3, яка і стає ім’ям цієї функції.

оператора-функції **unapply:**

<ім’я>**:= unapply**(<вираз>, <змінна 1>, <змінна 2>,...)

Тут <вираз> – вираз, яким задається тіло функції, а через коми перераховані змінні (<змінна 1>, <змінна 2>,...), від яких залежить функція. Нижче дані приклади такого способу завдання функції користувача:

```
> restart;
```

```
Fn4:=unapply(sqrt(x^2+y^2),x,y);
Fn4(3,4);
```

$$Fn4 := (x, y) \rightarrow \sqrt{x^2 + y^2}$$

$$\sqrt{25}$$

```
> Fn5:=unapply(cosh(x^3)/(1+x^2),x);
Fn5(1.5);
```

$$Fn5 := x \rightarrow \frac{\cosh(x^3)}{1+x^2}$$

$$4.501307985$$

Оператор **unapply** на відміну від -> дає можливість створювати функції користувача з вже готових привласнених виразів. Порівняйте

```
> restart:
f:=cosh(x^3)/(1+y^2):
f1:=unapply(f,x,y);
f2:=(x,y)->f;
```

$$f1 := (x, y) \to \frac{\cosh(x^3)}{1 + y^2}$$

$$f2 := (x, y) \to f$$

```
> f1(1,2);
f2(1,2);
```

$$\frac{1}{5}\cosh(1)$$

$$\frac{\cosh(x^3)}{1 + y^2}$$

У Maple існує ще один оператор створення функції користувача. Це оператор **define:**

define(<оператор>, <властивість 1>, <властивість 2>,...)

Тут <оператор> – ім'я оператора, що будується, <властивість 1>,

<властивість 2> і т.д. - найменування властивостей функції, які можуть бути, наприклад: **binary** – бінарний оператор, **diff** – диференціальний оператор, **linear** – лінійний оператор та ін., або значення у вигляді fx) = <значення>.

Головна різниця між оператором **define** та іншими операторами, що створюють функції користувача, полягає у тому, що оператор **define** дозволяє створювати функції у неявному або рекурентному вигляді, задаючи тільки їх властивості, що дуже корисно при розв'язку диференційних рівнянь.

Наприклад:

```
> restart:
define(Ll,linear);
Ll(y**2+3*z+4);
define(r,r(0)=1,r(1)=1,r(2)=2,r(n::posint)=r(n-1)*r(n-2)):
r(7);
```

$$Ll(y^2) + 3\,Ll(z) + 4\,Ll(1)$$

$$256$$

2. Процедури.

Процедурою називають модуль програми, що має самостійне значення і виконує одну або декілька операцій, звичайно достатньо складних і відмінних від операцій, виконуваних вбудованими операторами і функціями. Процедури є важливим елементом структурного програмування і служать засобом розширення можливостей системи Maple користувачем. Кожна процедура має своє унікальне ім'я і список параметрів (він може бути й порожнім).

Процедури викликаються так само, як вбудовані функції – вказівкою їх імені із списком фактичних параметрів. При цьому просто процедури звичайно не повертають яких-небудь значень після свого виконання, хоча можуть привласнювати значення вхідним до них змінним.

Процедури-функції у відповідь на звернення до них повертають деяке значення. Вони практично є функціями користувача з більш складною структурою.

Найпростіша форма завдання процедури наступна:

<ім'я>:=**proc(**параметри) <тіло процедури> **end proc;**

Параметри процедури задаються переліком імен змінних, наприклад **proc**(x) або **proc**(x,y,z). За допомогою оператора «**::**» після імені змінної *можна* визначити її тип, наприклад, в оголошенні **proc**(n**::Integer)** оголошується, що змінна *n* є цілочисельною.

Процедури викликаються виразом вигляду:

name(Фактичні параметри)

Фактичні параметри підставляються на місце формальних. Невідповідність фактичних параметрів типу заданих змінних звичайно веде до повідомлення про помилку і до відмови від виконання процедури.

Як приклад нижче наведена процедура **mdc** обчислення модуля комплексного числа *z* – в даному випадку це єдиний параметр процедури:

```
> mdc:=proc(z::complex) evalf(sqrt(Re(z)^2+Im(z)^2)) end proc;
      mdc := proc (z::complex) evalf( sqrt( ℜ(z)^2 + ℑ(z)^2 ) ) end proc
```

Тепер для обчислення модуля комплексного числа *z* достатньо задати звернення до процедури **mdc***(z)*, вказавши замість *z* конкретне комплексне число, наприклад для :

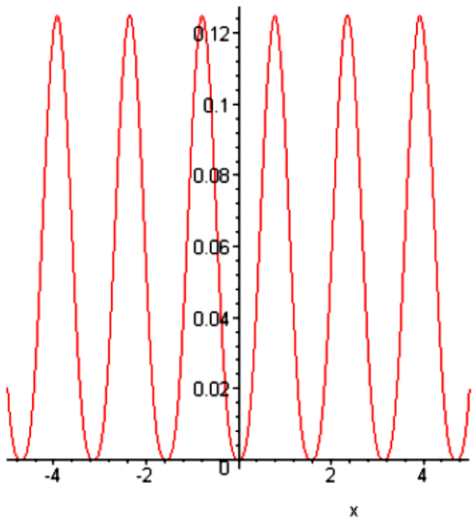
```
> z:=3+4*I; mdc(z); mdc(1+2*I);
                                  z := 3 + 4 I
                                       5.
                                  2.236067977
```

однак

```
> mdc("a");
Error, invalid input: mdc expects its 1st argument, z, to be of type complex, but received a
```

У процедурі можуть міститися будь-які складні операції і модулі. Головною перевагою застосування процедур є те, що вони є джерелом даних для побудови графіків.

```
> restart:
f:=proc(a)
sin(a)*cos(a):
abs(%^3)
end proc:
plot(f(x), x=-5..5);
```



3. Програмування символьних операцій

Дуже корисною властивістю системи Маріє є те, що вона дозволяє виконувати будь-яку кількість послідовних дій, тобто придатна для програмування. Для прикладу, розглянемо реалізацію знаходження коренів методом ділення відрізку навпіл.

Як відомо, цей метод полягає у тому, що відрізок, що містить єдиний корінь ділиться навпіл і з двох відрізків обирається той, який має різні знаки функції $f(x)=0$ на своїх кінцях, тобто містить корінь. Процедура, що реалізує знаходження коренів, виглядає наступним чином:

```
> restart:
HS:=proc(eq,x,xs,xe,e)
#Задаємо внутрішні змінні
local Function, Xstart, Xend, pres, Xmiddle, result, n, i::integer;
Xstart:=xs*1.:
Xend:=xe*1.:
pres:=e*1.:
# Розраховуємо кількість повторень
n:=ceil(ln((Xend-Xstart)/pres)/ln(2)):
#Створюємо функцію
Function:=unapply(eq,x):
#Задаємо цикл
for i from 1 to n do
# Ділимо відрізок навпіл
Xmiddle:=(Xstart+Xend)/2:
# Перевіряємо попадання у корінь
if evalb(Function(Xmiddle)*Function(Xstart)=0) then
i:=n
# Перевіряємо наявність кореня у лівому відрізку
elif evalb(Function(Xmiddle)*Function(Xstart)<0) then
Xend:=Xmiddle
else
# Якщо не попали у корінь і він не у лівому відрізку, то - у правому
відрізку.
Xstart:=Xmiddle
fi
od:
Xmiddle;
end:
```

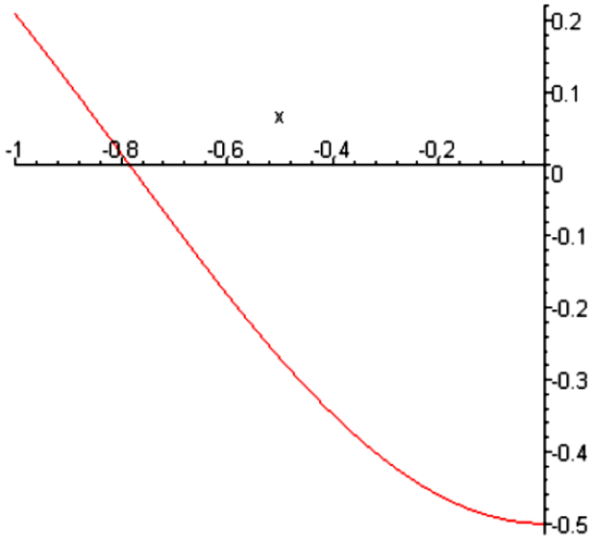
Тепер, якщо задати розв’язуване рівняння, точність розрахунків та інтервал пошуку, то можна одержати результат:

```
> eq:=sin(x)^2-0.5:
F:=evalf(HS(eq,x,-1.,0.,0.001));
```

$$F := -0.7853981634$$

Перевіримо графічно:

```
> plot(eq, x=-1..0);
```



Завдання

1. Розробити функцію користувача у загальному вигляді за допомогою оператора **define** та наочно показати її властивості.

2. Розробити функцію користувача визначеного типу та побудувати її графіки. Елементарні функції, що повинні входити до функції користувача обрати з таблиці:

Варіант/ Вираз	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	Т	Мо	Ст	ф	С	Т	Т	К	Мо	Т
2	К	Л	Е	Мо	ф	К	С	Е	Л	ф
3	Л	С	Т	Ст	К	Л	До	Т	Ст	До
Варіант/ Вирайз	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
1	Л	С	Л	К	Мо	Т	Е	Ст	Мо	Е
2	Ст	Т	ф	Мо	Л	К	ф	Л	Е	С
3	К	До	До	Ст	До	До	До	Мо	Ст	Т

До - добуток, Е - експонента, К - корінь, Л - логарифм, Мо - модуль, С - сума, Ст - ступеневий, Т - тригонометричний, Ф - факторіал.

4. Контрольні питання

1. Яка різниця між оператором-стрілкою, **unapply** та **define?**
2. Чим відрізняється процедура від функції користувача?

ПРАКТИЧНА РОБОТА № 12

Тема: Поняття об’єкта. Основні властивості об’єктів. Інкапсуляція. Класи. Модулі. Спадкування. Форми спадкування. Поліморфізм.

Мета: Одержати навики проектування і реалізації об’єктних типів та відношення спадкування класів.

Основні поняття: Об’єктний тип (клас), атрибут і властивість класу, операція і метод класу. Опис класів і використання об’єктів-екземплярів класів. Відношення між об’єктними типами (класами). Спадкування. Батьківські і дочірні класи. Пошук методу. Перевизначення методів. Поліморфізм. Статичні і динамічні (віртуальні) методи. Форми спадкування. Класи і підтипи.

Теоретичні відомості

Ключові поняття ООП – клас та об’єкт. Клас – це тип, який описує роботу об’єктів – екземплярів. Клас можна порівняти з кресленням, згідно з яким створюються об’єкти. Звичайно класи проектують таким чином, щоб їх об’єкти відповідали об’єктам предметної області. Об’єкт поряд з поняттям «клас» є важливим поняттям об’єктно-орієнтованого підходу в програмуванні. Під об’єктом розуміємо деяку сутність у віртуальному просторі, яка володіє визначеним станом і поведінкою. Як правило, при розгляді об’єктів виділяється те, що об’єкти належать одному чи декільком класам, які, у свою чергу, визначають поведінку (є моделлю) об’єкта. Об’єкт як структура даних являє собою іменований програмний компонент, який містить власні локальні дані й здатний виконувати певні дії над ними.

Основними властивостями об’єкта є:

1. Інкапсуляція – об’єднання атрибутів із процедурами і функціями, що працюють з цими даними.

2. Спадкування – завдання об’єкта, використання його для побудови ієрархії похідних об’єктів із спадкуванням доступу кожного з похідних об’єктів до коду і даних батьківського об’єкта.

3. Поліморфізм – завдання одного імені дії, що передається нагору і вниз по ієрархії об’єктів, із реалізацією цієї дії засобом, що відповідає кожному об’єкту в ієрархії.

Модулі

Модулі надають мові програмування Maple деякі властивості мов об’єктно-орієнтованого програмування. Вони служать для реалізації абстрактного типу даних на основі інкапсуляції – об’єднання даних і процедур їх опрацювання. Модулі задаються ключовим словом module з порожніми дужками () і завершуються словами end module або просто **end**:

name := module()

export eseq; local Iseq; global gseq:
option optseq: description desc:
Зміст модуля
end module (чи просто end)

Хоча структура модуля багато в чому нагадує структуру процедури, включаючи опис локальних та глобальних змінних, параметрів та описів, між ними є істотна різниця:

• модуль не має списку вхідних параметрів;
• в модулі можуть розміщуватися дані;
• модулі можуть використовуватися для створення пакетів процедур, доступ до яких забезпечується командою ***with***;
• модулі мають властивості у вигляді локальних змінних і методи у вигляді процедур інтерфейсу модулів;
• реалізація абстрактних типів даних за допомогою модулів прихована від користувача;
• модулі можуть містити оператор ***export eseq***, що описує вихідні змінні модуля;
• для доступу до вихідних змінних модуля може використовуватися спеціальний оператор «:-» (двокрапка і мінус);
• модулі та процедури можуть використовувати один в одного без обмеження рівня вкладеності;
• модулі можуть мати спеціальні конструктори об'єктів.

Інкапсуляція.

Припустимо, що наші практичні інтереси лежать в області побудови зображень тіл зоряного неба в двомірній площині. Основою всякого зображення є положення (позиція) окремого елемента на екрані, описаного координатами X і Y. Для того щоб задати двомірну позицію підходить тип запису.

```
pozition := module( )
 export X, Y;
 X, Y : Integer;
 end;
    module( ) export X, Y;  end module        (1)
```

Що можна робити з парою координат (X,Y)?

По-перше, може знадобитися задати значення координат (у програмуванні така процедура зветься ініціалізація):

```
Init := proc(CoordX, CoordY : : Integer)
 local X, Y : : Integer;
 X := CoordX :
 Y := CoordY :
 end;
proc(CoordX, CoordY::Integer)                    (2)
     local X, Y::Integer;
     X := CoordX; Y := CoordY
end proc
```

По-друге, може знадобитися знання фактичних значень координат, для цього вводимо дві функції:

```
fnGetX := X
                    X                            (3)
fnGetY := Y
                    Y                            (4)
```

Процедура Init і функція GetX і GetY повинні працювати з атрибутами модуля Pozition.

Введення об’єктів у СКМ Maple дозволяє зафіксувати це положення, оголосивши атрибути і дії в одному місці:

```
pozition := module( )
 export X, Y;
 fnGetX : Integer;
fnGetY : Integer;
 end;
   module( ) export X, Y;  end module            (5)
```

Тепер для ініціалізації типу Pozition досить викликати його модуль:

```
FirstPozition : pozition;
                  pozition                       (6)
fnGetX(FirstPozition);
fnGetY(FirstPozition);

              X(FirstPozition)
              Y(FirstPozition)                   (7)
```

Метод задається так само, як і процедура в модулі: в середині об’єкта записується заголовок, при цьому всі атрибути, які використовуються методом, повинні передувати його оголошенню. Визначення методу (розшифровування дій) відбувається поза оголошенням об’єкта, якому метод належить, супроводжуваним точкою.

```
Init := proc(CoordX, CoordY :: Integer)
 local X, Y :: Integer;
 X := CoordX:
 Y := CoordY:
 end;
proc(CoordX, CoordY::Integer)                    (2)
     local X, Y::Integer;
     X := CoordX; Y := CoordY
end proc
```

Примітка. Імена формальних параметрів методу не можуть збігатися з іменами атрибутів даних об’єкта.

Спадкування.

Продемонструємо приклад опису класу та його застосування:

```
> restart;
> Stack := module()
option package;
export ModuleApply, Push, Pop, Top;

 ModuleApply := proc() # Stack object constructor invoked with Stack(args)
                [args];
              end:
 Push := proc(s) [op(s),args[2..-1]] end:
 Pop := proc(s) if nops(s) > 1 then s[1..-2]; else [] end end:
 Top := proc(s) if nops(s) > 1 then s[-1] end end:
end module:
```

Опис класу (модуля)

```
> with( Stack );
```

[ModuleApply, Pop, Push, Top]

```
> a := Stack( L, M, N, O );
```

$$a := [L, M, N, O].$$

Зміст об’єктно-орієнтованого програмування полягає в роботі з атрибутами об’єкта через його методи. Далі наводяться приклади застосування методів класу до об’єктів.

```
 > Top( a );
s := Pop( a );
s := Pop( a );
s := Push( a, P );
```

O

s := [L, M, N]

s := [L, M, N]

s := [L, M, N, O, P]

```
> b := Stack();
b := Push( b, A, B );
```

b := [ ]

b := [A, B]

```
> c:=Stack(1,2,3,4);
```

c := [1, 2, 3, 4]

```
> Top(a);Top(b);Top(c);
```

O

B

4

Поліморфізм.

Розглянемо підтримку роботи внутрішніх програм із зовнішніми. Як відомо, Maple може експортувати свою функціональність. Тобто існує можливість виклику функцій Maple із зовнішніх програм. А також реалізована функція імпорту функціональності. Тобто Maple може виконувати функції, реалізовані в зовнішніх бібліотеках.

В обох випадках зовнішні програми (чи бібліотеки) можуть бути реалізовані із використанням цілого ряду технологій та мов програмування. Java технологія добре інтегрується із Maple, а також є однією із найкращих реалізацій парадигм ООП.

Експорт функціональності

Технологія OpenMaple – це набір функцій, що надають доступ до алгоритмів та структур даних Maple, скомпільованих на C, Java, чи Visual Basic-програмах.

Java OpenMaple являє собою інтерфейс між обчислювальним ядром Maple та Java- програмою. Цей інтерфейс реалізований з використанням Java-класів та Java-інтерфейсів. Класи дозволяють Java-програмам викликати ядро Maple. Інтерфейси використовуються для того, щоб ядро Maple викликало Java-класи або методи класів. Тобто призначення інтерфейсів – повернення результату до Java-програми для обробки та правильного виводу.

Слід зауважити, що інтерфейси використовуються тільки для отримання результатів після запиту класом функції ядра Maple і не можуть бути використані для оберненої задачі, тобто імпорту функцій в ядро Maple.

Імпорт функціональності

Команда ***define_external*** зв'язує функцію, описану в зовнішній бібліотеці, з ядром Maple та організовує інтерфейс між процедурами Maple і вказаною функцією. Функція може бути декларована в бібліотеці DLL у Windows, розподіленій бібліотеці в UNIX чи Java-класі.

Приклад роботи із класом, описаним на **Java:**

```
> restart;
> with(mapleoop):
> initooplib("c:/path/to/class/mylib.class");
> a:=Maclaurin(exp(x));
> a->int();
        exp(x)
> a->diff();
        exp(x)
> a->Expansion();
        1+1*x+1/2*x^2+1/6*x^3+1/24*x^4+1/120*x^5+O(x^6)
```

Контрольні завдання:

Створити опис класу для представлення комплексних чисел, який інкапсулює в себе відповідні операції та функції над комплексними числами і змінними. Скласти програму, яка демонструє роботу з даним класом. Програма повинна мати меню для перевірку всіх методів класу.

Короткі теоретичні відомості

Алгебраїчна форма запису комплексного числа:
$Z = A + jB = ReZ + jImZ$, де A – дійсна частина, а B – мнима частина комплексного числа Z, j – мнима одиниця.
Тригонометрична форма запису комплексного числа:
$Z=Re^{j\theta}$ де $R=\sqrt{A^2+B^2}$, $\theta=arctg(A/B)$.

Варіанти завдань:

1. Z_1+Z_2, де $Z_1=A_1+jB_1$, $Z_2=A_2+jB_2$
2. Z_1-Z_2
3. Z_1*Z_2
4. Z_1/Z_2
5. Z^2, де $Z=A+jB$
6. $1/Z$
7. $1/Z^2$
8. $\sqrt{Z}$
9. exp Z
10. ln Z
11. $\sqrt[n]{Z}$, де $n>0$
12. Z^n, де $n>=0$
13. sin Z
14. cos Z
15. tg Z.

Додаток Г

Конспект уроку на тему:

Формальне поняття алгоритму. Машина Тюрінга.

Мета уроку:

освітня:

- ознайомити учнів з поняттям алгоритмічно нерозв'язної задачі;
- розширити поняття алгоритму, навчити описувати будову машини Тюрінга і принципи її роботи, будувати машину Тюрінга для розв'язування найпростіших задач;
- формувати ключові компетентності;
- забезпечити закріплення умінь аналізувати, систематизувати, доводити.

розвиваюча:

- формувати навички роботи з комп'ютером;
- розвивати логічне мислення;
- розвивати алгоритмічне мислення, здатність до формалізації.

виховна:

- формувати культуру пізнавальної діяльності;
- формувати культуру колективної діяльності, виховувати почуття відповідальності за результати своєї праці, використовувані іншими людьми.

Тип уроку: урок вивчення нового матеріалу.

Обладнання і матеріали: підручник, комп'ютер, листки із завданнями для групової роботи.

ХІД УРОКУ

І. Організаційний момент. Перевірка домашнього завдання.

ІІ. Підготовка учнів до засвоєння нового матеріалу: повідомлення теми і цілей уроку, бесіда з учнями про відомі їм різні тлумачення поняття алгоритму.

III. Вивчення нового матеріалу.

Формальне (математично строге) визначення алгоритму ввели незалежно один від одного в 1936 році Алан Тюрінг і Еміль Пост. Мета створення Тюрінгом абстрактної уявної машини – отримання можливості доведення існування або не існування алгоритмів розв'язування різних задач.

Машина Тюрінга – це математичний апарат, створений для розв'язування певних завдань.

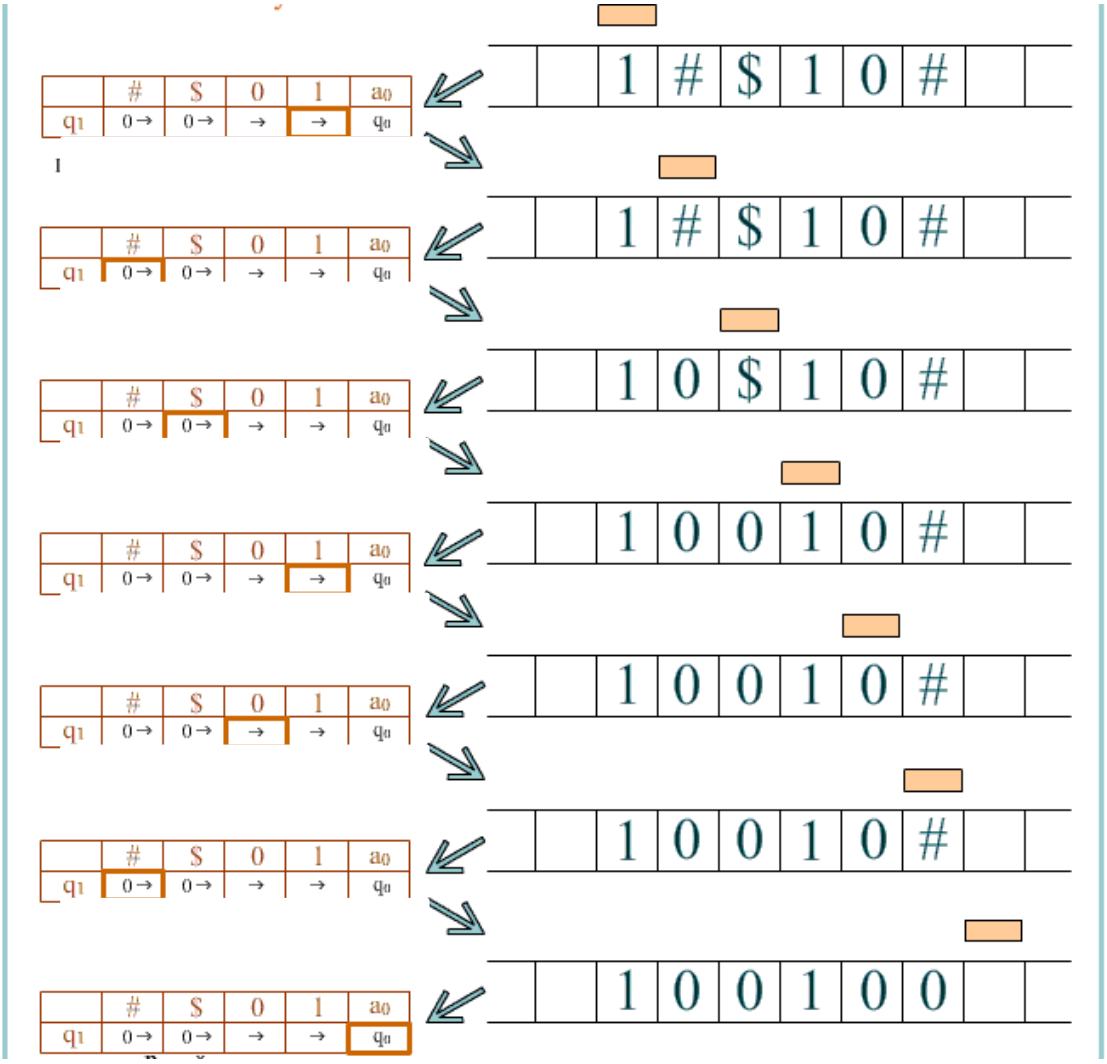
Опис машини Тюрінга

Нескінченна стрічка, розділена на комірки (запам'ятовувальний пристрій)	Автомат (головка зчитування/запису, керована програмою)
Два кінцевих алфавіти (для різних МТ можуть бути різними): 1. Алфавіт вхідних символів (зовнішній) $A = \{a_0, a_1, ..., a_m\}$ 2. Алфавіт станів (внутрішній) $Q = \{q_0, q_1, ..., q_p\}$	
Стан q_0 – пасивний (машина закінчила роботу) Стан q_1 – початковий (машина починає роботу) Комірка a_0 – порожня буква (ознака того, що комірка порожня) Комірка ***зупинки*** – комірка, в якій записано, що автомат повинен перейти в стан q_0 (дійшовши до неї, машина зупиняється).	

IV. Первинна перевірка засвоєння знань.

Демонстрація прикладу побудови машини Тюрінга.

Припустимо, на стрічці є слово, що складається з символів, кожен з яких співпадає з даним із символів #, $, 1 і 0. Потрібно замінити всі символи # і $ на нулі. У момент запуску головка знаходиться над першою зліва буквою слова. Завершується програма тоді, коли головка виявляється над порожнім символом після самої правої букви слова.



V. Закріплення знань.

Пропонуються завдання на реалізацію алгоритмів з допомогою машини Тюрінга. Потім групова робота з отриманими на листках завданнями для кожної групи. Після обговорення завдань у групах учні пропонують свої розв’язки завдань.

Приклад завдання для побудови машини Тюрінга:

Потрібно побудувати машину Тюрінга, яка додає одиницю до кожного числа на стрічці. Вхідне слово складається з цифр цілого десяткового числа, записаних в послідовні комірки на стрічці. У початковий момент машина знаходиться навпроти першої цифри числа справа.

Розв’язок.

					1	0	5	9	4										

Машина повинна додати одиницю до останньої цифри числа. Якщо остання цифра дорівнює 9, то її замінити на 0 і додати одиницю до попередньої цифри. Алгоритм для даної машини Тюрінга може виглядати так:

A \ Q	q0
a0	1S q0
0	1S q0
1	2S q0
2	3S q0
3	4S q0
4	5S q0
5	6S q0
6	7S q0
7	8S q0
8	9S q0
9	0← q0

У цій машині Тюрінга q_1 – стан зміни цифри, q_0 – стан зупинки. Якщо в стані q_l автомат бачить цифру 0..8, то він замінює її на 1..9 відповідно і переходить у стан q_0, тобто машина зупиняється. Якщо ж він бачить цифру 9, то замінює її на 0, зсувається вліво, залишаючись в стані q_0. Так продовжується до тих пір, поки автомат не зустріне цифру, меншу від 9. Якщо ж всі цифри були рівні 9, то він замінить їх нулями, запише 0 на місці старшої цифри, зміститься вліво і в порожній клітинці запише 1. Потім перейде в стан q_0, тобто зупиниться.

Приклад завдань для групової роботи

Завдання для групової роботи

1. Опишіть, який алгоритм виконує дана машина Тюрінга. Відомо, що в початковому стані автомат зчитує перший лівий символ вхідного слова.

	a_0	0	1
q_1	a_0Н!	1Пq_1	0Пq_1

2. Дано десятковий запис натурального числа n> 1. Розробіть машину Тюрінга, яка зменшувала б задане число n на 1. Автомат в стані q_1 зчитує першу праву цифру числа. Крім самої програми-таблиці опишіть словами, що виконується машиною в кожному стані.

q_1										

VI. Контроль і самоперевірка знань.

По черзі заслуховуються розв’язання завдань кожної групи, обговорюються результати. Підводяться підсумки виконаної роботи, заповнені робочі листи передаються вчителю.

VII. Підведення підсумків уроку.

1. Формулювання висновків про уточнення поняття алгоритму, його формалізацію.

2. Оголошення результатів уроку.

3. Домашнє завдання: придумати завдання і написати для його розв’язування машину Тюрінга.

4. Рефлексія:

- Що сподобалося на уроці, а що ні?
- Які знання про алгоритми ви застосовували на уроці?
- Скільки складових містить команда машини Тюрінга?
- Які труднощі ви відчули при розв’язуванні задач?

Додаток Д

Конспект уроку на тему:

Застосування циклічних структур для розв'язання задач

Мета: формування у школярів вмінь і навичок застосування циклічних структур для розв'язування задач та використання різних типів циклів, їх запису мовою графічних-схем і мовою програмування в СКМ Maple; виховання інформаційної культури.

Тип уроку: урок розв'язування задач та повторення вивченого.

ХІД УРОКУ

І. Організаційний момент. Повідомлення теми і мети уроку.

ІІ. Актуалізація опорних знань учнів.

Повторення загальних відомостей про цикли.

Загальний вигляд оператора циклу з передумовою:

Навчальна алгоритмічна мова:	**СКМ *Maple:***
Команда повторення – цикл «поки»:	
поки <*умова*>	while <*умова*>
пц	do <*P*>
<*P*>	end do
кц	

Тут логічний вираз приймає одне з двох значень **true** або **false,** P – це набір простих чи складених операторів. Цикл з передумовою працює за таким принципом: *повторення оператора P буде виконуватися доти, поки умова в операторі циклу отримує значення* true. *Якщо тільки на деякому кроці циклу умова набуде значення* **false,** *цикл припинить свою роботу.*

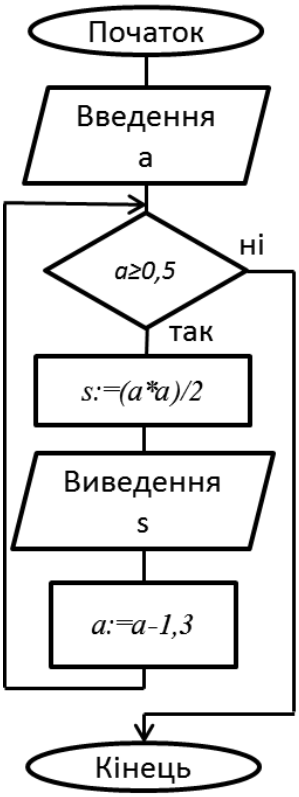
Оператор дуже простий і зрозумілий. Схема алгоритму оператора повторення з передумовою:

III. Розв’язування вправ.

Розглянемо ***приклад 1.***

Створити алгоритм і програму, за допомогою яких можна обчислити і вивести на екран величини площ прямокутних рівнобедрених трикутників, якщо відомо, що катет першого дорівнює a (a>2) см, катет кожного наступного зменшується на 1,3 см. Обчислення площ припиняється, коли катет трикутника стає меншим 0,5 см.

Блок-схема:



Текст програми мовою СКМ *Maple:*

```
s := 0 :
a := 6 :
while a ≥ 0.5 do s := a·a/2 ; print("s=", s); a := a − 1.3 : end do:
                    "s=", 18
                    "s=", 11.04500000
                    "s=", 5.780000000
                    "s=", 2.205000000
                    "s=", 0.3200000000
```

Загальний вигляд оператора циклу з параметром:

Навчальна алгоритмічна мова:	**СКМ *Maple:***
Команда повторення з параметром – цикл «для»:	
для *x* від $x_{поч}$ до $x_{кін}$ [крок $x_{крок}$] пц <*P*> кц	for <ім'я> from $x_{поч}$ by *крок* to $x_{кін}$ <*P*> end do

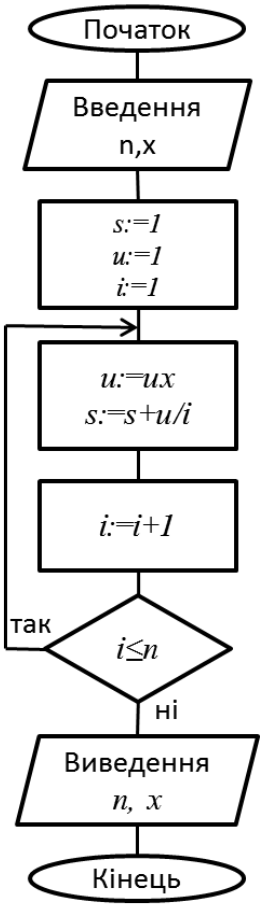
<ім'я> параметр циклу – змінна цілого типу, $x_{поч}$ – початкове значення параметра циклу, $x_{крок}$ – крок зміни параметра циклу, $x_{кінц}$ – кінцеве значення параметра циклу, *P* – простий чи складений оператор.

Службове слово **to** означає, що зміна значення параметра циклу йде від $x_{поч}$ до $x_{кінц}$ в порядку збільшення, а **downto –** в порядку зменшення.

Приклад 2. Розробіть алгоритм і програму обчислення виразу:

$$1 + x + \frac{x^2}{2} + \frac{x^3}{3} + \frac{x^4}{4} + \ldots + \frac{x^n}{n}$$

Блок-схема:



Текст програми мовою СКМ *Maple:*

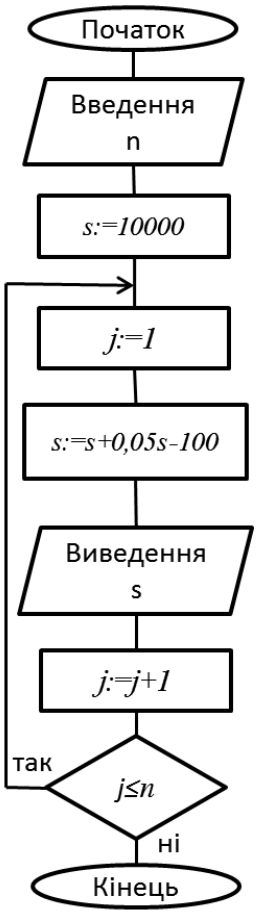
```
u := 1 :
n := 4 :
x := 3 :
S := 1 :
for i from 1 to n do u := u·x : S := S + u/i : end do:
print("сума =", S);
```

"сума =", $\frac{151}{4}$

Приклад 3.

Первинний вклад до банку становить 10000 грн. За кожний рік нараховується 5% річних і кожного року з накопиченої суми знімається 100 грн. Створіть алгоритм і програму визначення величини вкладу за кожний з перших n років.

Блок-схема:



Текст програми мовою СКМ *Maple:*

```
n := 6 :
s := 10000 :
for j from 1 to n do s := s + 0.05 · s − 100 : print("за", j, "-ий рік=", s)
    end do;
```

```
                    10400.00
        "за", 1, "-ий рік=", 10400.00
                    10820.0000
        "за", 2, "-ий рік=", 10820.0000
                    11261.00000
        "за", 3, "-ий рік=", 11261.00000
                    11724.05000
        "за", 4, "-ий рік=", 11724.05000
                    12210.25250
        "за", 5, "-ий рік=", 12210.25250
                    12720.76512
        "за", 6, "-ий рік=", 12720.76512
```

IV. Підсумок уроку.

Запитання для перевірки засвоєння знань

1. Що називається циклом?
2. Які типи циклів вам відомі?

V. Домашнє завдання.

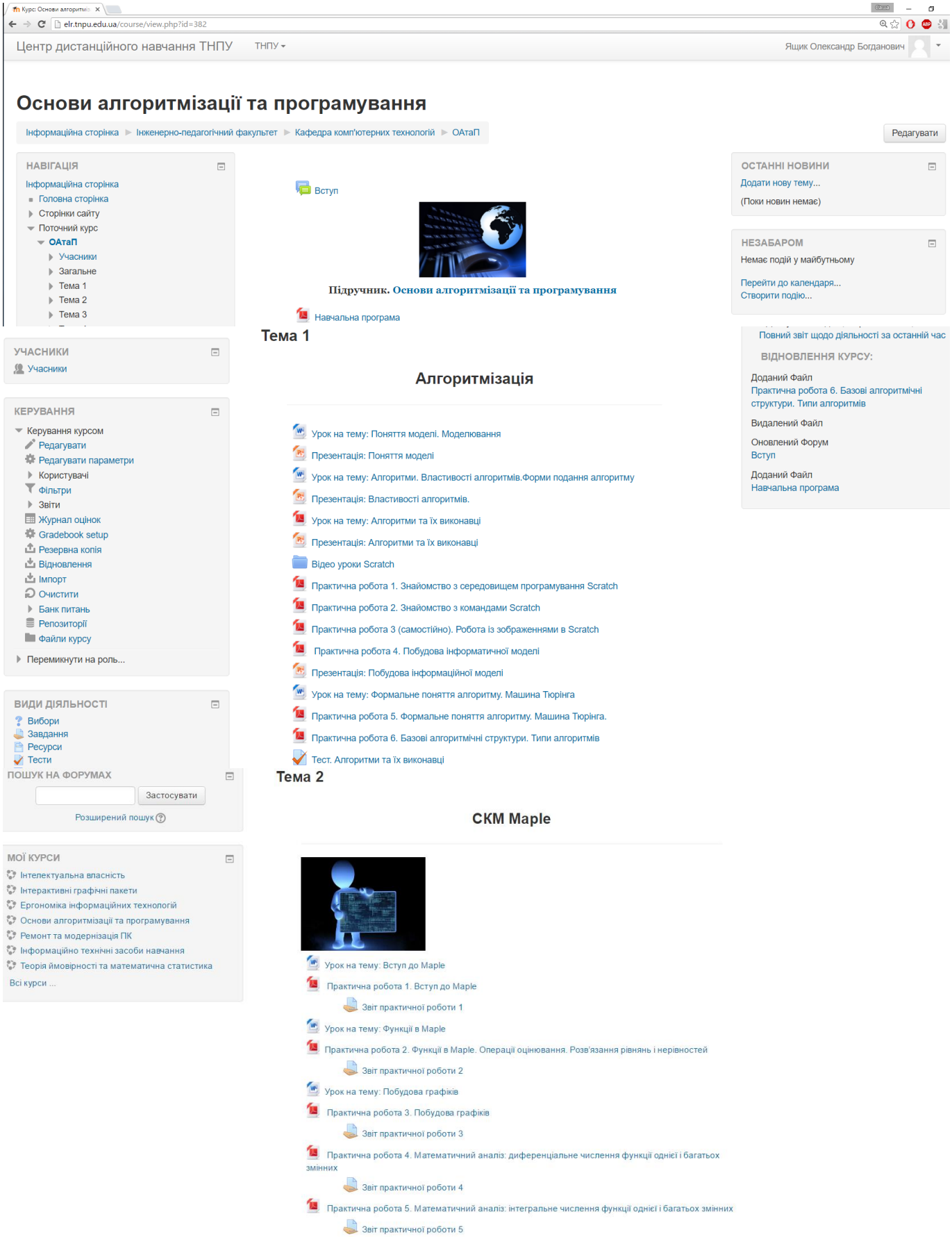
Вивчити теоретичний матеріал уроку.

1. Скласти програму, яка виводить на екран перших 50 парних чисел. Кожне число в окремому рядку.

2. Вивести на екран квадрати чисел від 1 до 10. Скласти програму в 2-х варіантах, використовуючи різні оператори циклу.

Додаток Е

Електронний навчальний курс (http://elr.tnpu.edu.ua/course/view.php?id=382)



Елементи програмування в Maple

Урок на тему: Застосування циклічних структур для розв'язання задач

Практична робота *6*. Умовні вирази, оператори циклу, векторизація циклів

Відео. Умовні оператори, розгалуження

Звіт практичної роботи *6*

Практична робота *7*. Лінійна алгебра

Відео. Розв'язання системи лінійних рівнянь

Звіт практичної роботи *7*

Урок на тему: Масиви, вкладені цикли

Практична робота *8*. Масиви, вкладені цикли, візуалізація масивів даних

Звіт практичної роботи *8*

Урок на тему: Метод Жордана-Гауса

Практична робота *9*. Вивчення функцій для опрацювання матриць. Метод Жордана-Гауса

Звіт практичної роботи *9*

Урок на тему: Функції користувача

Практична робота *10*. Функції користувача. Процедури. Програмування символьних операцій.

Звіт практичної роботи *10*

Практична робота *11*. Поняття об'єкта. Основні властивості об'єктів. Інкапсуляція. Класи. Модулі. Спадкування. Форми спадкування. Поліморфізм.

Звіт практичної роботи *11*

Тестовий контроль

Підсумкове тестування

Тема 4

Додаткові матеріали

Основи алгоритмізації та програмування

Задачі підвищеної складності із програмування

Розв'язання олімпіадних задач з програмування

Рекомендована література

Анкета

Додаток Є

Приклад контрольної роботи

Тема. Тематичне оцінювання по темі «Моделювання. Основи алгоритмізації».
Мета.

Навчальна. Перевірити знання, уміння та навички роботи учнів по темі «Моделювання. Основи алгоритмізації».

Розвиваюча. Розвивати логічне та алгоритмічне мислення.

Виховна. Виховувати культуру оформлення виконаної роботи.

Матеріали для роботи з учнями:Презентація.

План

1. Організаційний момент.
2. Контрольна робота.
3. Запитання до уроку.
4. Домашнє завдання.

Хід уроку

1. Організаційний момент.

Обговорення правил виконання, оформлення та здачі робіт.

2. Контрольна робота.

Варіант 2

(1-4 балів)

1. Від чого пішло слово “алгоритм”?
а) від назви країни; б) імені вченого;
в) назви науки; г) імені грецького бога.

2. Яку властивість не повинен мати алгоритм?
а) зрозумілість; б) раціональність;
в) дискретність; г) скінченність.

3. Який етап входить в етапи розв’язування задач з використанням ЕОМ?
а) створення макету програми; б) створення алгоритму;
в) аналіз формули; г) отримання умови задачі.

4. Напишіть загальний вигляд команди розгалуження?

(5, 6 балів)

5. Напишіть алгоритм розрахунку S по формулі: $S = \frac{3x+1}{3}$. Значення x вводить людина.

(7-9 балів)

6. Напишіть алгоритм та намалюйте його блок-схему. Задається число F. Розрахувати та вивести корені чисел від 1 до F.

(10,11 балів)

7. Напишіть алгоритм та намалюйте його блок-схему. Задається A. Розрахувати та вивести X.

$$X = \begin{cases} A^2 + A, & A>100 \\ A^5 + 10A, & A<100 \end{cases}$$

(12 балів)

8. Скласти та записати алгоритм наповнення відра водою за допомогою стакана.

Варіант 1

(1-4 балів)

1. Від чого пішло слово “алгоритм”?
а) від назви країни; б) імені вченого;
в) назви науки; г) імені грецького бога.

2. Яку властивість повинен мати алгоритм?
а) цілеспрямованість; б) раціональність;
в) простоту; г) скінченність.

3. Який етап не входить в етапи розв’язування задач з використанням ЕОМ?
а) створення програми; б) створення алгоритму;
в) аналіз результатів; г) отримання умови задачі.

4. Напишіть загальний вигляд команди повторення з параметром.

(5, 6 балів)

5. Напишіть алгоритм розрахунку У по формулі: $У = \frac{x+2}{5}$. Значення х вводить людина.

(7-9 балів)

6. Напишіть алгоритм та намалюйте його блок-схему. Задається число К. Розрахувати та вивести квадрати чисел від 1 до К.

(10,11 балів)

7. Напишіть алгоритм та намалюйте його блок-схему. Задається Р. Розрахувати та вивести К.

$$К = \begin{cases} \sqrt{P}, & P>0 \\ P^2, & P<0 \end{cases}$$

(12 балів)

8. Скласти та записати алгоритм наповнення відра водою за допомогою стакана.

3. Запитання до уроку.
Обговорення питань, які викликали найбільше труднощів при виконання контрольної роботи.

4. Домашнє завдання.
Скласти кросворд на 12 термінів до теми.

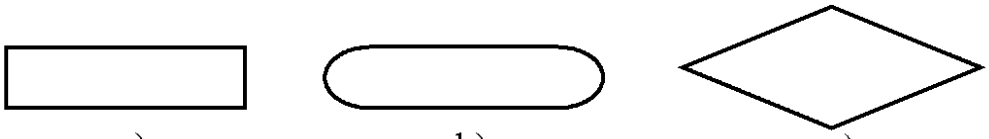
Додаток Ж

Тестові завдання

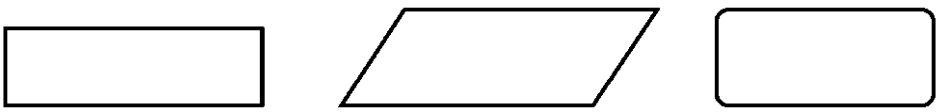
1. ***Що таке*** алгоритм?
   a) Це чітко визначена для конкретного виконавця послідовність дій, які спрямовані на досягнення поставленої мети або розв'язування задачі певного типу.
   b) Послідовність дій, яка дозволяє розв'язати певну задачу.
   c) Послідовність дій, яка записується на певній мові програмування і призначена для виконання на комп'ютері.
2. ***Перерахуйте*** властивості алгоритму.
   a) Скінченність, результативність, формальність, визначеність, масовість, зрозумілість, дискретність.
   b) Масовість, скінченність, визначеність, дискретність, правильність, зрозумілість.
   c) Скінченність, масовість, результативність, формальність, правильність, логічність, зрозумілість, визначеність.
3. В чому полягає така властивість як ***скінченність***?
   a) Виконання кожного алгоритму не повинно бути циклічним.
   b) Виконання кожного алгоритму повинно завершуватись за скінчене число кроків.
   c) Виконання кожного алгоритму повинно скінчитися тільки при отриманні задовільного результату.
4. В чому полягає така властивість як ***результативність?***
   a) Виконання алгоритму завжди повинно приводити до певного результату.
   b) Виконання алгоритму повинно завершитися тільки при отриманні задовільного результату.
   c) Виконання алгоритму завжди повинно приводити до певного результату (можливо, негативного). Воно не може закінчуватись невизначеною ситуацією або ж не закінчуватись взагалі.

5. В чому полягає така властивість як ***формальність?***
   a) Виконавець повинен вникнути в суть алгоритму. Ця властивість має особливе значення для автоматизації виконання алгоритмів.
   b) Комп'ютер повинен розуміти, що він робить. Інакше алгоритм не має значення.
   c) Виконавець відповідно до алгоритму повинен одержати результат, не вникаючи в його суть. Ця властивість має особливе значення для автоматизації виконання алгоритмів. Комп'ютери на можуть розуміти суть завдань і окремих вказівок алгоритмів.
6. Яким чином можна ***описати алгоритм?***
   a) За допомогою слів, спеціальних мов, використовуючи математичні формули, таблиці, графіки, блок-схеми та інші засоби.
   b) За допомогою мов програмування, блок-схем, математичних формул, графіків.
   c) За допомогою спеціальних мов, спеціальних графічних блоків, математичних формул, таблиць.
7. В чому полягає така властивість як ***визначеність?***
   a) В алгоритму всі дії повинні бути визначеними.
   b) В алгоритмі повинні бути вказівки, які зрозумілі виконавцеві.
   c) Будь-який алгоритм повинен бути описаний так, щоб при його розшифруванні у виконавця не виникло двозначних вказівок. Тобто різні виконавці згідно з алгоритмом повинні діяти однаково та прийти до одного й того ж результату.
8. В чому полягає така властивість алгоритму як ***масовість?***
   a) За допомогою складеного алгоритму повинні розв'язувати задачі будь-які виконавці.
   b) За допомогою складеного алгоритму повинен розв'язуватись цілий клас задач.
   c) Під масовістю алгоритму розуміється єдиність в тлумаченні правил виконання дій і порядок їх виконання.

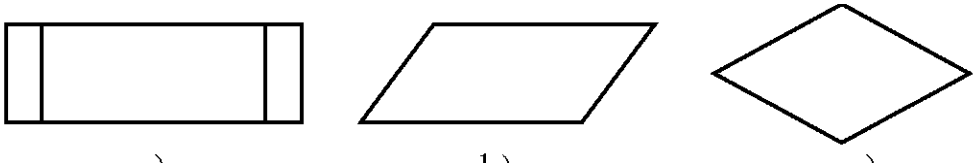
9. В чому полягає така властивість алгоритму як ***зрозумілість***?
   a) Обов'язкове завершення кожної дії, що складає алгоритм, а також завершеність виконання алгоритму в цілому.
   b) В алгоритмі повинні бути лише операції, які знайомі виконавцеві.
   c) Виконання кожного алгоритму повинно завершуватись за скінчене число кроків.
10. В чому полягає така властивість алгоритму як ***дискретність?***
   a) Можливість використовувати його для розв'язування подібних задач.
   b) В алгоритмі всі його команди чітко відокремлені одна від одної.
   c) Алгоритм написаний таким чином, що виконавець правильно сприйме кожну команду і зможе її виконати.
11. ***Що таке*** алгоритмічна мова?
   a) Система правил і позначень для однотипного запису алгоритмів та їх виконавця.
   b) Система засобів для розробки алгоритмів.
   c) Організована послідовність дій, призначена для певного виконавця.
12. Який з наведених блоків відповідає ***початку та кінцю алгоритму?***



13. Який з наведених блоків відповідає ***введенню та виведенню даних?***



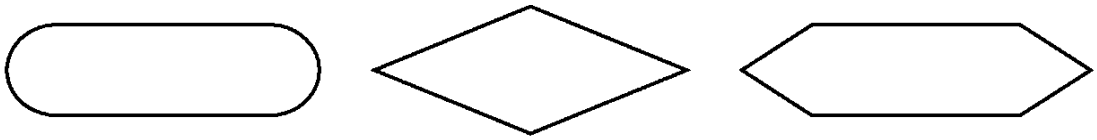
14. Який з наведених блоків використовується ***для перевірки умов?***



15. Який з наведених блоків використовується для ***звернення до підпрограм?***

a) b) c)

16. Який з наведених блоків використовується для запису ***циклу з параметрами?***



a) b) c)

17. Який з наведених блоків використовується ***при обчисленні виразів?***

a) b) c)

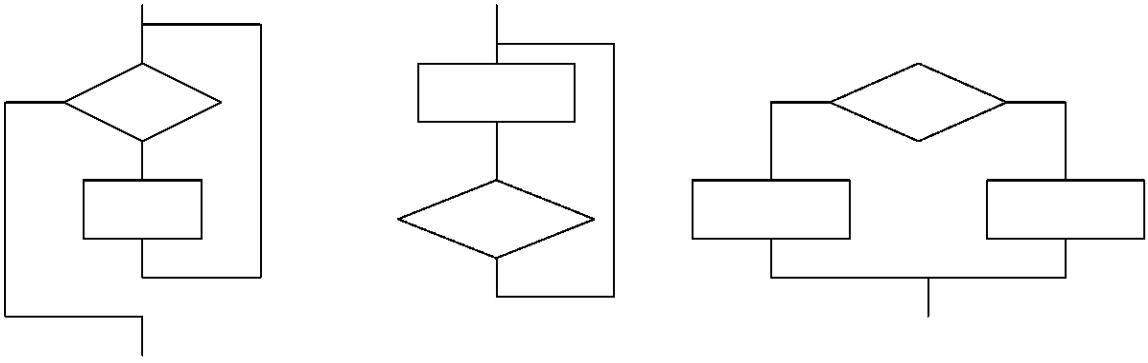
18. Перерахуйте ***базові структури алгоритмів*** (основні типи алгоритмів)?

a) Діалогові, допоміжні, лінійні.

b) Лінійні (слідування), розгалуження, циклічні.

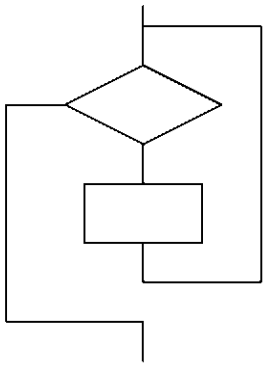
c) Лінійні, результати, аргументи.

19. Яка з наведених блок-схем відповідає базовій структурі алгоритмів - розгалуженню в повній формі?

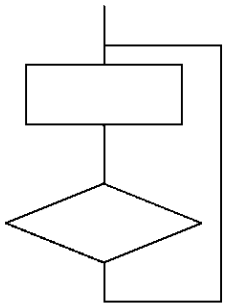


a) b) c)

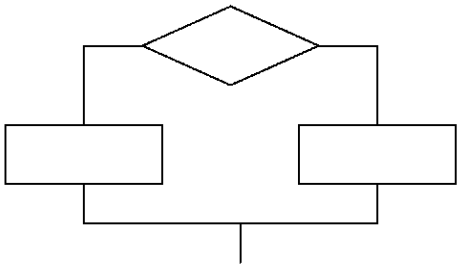
20. Яка з наведених блок-схем відповідає базовій структурі алгоритмів - ***циклу «ПОКИ»?***



a)

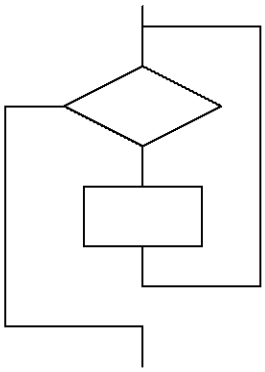


b)

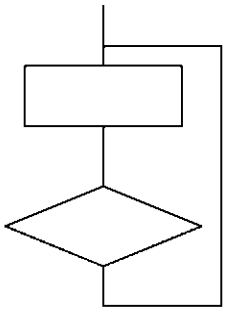


c)

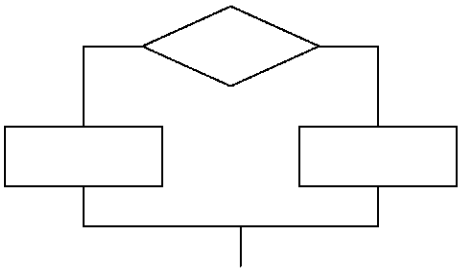
21. Яка з наведених блок-схем відповідає базовій структурі алгоритмів ***циклу «ДО»?***



a)

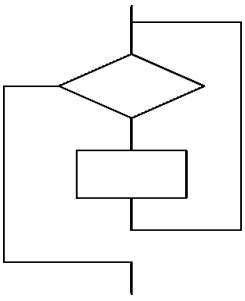


b)

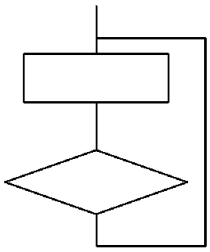


c)

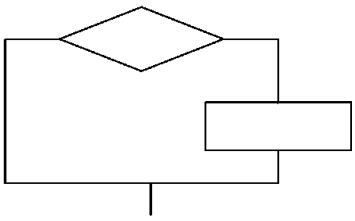
22. Яка з наведених блок-схем відповідає базовій структурі алгоритмів - розгалуженню в скороченій формі?



a)

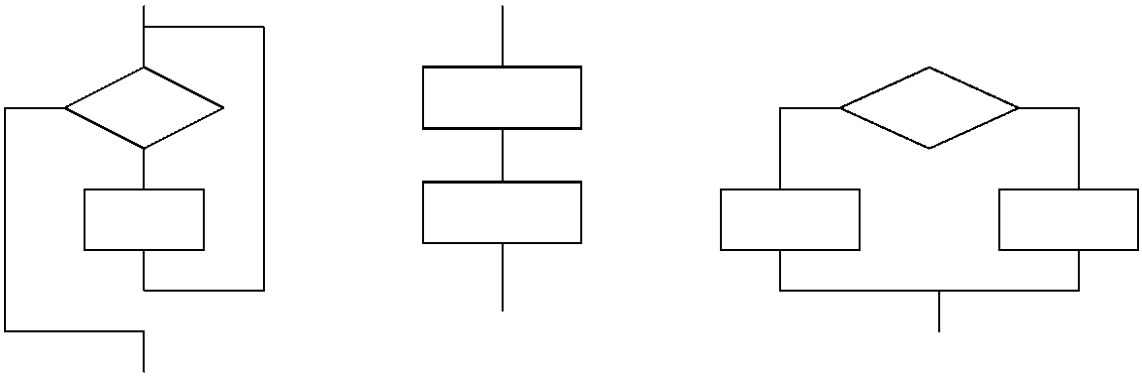


b)



c)

23. Яка з наведених блок-схем відповідає базовій структурі алгоритмів - ***лінійному алгоритму?***



a) b) c)

24. Які напрямки прийнято за основні напрямки виконання дій?
    a) Зверху вниз і зліва направо.
    b) Зверху вниз і справа наліво.
    c) Знизу вверх і зліва направо.
25. Як ви вважаєте, чи можна за допомогою базових структур алгоритмів зображати алгоритми будь-якої складності?
    a) Так.
    b) Ні.
26. Який алгоритм називається ***лінійним?***
    a) Це такий алгоритм, в якому вказівки виконуються послідовно відповідно до їх розміщення.
    b) Це такий алгоритм, в якому одні і ті самі вказівки (операції, оператори) виконуються багаторазово стосовно до різних значень змінних.
    c) Це такий алгоритм, в якому виконуються ті або інші вказівки залежно від результату перевірки деякої умови (або сукупності умов).
27. Який алгоритм називається ***розгалуженим?***
    a) Це такий алгоритм, в якому вказівки виконуються послідовно відповідно до їх розміщення.
    b) Це такий алгоритм, в якому одні і ті самі вказівки (операції, оператори) виконуються багаторазово стосовно до різних значень змінних.

c) Це такий алгоритм, в якому виконуються ті або інші вказівки залежно від результату перевірки деякої умови (або сукупності умов).

28. Який алгоритм називається ***циклічним?***

a) Це такий алгоритм, в якому вказівки виконуються послідовно відповідно до їх розміщення.
b) Це такий алгоритм, в якому одні і ті самі вказівки (операції, оператори) виконуються багаторазово стосовно до різних значень змінних.
c) Це такий алгоритм, в якому виконуються ті або інші вказівки залежно від результату перевірки деякої умови (або сукупності умов).

29. Який алгоритм називається ***рекурентним?***

a) Це такий алгоритм, в якому значення змінної в кожному циклі обчислюється через значення цієї ж змінної при попередньому виконанні циклу.
b) Це такий алгоритм, в якому одні і ті самі вказівки (операції, оператори) виконуються багаторазово стосовно до різних значень змінних.
c) Це такий алгоритм, в якому виконуються ті або інші вказівки залежно від результату перевірки деякої умови (або сукупності умов).

30. Які ***методи сортування*** ви знаєте?

a) Методи вставки, пошуку й обміну.
b) Методи обміну, вибору й вставки.
c) Методи пошуку, двійкового пошуку, вставки.
d) Методи пошуку, вставки й вибору.

31. ***Розташуйте по порядку*** етапи підготовки і розв'язування задачі на ЕОМ.

a) Вибір методу розв'язування задачі.
b) Математична формалізація задачі.
c) Перевірка алгоритму.
d) Постановка задачі.
e) Налагодження й тестування програми.
f) Експлуатація програми.

g) Розробка блок-схеми алгоритму.

h) Написання тексту програми.

i) Розв’язування задачі, аналіз результатів.

24. Що ви розумієте під ***постановкою задачі?***

a) Це перевірка правильності роботи програми і виправлення знайдених помилок.

b) Це опис задачі у вигляді формул, рівнянь, співвідношень, обмежень.

c) Це запис алгоритму на алгоритмічній мові програмування. ф Це використання програми замовниками або користувачами.

d) Це чітке формулювання задачі, визначення вхідних даних для її розв’язування і точні вказівки відносно того, які результати і в якому вигляді повинні бути отримані.

25. В чому полягає ***математична формалізація*** задачі?

a) Це перевірка правильності роботи програми і виправлення знайдених помилок.

b) Це графічний запис алгоритму на основі вибраного методу.

c) Це опис задачі у вигляді формул, рівнянь, співвідношень, обмежень.

d) Це запис алгоритму на алгоритмічній мові програмування.

e) Це чітке формулювання задачі, визначення вхідних даних для її розв’язування і точні вказівки відносно того, які результати і в якому вигляді повинні бути отримані.

26. В чому полягає ***вибір методу*** розв’язування задачі?

a) Це графічний запис алгоритму на основі вибраного методу.

b) У виборі того чи іншого способу розв’язування задачі.

c) Це опис задачі у вигляді формул, рівнянь, співвідношень, обмежень.

d) Це запис алгоритму на алгоритмічній мові програмування.

e) Це ручна перевірка («прокрутка») окремих розв’язків задачі.

27. Що ви розумієте під ***побудовою блок-схеми*** задачі?

a) Це перевірка правильності роботи програми і виправлення знайдених помилок.

b) Це графічний запис алгоритму на основі вибраного методу.

c) Це запис алгоритму на алгоритмічній мові програмування.

d) Це чітке формулювання задачі, визначення вхідних даних для її розв’язування і точні вказівки відносно того, які результати і в якому вигляді повинні бути отримані.

e) Це остаточна перевірка правильності реалізації всіх попередніх етапів.

28. Що ви розумієте під ***перевіркою*** алгоритму?

a) Це перевірка правильності роботи програми і виправлення знайдених помилок.
b) Це опис задачі у вигляді формул, рівнянь, співвідношень, обмежень.
c) Це використання програми замовниками або користувачами.
d) Це ручна перевірка («прокрутка») окремих розв’язків задачі.
e) Це остаточна перевірка правильності реалізації всіх попередніх етапів.

29. В чому полягає написання ***тексту програми?***

a) Це опис задачі у вигляді формул, рівнянь, співвідношень, обмежень.
b) Це запис алгоритму на алгоритмічній мові програмування.
c) Це використання програми замовниками або користувачами.
d) Це ручна перевірка («прокрутка») окремих розв’язків задачі.
e) Це остаточна перевірка правильності реалізації всіх попередніх етапів.

30. Що ви розумієте під ***налагодженням і тестуванням*** програми?

a) Це перевірка правильності роботи програми і виправлення знайдених помилок.
b) Це опис задачі у вигляді формул, рівнянь, співвідношень, обмежень.
c) Це запис алгоритму на алгоритмічній мові програмування.
d) Це використання програми замовниками або користувачами.

e) Це ручна перевірка («прокрутка») окремих розв’язків задачі.

31. ***Що ви розумієте під*** розв’язуванням задачі і аналізом результатів?
    a) Це перевірка правильності роботи програми і виправлення знайдених помилок.
    b) Це запис алгоритму на алгоритмічній мові програмування.
    c) Це ручна перевірка («прокрутка») окремих розв’язків задачі.
    d) Це чітке формулювання задачі, визначення вхідних даних для її розв’язування і точні вказівки відносно того, які результати і в якому вигляді повинні бути отримані.
    e) Це остаточна перевірка правильності реалізації всіх попередніх етапів.
32. Що таке ***експлуатація*** програми?
    a) Це опис задачі у вигляді формул, рівнянь, співвідношень, обмежень.
    b) Це запис алгоритму на алгоритмічній мові програмування.
    c) Це використання програми замовниками або користувачами.
    d) Це ручна перевірка («прокрутка») окремих розв’язків задачі.
    e) Це остаточна перевірка правильності реалізації всіх попередніх етапів.
33. Як називається процес побудови алгоритмів та програм, що виконується в такій послідовності:
    a) Попередній аналіз задачі з метою розбиття її на окремі прості частини (модулі).
    b) Послідовна (зверху донизу) деталізація частин та складання програм для кожного з модулів.
    c) Професійне програмування.
    d) Структурне програмування.
    e) Початкове програмування.