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

На правах рукопису

МІНТІЙ Ірина Сергіївна

УДК 378.016:004.432.42(043.3)

ФОРМУВАННЯ У СТУДЕНТІВ ПЕДАГОГІЧНИХ УНІВЕРСИТЕТІВ КОМПЕТЕНТНОСТЕЙ З ПРОГРАМУВАННЯ НА ОСНОВІ ФУНКЦІОНАЛЬНОГО ПІДХОДУ

13.00.02 – теорія та методика навчання (інформатика)

Дисертація на здобуття наукового ступеня
кандидата педагогічних наук

Науковий керівник:
СЕМЕРІКОВ Сергій Олексійович
доктор педагогічних наук, професор

Київ – 2013

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ВНЗ	вищий навчальний заклад
ГСВО	галузеві стандарти вищої освіти
ЕОМ	електронна обчислювальна машина
ЕС	експертна система
ЗІЕІТ	Запорізький інститут економіки та інформаційних технологій
ІКТ	інформаційно-комунікаційні технології
КДПУ	Криворізький державний педагогічний університет
КНУ	Криворізький національний університет
КОЗН	комп’ютерно-орієнтовані засоби навчання
НМетАУ	Національна металургійна академія України
ООП	об’єктно-орієнтоване програмування
СНД	Співдружність незалежних держав
ЕСПН	електронна система підтримки навчання
НМК	навчально-методичний комплекс
США	Сполучені Штати Америки
ФМП	функціональна мова програмування
ФП	функціональний підхід

ВСТУП

Актуальність дослідження. Кінець XX – початок XXI століть відзначався інтенсифікацією змін в усіх сферах суспільного життя, що неминуче викликають зміни в системі освіти, адже «замовником» її кінцевого результату є саме суспільство. Враховуючи швидкоплинність сучасних суспільних процесів, сьогодні неможливо підготувати фахівця, знання та вміння якого будуть актуальними протягом всього життя. Саме тому змінилась саме парадигма підготовки фахівців: необхідна підготовка не просто обізнаних, поінформованих, навчених певним діям спеціалістів, а фахівців, здатних до самоосвіти, ефективної взаємодії у розв'язанні соціальних, економічних та виробничих проблем, мобільних, відповідальних, самостійних, ініціативних, тобто про необхідність підготовки *компетентних фахівців*. Для фахівців, які працюють в сфері ІКТ, така підготовка є особливо актуальною ще й тому, що саме ця сфера є однією з найдинамічніших. Провідним напрямом перебудови підготовки є фундаменталізація навчання інформатичних дисциплін у вищій школі.

Одним зі шляхів оновлення змісту освіти, узгодження її з сучасними потребами є орієнтація освіти на компетентнісний підхід та створення ефективних механізмів його запровадження [129]. Над розробкою теоретичних основ компетентнісного підходу в освіті працюють вітчизняні і зарубіжні дослідники: І. І. Бабин [43], В. І. Байденко [45], Н. М. Бібік [50], М. С. Головань [74], О. М. Гончарова [79], О. М. Дахін [87], С. О. Дружилов [92], О. А. Дубасенюк [234], І. О. Зімняя [115], П. Згага [28], Е. Ф. Зеєр [114], В. В. Краєвський [133], Н. В. Кузьміна [137], А. К. Маркова [159], О. В. Овчарук [129], Л. А. Петровська [220], Т. П. Пєтухова [221; 222], О. І. Пометун [228], Дж. Равен [238], Н. Ф. Радіонова [239], С. А. Раков [240; 241], Е. Е. Симанюк [114], О. М. Спірін [273], В. А. Сухомлин [278], А. В. Хуторський [133; 296] та ін.

Говорячи про необхідність підготовки компетентного випускника школи, не можна забувати, що підготувати його може лише компетентний учитель, тому першочерговою задачею є реалізація компетентнісного підходу в процесі підготовки майбутніх учителів. Питання професійної компетентності вчителя та визна-

чення необхідних для його професії якостей розглядаються у працях І. А. Акуленко [138], В. П. Андрущенка [38], О. Г. Вернер [57], Г. Д. Воронцова [65], В. М. Гриньової [83], М. І. Жалдака [100], Л. Г. Карпової [121], А. Г. Кіріллова [124], Т. П. Кобильника [125], Н. В. Кузьміної [137], А. І. Кузьмінського [139; 138], О. Г. Ларіонової [145], О. В. Лебедєвої [147], І. В. Левченко [150], А. К. Маркової [159], Г. О. Михаліна [166; 167], Л. М. Мітіної [165], О. І. Нікіфорової [205], Н. Ф. Радіонової [239], С. А. Ракова [240; 241], Ю. С. Рамського [100; 244], Є. М. Смирнової-Трибульской [262; 264], О. М. Соловової [267], О. М. Спіріна [273], Н. А. Тарасенкової [138], А. П. Тряпіциної [239] та ін.

На особливу увагу заслуговує підготовка вчителя інформатики. Головною причиною цього є постійний розвиток науки інформатики та засобів ІКТ. Професійній підготовці вчителя інформатики присвячено праці Н. В. Апатової [39], С. О. Бешенкова [213], В. Ю. Бикова [49], Л. І. Білоусової [52], А. І. Бочкіна [11], А. Ф. Верланя [59], А. М. Гуржія [86], А. П. Єршова [94], М. І. Жалдака [95; 96; 102], В. Є. Жужжалова [107], А. Г. Кіріллова [46], О. М. Костікова [132], О. А. Кузнєцова [135], Е. І. Кузнєцова [136], В. В. Лаптєва [142], М. П. Лапчика [143; 144], І. В. Левченко [150], Ю. І. Машбиця [160], В. М. Монахова [213], Н. В. Морзе [202], С. Пейперта [218], С. А. Ракова [240; 241], Ю. С. Рамського [242; 245], М. І. Рагуліної [239], Н. І. Рижової [142], Є. М. Смирнової-Трибульской [263], О. В. Співаковського [269; 270], О. М. Спіріна [273], І. О. Теплицького [283], А. А. Харківської [293], М. В. Швецького [142] та ін.

Плідно працюють в напрямі теоретичної розробки професійних компетентностей вчителя інформатики молоді дослідники В. В. Ачкан [42], В. М. Жукова [108], Т. П. Кобильник [125], К. Р. Ковальська [126], О. В. Кучай [141], Ю. М. Лебеденко [149], М. В. Рафальська [100], Я. Б. Сікора [255], Ж. А. Чорна [302], В. С. Шелудько [307] та ін.

Навчання програмування надає можливість природним чином відобразити зв’язок двох головних складових інформатики: математичної інформатики та інформаційних технологій, тому й компетентності з програмування посідають важливе місце в системі інформатичних компетентностей майбутнього вчителя інфо-

рматики. Методиці навчання програмування та формуванню компетентностей з програмування присвячені роботи Х. Абельсона [29], А. Ахо [40], Н. Вірта [60], Л. В. Гришко [84], Е. Дейкстри [88], А. П. Єршова [94], М. І. Жалдака [100], В. Є. Жужжалова [39], Б. Кернігана [122], І. Г. Косової [131], С. Макконнелла [155], Н. В. Морзе [200], І. О. Одінцова [208], Р. Пайка [122], О. П. Поліщука [226], С. Прати [229], Ю. С. Рамського [244; 245], Дж. Сассман [29], Д. Д. Сассмана [29], Р. У. Себести [249], Д. А. Слінкіна [259], О. В. Співаковського [269; 270], І. О. Теплицького [281], Ю. В. Триуса [284; 285], Д. Ульмана [40], Дж. Хопкрофта [40], Г. Шилдта [308] та ін.

У процесі формування компетентностей з програмування можуть бути застосовані різні підходи, найбільш поширені з яких – імперативний та об'єктний – опановуються як послідовно, так і незалежно один від одного [6]. Проте фундаменталізація навчання програмування вимагає посилення ролі математичної інформатики, тому важливим є встановлення зв'язків математичної інформатики з програмуванням, математики з інформатикою з самого початку навчання програмування, що, в свою чергу, вимагає нового підходу до формування компетентностей з програмування. У розв'язанні проблеми формування компетентностей з програмування студентів педагогічних університетів в умовах фундаменталізації навчання інформатичних дисциплін може суттєво допомогти функціональний підхід до програмування.

Актуальність дослідження підсилюється ще й тим, що за кордоном для навчання основ програмування в шкільному курсі інформатики активно використовуються мови та середовища функціонального підходу: Logo, Squeak, Scratch, Alice. Вивченню функціональних мов програмування присвячено праці Х. Абельсона [29], С. В. Головльової [77], І. Г. Косової [131], С. М. Малярчука [158], Ю. І. Машбиця [160], С. О. Нігіяна [204], Д. Д. Сассмана [29], Дж. Сассман [22], Р. У. Себести [249], І. О. Теплицького [281; 282], А. Філда [291], П. Хендерсона [294], Е. Хювьонена [298; 299] та ін. У той же час у процесі навчання основ програмування майбутніх учителів не використовуються навіть елементи функціонального підходу. Як результат – неготовність випускників вищих навчальних за-

кладів – майбутніх учителів інформатики – до використання засобів функціонального підходу у шкільному курсі інформатики, особливо – у профільній школі.

Ураховуючи, що функціональний підхід є основою фундаменталізації навчання програмування майбутніх учителів інформатики, а компетентнісно-орієнтоване навчання програмування на основі функціонального підходу є теоретично та методично не розробленим напрямом як у вітчизняній, так і в зарубіжній теорії та методиці навчання інформатики, було обрано тему дослідження **«Формування у студентів педагогічних університетів компетентностей з програмування на основі функціонального підходу».**

Зв'язок роботи з науковими програмами, темами. Дисертаційне дослідження виконано відповідно до тематичного плану науково-дослідної роботи Інституту інформатики Національного педагогічного університету імені М. П. Драгоманова (номер державної реєстрації 0111U000526).

Тема затверджена на засіданні Вченої ради Національного педагогічного університету імені М. П. Драгоманова 26 червня 2009 року (протокол №12) та узгоджена в Міжвідомчій раді з координації наукових досліджень з педагогічних і психологічних наук в Україні 27 жовтня 2009 року (протокол № 7).

Мета дослідження полягає в розробці та науковому обґрунтуванні методичної системи формування у студентів напряму підготовки «Інформатика*» вищих педагогічних навчальних закладів компетентностей з програмування на основі функціонального підходу.

У процесі дослідження висувалась *гіпотеза*, що розробка і використання методичної системи навчання програмування на основі функціонального підходу сприятиме: а) формуванню у студентів компетентностей з програмування на високому рівні; б) поглибленню міжпредметних зв'язків математики та інформатики; в) фундаменталізації навчання програмування.

У відповідності до мети було поставлено наступні ***задачі***:

1. Узагальнити вітчизняний та зарубіжний досвід застосування компетентнісного підходу до підготовки майбутніх учителів;

2. Розробити структуру компетентностей з програмування та показники

сформованості їх складових;

3. Дослідити можливості формування компетентностей з програмування на основі функціонального підходу;

4. Розробити програмно-методичне забезпечення курсу «Вступ до програмування» для студентів напряму підготовки «Інформатика*» педагогічних університетів з використанням мов функціонального програмування;

5. Розробити окремі компоненти методичної системи навчання об'єктно-орієнтованого програмування та систем штучного інтелекту на основі функціонального підходу;

6. Експериментально дослідити ефективність методичної системи формування у студентів напряму підготовки «Інформатика*» педагогічних університетів компетентностей з програмування на основі функціонального підходу.

Об'єкт дослідження – процес формування компетентностей з програмування у майбутніх учителів інформатики.

Предмет дослідження – методична система навчання програмування майбутніх учителів інформатики на основі функціонального підходу.

Для розв'язування поставлених завдань використовувалися такі **методи дослідження**:

а) *теоретичні* – аналіз чинних стандартів освіти, навчальних програм, підручників і навчальних посібників, монографій, дисертаційних досліджень, статей і матеріалів науково-методичних конференцій з проблеми дослідження (розділ 1 (тут і надалі підрозділи дисертації)) – з метою визначення теоретичних засад дослідження; з питань програмування та методики його навчання (розділ 1, 1.2-1.5, розділ 2) – з метою визначення напрямів розвитку методики навчання програмування;

б) *емпіричні* – аналіз результатів навчання студентів у відповідності до проблеми дослідження, цілеспрямовані педагогічні спостереження, бесіди з викладачами та студентами, анкетування, тестування, аналіз досвіду роботи викладачів за основними положеннями дослідження (розділ 2, 2.2) – для констатування стану розв'язання проблеми; педагогічний експеримент (розділ 2, 2.3) – з метою переві-

рки ефективності розробленої методичної системи.

Наукова новизна одержаних результатів дисертаційного дослідження полягає полягає в тому, що автором було:

– *теоретично обґрунтовано та розроблено:*

1) структуру компетентностей з програмування майбутнього вчителя інформатики, виділено рівні сформованості компетентностей з програмування та визначено показники їх сформованості;

2) методичну систему формування у студентів педагогічних університетів компетентностей з програмування на основі функціонального підходу;

– *удосконалено* систему інформатичних компетентностей майбутнього вчителя інформатики;

– *дістали подальшого розвитку:*

1) методика навчання інформатики у вищій педагогічній школі;

2) система загальнопрофесійних компетентностей вчителя інформатики.

Практичне значення одержаних результатів дослідження полягає у створенні й впровадженні в практику підготовки майбутніх учителів інформатики методичної системи формування компетентностей з програмування на основі функціонального підходу, а саме:

1) *обґрунтовано* цілі навчання і зміст курсу «Вступ до програмування» на основі функціонального підходу;

2) *розроблено* посібник «Схематичне програмування (початки програмування: функціональний підхід)»;

3) *визначено* методи та форми організації навчання у процесі формування компетентностей з програмування на основі функціонального підходу;

4) *досліджено* можливості застосування програмних засобів мобільного навчання в процесі формування компетентностей з програмування на основі функціонального підходу:

– *локалізовано* середовище програмування DrRacket та досліджено дидактичні можливості його використання в процесі навчання програмування;

– *створено* нові інтерфейси мобільного інтерпретатора мови програмування

Scheme;

5) *розроблено* навчально-методичний комплекс «Вступ до програмування».

Особистий внесок здобувача. У працях, опублікованих у співавторстві, автору належать такі результати:

– розроблено методику навчання окремих розділів фундаментальної інформатики [169; 190];

– досліджено методичні основи формування інформатичних компетентностей у процесі навчання математики в середніх загальноосвітніх навчальних закладах [175];

– проаналізовано розвиток поняття «компетентність» та основні підходи стосовно відбору і визначення ключових компетентностей [174];

– розглянуто дидактичні можливості функціонального підходу у формуванні компетентностей з програмування [193];

– досліджено методичні основи формування компетентностей з програмування засобами мобільних інтерпретованих мов програмування (на прикладі мови Scheme), локалізовано середовище програмування DrRacket, створено нові інтерфейси користувача та розглянуто дидактичні можливості мобільного інтерпретатора Scheme [179];

– визначено та проаналізовано компоненти професійних компетентностей фахівця у галузі інформаційних технологій [185; 186];

– проаналізовано підходи до визначення поняття «методична система», визначено її компоненти та принципи розвитку і вдосконалення [181];

– розроблено програму [188];

– розроблено концепцію, зібрано матеріал, виконано опрацювання результатів, підготовлено текст статті [168].

Апробація результатів дисертації. Основні положення і результати дослідження доповідались, обговорювались і знайшли схвалення на наукових конференціях різного рівня:

– VI, VIII, IX та X міжнародних науково-практичних конференціях «Теорія та методика навчання фундаментальних дисциплін у вищій школі» (Кривий Ріг,

2006, 2010, 2011, 2012);

– VI міжнародній науково-технічній конференції «Комп'ютерні технології в будівництві» (Київ–Севастополь, 2008);

– Міжнародній науково-практичній конференції студентів та молодих науковців «Молодий науковець XXI століття» (Кривий Ріг, 2008);

– Міжнародній науково-методичній конференції «Проблеми математичної освіти» (Черкаси, 2009);

– VII, VIII, IX та X міжнародних науково-технічних конференціях «Новітні комп'ютерні технології» (Київ–Севастополь, 2009, 2010, 2011, 2012);

– Всеукраїнській науково-методичній конференції «Розвиток інтелектуальних умінь і творчих здібностей учнів та студентів у процесі навчання математики» (Суми, 2009);

– VII Всеукраїнській науково-практичній конференції «Інформаційні технології в освіті, науці і техніці» (Черкаси, 2010);

– I Всеукраїнській науково-методичній конференції студентів, аспірантів, молодих науковців «Інноваційні інформаційно-комунікаційні технології навчання математики, фізики, інформатики у середніх та вищих навчальних закладах» (Кривий Ріг, 2011);

– VII (XVII) міжнародній науково-практичній конференції «Засоби і технології сучасного навчального середовища» (Кіровоград, 2011).

Результати дослідження обговорювалися на засіданнях Всеукраїнського науково-методичного семінару «Актуальні проблеми методики навчання інформатики» (Київ, НПУ ім. М. П. Драгоманова, 2010, 2012), на засіданнях науково-методичного семінару кафедри інформатики та прикладної математики ДВНЗ «Криворізький національний університет» (2006–2012), на засіданні науково-методичного семінару кафедри інформаційних технологій Черкаського державного технологічного університету (2009), на засіданнях науково-методичного семінару кафедри інформатики Харківського національного педагогічного університету ім. Г. С. Сковороди (2010, 2012), а також апробовані шляхом публікацій.

Упровадження результатів дослідження здійснювалось у процесі:

– експериментального навчання курсу «Вступ до програмування» студентів І курсу спеціальностей «Інформатика», «Математика та інформатика», «Фізика та інформатика» у КДПУ (КНУ);

– вивчення розділу «Алгоритмізація та програмування» шкільного курсу інформатики студентами спеціальностей «Інформатика», «Математика та інформатика», «Фізика та інформатика», «Хімія та інформатика», «Трудове навчання та основи інформатики» у КДПУ (КНУ);

– вивчення розділу «Інформаційно-комунікаційні технології навчання» курсу «Нові інформаційні технології» студентами спеціальностей «Українська мова та література», «Українська та англійська мова», «Українська мова та психологія», «Історія та географія» у КДПУ (КНУ);

– викладання курсів «Об'єктно-орієнтоване програмування», «Подієорієнтоване програмування», «Візуальне програмування», «Системи штучного інтелекту», «Комп'ютерне моделювання», «Інформатика», «Функціональне програмування», «Експертні системи», «Комп'ютерні технології», «Основи інформатики та обчислювальної техніки» для студентів спеціальностей «Інформатика», «Математика та інформатика», «Фізика та інформатика», «Хімія та інформатика» КДПУ (КНУ), студентів спеціальностей «Професійне навчання« та «Комп'ютерні системи та мережі» Криворізького технічного університету (КНУ), студентів спеціальностей «Комп'ютерні системи та мережі», «Прикладна математика» ЗІЕІТ та студентів спеціальностей «Електромеханічні системи автоматизації та електропривод», «Автоматизоване управління технологічними процесами і виробництвами» НМетАУ (КНУ);

– лабораторно-обчислювального практикуму на фізико-математичному факультеті КДПУ (КНУ);

– підготовки студентів фізико-математичного факультету КДПУ (КНУ) за програмою Intel «Навчання для майбутнього».

Впровадження результатів дисертаційного дослідження у педагогічну практику підтверджується довідками Криворізького державного педагогічного універ-

ситету (№26/2-533 від 22.12.2010 р.), Національної металургійної академії України (№633 від 29.04.2011 р.), Криворізького відокремленого підрозділу Запорізького інституту економіки та інформаційних технологій (№63 від 02.03.2011 р.), Криворізького технічного університету, Глухівського національного педагогічного університету імені Олександра Довженка (довідка №2210 від 27.09.2012 р.).

Публікації. За матеріалами дослідження опубліковано 28 робіт, з них 9 статей у наукових фахових виданнях [171; 172; 173; 175; 177; 179; 182; 184; 185], з яких 6 одноосібні [171; 172; 173; 177; 182; 184], один навчальний посібник [191], 7 статей у збірниках наукових праць [169; 170; 181; 187; 189; 190; 192] та 11 матеріалів і тез конференцій, семінарів [168; 174; 176; 178; 180; 183; 186; 188; 193; 194; 195].

Структура та обсяг дисертації. Робота складається з переліку умовних позначень, вступу, 2 розділів, висновків до розділів, висновків, списку використаних джерел (310 найменувань, з них 28 іноземними мовами) та 6 додатків. Загальний обсяг дисертації 254 сторінки, із них 174 сторінки основного тексту. Робота містить 34 рисунки і 19 таблиць, розміщених на 34 сторінках.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ ФОРМУВАННЯ КОМПЕТЕНТНОСТЕЙ З ПРОГРАМУВАННЯ У СТУДЕНТІВ НАПРЯМУ ПІДГОТОВКИ «ІНФОРМАТИКА*» ПЕДАГОГІЧНИХ УНІВЕРСИТЕТІВ

1.1 Компетентнісний підхід у підготовці майбутніх учителів

Протягом останніх років стало загальнозрозуміло, що здебільшого освіта є занадто академічною та не готує молодих людей до майбутнього життя, становлення їх активної життєвої позиції й успішного виконання професійних обов'язків. Безперечно, знання, вміння та навички, які здобуваються в навчальних закладах є корисними. Але вони, як результат освіти, вже не задовольняють потреби сьогодення. Тому, починаючи з 90-х років XX століття вітчизняні та зарубіжні педагоги активно висловлюються про необхідність використання компетентнісного підходу в освіті. Провідна ідея цього підходу в тому, що головним завданням освіти має стати не отримання розрізнених, фрагментованих знань, вмінь та навичок з різних предметів, а формування здатності та готовності людини до плідної, ефективної діяльності в різноманітних сферах життя: особистій, професійній, соціальній. Основне завдання компетентнісного підходу – створення умов для набуття учнями/студентами досвіду діяльності в різних соціально-значущих ситуаціях. Ще одна відмінність полягає в тому, що, на відміну від знаннєвого підходу, коли результат освіти задається на вході (визначений перелік знань, вмінь та навичок), результат освіти задається на виході. Тобто, освітній процес спрямований в майбутнє. Звісно, надзвичайно складно спрогнозувати, які знання, вміння та навички будуть потрібні майбутнім фахівцям через десяток років. Тому результатом освіти при компетентнісному підході має стати не просто освіченість людини, а її компетентність.

1.1.1 Основні підходи до визначення поняття компетентності. Необхідність запровадження компетентнісного підходу в освіті плідно обговорюється членами відомих міжнародних організацій, серед яких ЮНЕСКО, ЮНІСЕФ, ПРООН, Рада Європи, Організація економічного співробітництва та розвитку,

Міжнародний департамент стандартів тощо [3]. Не зважаючи на значну кількість робіт, присвячених даному питанню, науковці досі не дійшли одностайної думки щодо трактування основоположного поняття компетентнісного підходу – «компетентність», а, відповідно, неоднозначною є і відповідь на питання, яку ж людину можна вважати «компетентною».

Різноманітні підходи до визначення даного поняття як вітчизняними, так і зарубіжними вченими аналізуються у роботах М. В. Алексєєва [32], О. Є. Антонової [234, 89], В. В. Ачкан [42], С. П. Бондар [54], М. С. Голованя [71], В. М. Гриньової [83], О. М. Дахіна [87], Е. Ф. Зеєра [114], І. О. Зімньої [115], О. Г. Ларіонової [149], Л. П. Маслак [234, 89], О. В. Овчарук [129, 7], О. І. Пометун [129, 19], Т. П. Пєтухової [221], Е. Е. Симанюк [114], Я. Б. Сікори [254] та ін.

Спробуємо узагальнити отримані ними результати та виділити головні стійкі ознаки даного поняття, але спочатку розглянемо тлумачення компетентності, які даються в різних словниках і енциклопедіях (табл. 1.1).

Таблиця 1.1

Тлумачення понять «компетентність» та «компетентний»
(за словниками та енциклопедіями)

№	Назва словника	Тлумачення понять	
		«компетентний»	**«компетентність»**
1.	Великий тлумачний словник сучасної української мови [58, 445]	який має достатні знання в якій-небудь галузі; який з чим-небудь добре обізнаний; тямущий	властивість за значенням компетентний
2.	Словник української мови [51, 250]	1. який має достатні знання в якій-небудь галузі; який з чим-небудь добре обізнаний; тямущий; 2. який ґрунтується на знанні; кваліфікований	властивість за значенням компетентний
3.	Сучасний тлумачний		(лат. competens – відповід-

№	Назва словника	Тлумачення понять	
		«компетентний»	«компетентність»
	психологічний словник [304, 203]		ний, здатний) психосоціальна якість, яка означає силу і впевненість, що виходять із почуття власної успішності й корисності, які дають людині усвідомлення своєї спроможності ефективно взаємодіяти з оточенням
4.	Словник іншомовних слів [260, 431]	(лат. competens (competentis) – належний, відповідний) 1. досвідчений у певній галузі, в якомусь питанні; 2. повноважний, повноправний у розв'язанні якоїсь справи	поінформованість, обізнаність, авторитетність
5.	Соціологія: словник термінів і понять [268, 141]		(лат. competo – відповідність, здатність) психосоціальна якість, яка означає силу і впевненість, джерелом яких є відчуття власної успішності та корисності. Сприяє усвідомленню особистістю власної здатності ефективно взаємодіяти з оточенням
6.	Економічна енцикло-		(лат competens – відповід-

№	Назва словника	Тлумачення понять	
		«компетентний»	«компетентність»
	педія [93, 803]		ний) 1. сфера повноважень органу, який здійснює управління, а також посадової особи, коло питань, з яких вони мають право прийняття рішень; 2. знання і досвід у певній сфері

Як бачимо, трактування досить різні. Це і об'єктивні характеристики – знання, досвід, сфера повноважень; так і суб'єктивні – психосоціальна якість, обізнаність, кваліфікованість, повноправність, авторитетність, повноважність.

В табл. 1.2 показано, як до визначення поняття «компетентність» підходять різні автори.

Таблиця 1.2

Тлумачення поняття «компетентність» різними авторами

№	Автор	Тлумачення «компетентності»
1.	Міжнародна комісія ради Європи [7]	загальні, або ключові, вміння, базові вміння, фундаментальні шляхи навчання, ключові кваліфікації, кроснавчальні вміння або навички, ключові уявлення, опори, або опорні знання
2.	Експерти країн Європейського Союзу [17]	здатності (знання, навички, цінності, ставлення), що сприяють успішно вирішувати всі життєві питання: особисті, соціальні та професійні
3.	Дж. Равен [237, 6]	специфічна здатність, яка необхідна для ефективного використання певних дій у визначеній предметній області та містить вузькоспеціальні знання, певного роду предметні на-

№	Автор	Тлумачення «компетентності»
		вички, способи мислення та розуміння відповідальності за свої дії
4.	І. О. Зімняя [116, 16]	актуальна, сформована якість особистості, яка базується на знаннях; інтелектуально і особистісно обумовлена соціально-професійна характеристика людини, її особистісна якість
5.	О. М. Дахін [275, 3]	наявність у людини необхідних знань і здатностей, які дають змогу аналізувати, робити висновки й приймати ефективні рішення, а також раціонально діяти, реалізуючи вказані рішення
6.	І. Г. Агапов, М. Е. Шишов [309]	загальна здатність і готовність особистості до діяльності, що основана на знаннях і досвіді, які придбані завдяки навчанню, орієнтовані на самостійну участь особистості в навчально-пізнавальному процесі, а також спрямовані на її успішне включення в трудову діяльність
7.	О. Є. Антонова, Л. П. Маслак [234, 89]	гармонійне, інтегроване, системне поєднання знань, умінь і навичок, норм, емоційно-ціннісного ставлення та рефлексії, що складають мінімальну готовність особистості до вирішення практичних завдань
8.	Н. М. Бібік [129, 46]	в інтегрованому вигляді представляє освітні результати, які досягаються не лише засобами змісту освіти, але й соціальної взаємодії; як у міжособистісному, так і в інституційному культурному контексті. Таких результатів можна спрогнозувати безліч (у сфері когнітивній, діяльнісній, мотиваційній, соціальній)
9.	С. П. Бондар [54, 99]	загальна здатність і готовність до продуктивної діяльності, інтегрована характеристика якості особистості, результативний блок, сформований через досвід, знання, вміння, став-

№	Автор	Тлумачення «компетентності»
		лення, поведінкові реакції
10.	М. С. Головань [74, 30]	інтегративне утворення особистості, що інтегрує в собі знання, уміння, навички, досвід і особистісні властивості, які обумовлюють прагнення, здатність і готовність розв'язувати проблеми і завдання, що виникають в реальних життєвих ситуаціях, усвідомлюючи при цьому значущість предмету і результату діяльності
11.	С. А. Раков [241, 21]	рівень досягнення компетенцій. Компетенції – еталон досвіду дій, знань, умінь, навичок, творчості, емоційно-ціннісної діяльності, який встановлює суспільство
12.	Комісія з розробки ГСВО [68, 13]	інтегрована характеристика якостей особистості, результат підготовки випускника вузу для виконання діяльності в певних професійних та соціально-особистістних предметних областях, який визначається необхідним обсягом і рівнем знань та досвіду у певному виді діяльності.

Отже, на нашу думку, ***компетентність*** – це освітній результат (який формується не лише в процесі формальної освіти (навчання в школі), але й неформальної (родина, друзі, робота, політика, релігія, культура й ін.)); сформована особистісна якість, яка містить наступні ***складові***:

– когнітивно-змістову (гносеологічну): знання;

– операційно-технологічну (праксеологічну): навички, уміння, досвід діяльності;

– ціннісно-мотиваційну (аксіологічну): мотивація, ціннісне ставлення;

– соціально-поведінкову: комунікабельність, здатність до адаптації, здатність до інтеграції, вміння спілкуватися, розуміти, поважати та оцінювати різні підходи до розв'язання однієї задачі.

Виокремлені складові не є розрізненими, вони тісно взаємодіють між собою.

Система компетентностей в освіті має трирівневу ієрархічну структуру [129, 21]:

– *ключові* компетентності (міжпредметні або надпредметні компетентності);

– *загальнопредметні* (їх можна сформувати під час вивчення певного предмету або освітньої галузі на протязі всього терміну навчання);

– *предметні* (їх можна сформувати під час вивчення конкретної навчальної дисципліни протягом визначеного часу).

При цьому С. А. Раков зазначає, що «кожна ключова компетентність повинна «проектуватись» на загальногалузеві компетентності, які у свою чергу повинні «проектуватись» на предметні компетентності (під терміном «проектуватись» розуміється таке визначення компетентностей нижчого рівня, щоб вони у сукупності забезпечували компетентності вищих рівнів). Разом із тим, ключові компетентності не складаються просто з набору відповідних галузевих та предметних компетентностей – вони інтегрують галузеві компетентності у складну структурну компоненту, у якій елементи пов’язані між собою різноманітними зв’язками та відношеннями» [241, 26].

Надпредметні (ключові) компетентності мають наступні характеристики:

– є синтетичними, такими, що поєднують певний комплекс знань, умінь та ставлень, що набувається протягом засвоєння всього змісту освіти;

– не пов’язані з конкретним предметом, до них належать компетентності, що їх можна набути під час засвоєння не одного предмета, а тільки декількох або всіх одночасно (тобто використовуючи всі навчальні можливості, пропоновані формальною й неформальною освітою);

– можуть бути метафорично визначені як персональні засоби, «ноу-хау», «процедурні знання» учнів, які формуються в них після того, як вони «забувають» фактичні знання, здобуті в школі протягом шкільного життя [129, 21].

Країни-члени організації економічного співробітництва та розвитку (ОЕСР) визначили ***ключові компетентності в трьох широких категоріях*** [25]. По-перше, люди мають бути в змозі використовувати весь діапазон засобів для ефективної взаємодії з навколишнім середовищем: як фізичних (інформаційні техно-

логії), так і соціокультурних (мова). Вони мають знати та розуміти ці засоби досить добре, щоб активно використовувати їх у власних цілях. По-друге, у все більш і більш взаємозалежному світі, люди мають бути здатними взаємодіяти з іншими людьми. Оскільки вони будуть стикатися з різними групами людей, важливо, щоб вони були здатні взаємодіяти в соціально гетерогенних групах. По-третє, люди мають бути готові приймати відповідальність за керування власним життям.

Перша категорія компетентностей: *інтерактивне використання засобів*.

Інтерактивне використання засобів передбачає більше, ніж володіння просто технічними навичками (наприклад, читати текст, використовувати програмне забезпечення). Люди мають також створювати та застосовувати знання та навички. Це вимагає розуміння того, як засоби допомагають взаємодіяти зі світом та як їх використовувати для того, щоб досягти кращих результатів. У цьому розумінні засоби не просто пасивні посередники, але й активні учасники в діалозі між людиною та навколишнім середовищем. До цієї категорії належать такі здатності:

– інтерактивне використання мови, символіки та текстів (ефективне використання навичок усного та письмового мовлення, обчислення, математичних навичок в різноманітних ситуаціях);

– інтерактивне використання знань та інформації (визнання та визначення невідомого; визначення місцезнаходження та способів доступу до відповідних джерел інформації; оцінювання якості, доцільності та цінності як інформації, так і її джерел; упорядкування знань та інформації);

– інтерактивне використання технологій (технічні досягнення надають людям нові можливості задовольняти власні потреби більш ефективно та новими способами; ІКТ мають потенціал для перетворення способів співробітництва людей (зменшуючи важливість місцезнаходження), доступу до інформації (надаючи швидкий доступ до великих обсягів даних) та взаємодії з іншими; важливо, щоб люди могли поєднати можливості технологічних засобів з власними потребами).

Друга категорія компетентностей: *взаємодія в соціально гетерогенних групах*.

Протягом всього життя кожна людина залежна від взаємодії з іншими людьми. Оскільки суспільство стає все більш різношаровим та фрагментованим, стає все складніше керувати міжособистісними стосунками. Компетентності цієї категорії необхідні людині для навчання, життя та роботи у тісній взаємодії з іншими людьми. Це здатності до:

– гарних стосунків з іншими (співчуття, керування емоціями);

– співробітництва, роботи в команді (вміння подати власні ідеї та вислухати ідеї інших, проводити переговори, приймати рішення, враховуючи різні точки зору);

– розв'язування конфліктів (здатність аналізувати загрози для власних інтересів, походження конфлікту та розмірковувати з різних точок зору, визнаючи різні можливі рішення; складати ієрархії потреб та цілей, визначаючи найголовніше).

Третя категорія компетентностей: *автономна дія*.

Автономна дія не означає функціонування в соціальній ізоляції. Вона передбачає розуміння навколишнього середовища, соціальної динаміки та власних ролей. Люди мають автономно діяти для ефективної участі в розвитку суспільства та функціонування в різних сферах життя, включаючи роботу, сім'ю та суспільне життя. Вони повинні вміти робити власний вибір, а не діяти за вказівками. Ця категорія передбачає такі здатності:

– діяти в широкому контексті (прийняття до уваги норм моралі, соціально-економічних закладів; визначення наслідків власних дій);

– складати власні плани (здатність визначати проект та встановлювати мету, ідентифікувати та оцінювати наявні та необхідні ресурси (наприклад: час, гроші), розміщувати за пріоритетами та вдосконалювати власні цілі, робити висновки з минулих проектів при побудові нових);

– відстоювати власні права, інтереси та потреби (здатність розуміти власні інтереси, наводити аргументи для визнання своїх прав та потреб, пропонувати альтернативні рішення).

Єврокомісія виділяє ***8 ключових компетентностей***, якими повинен воло-

діти кожен європеєць [19]:

1) *компетентності в галузі рідної мови*: здатність виражати та розуміти думки, почуття, факти та точки зору в усній та письмові формі (слухаючи, говорячи, читаючи та пишучи), та мовно взаємодіяти в усіх сферах суспільного життя;

2) *компетентності в сфері іноземних мов* (здатність, що доповнює компетентності в галузі рідної мови, сприяючи міжкультурному спілкуванню та розумінню);

3) *математичні, фундаментальні природничонаукові та технічні компетентності* (математичні компетентності – це здатність розвивати та використовувати математичне мислення для розв'язання щоденних задач; фундаментальні природничонаукові та технічні компетентності – це здатність використовувати знання та методи для розуміння явищ навколишнього світу, розуміння змін, спричинених людською діяльністю та відповідальністю кожної людини за них);

4) *комп'ютерні компетентності* (впевнене та творче використання ІКТ);

5) *навчальні компетентності* (вміння організовувати процес власного або ж колективного навчання, у відповідності з власними потребами та можливостями);

6) *міжособистісні, міжкультурні, соціальні та громадянські компетентності* (міжособистісні, міжкультурні та соціальні компетентності – дотримання етичних норм, плідна участь в соціальному житті суспільства; громадянські компетентності – знання соціальних, політичних принципів та структур (демократія, правосуддя, рівність, громадянство, громадянські права), активна громадська позиція);

7) *компетентності підприємництва* (здатність втілювати ідеї в життя, здатність діяти, ризикувати, планувати та керувати проектами, ставити та досягати власних цілей);

8) *культурні компетентності* (здатність творчо виражати ідеї, емоції, факти за допомогою мистецтва: в музиці, літературі, живописі чи ін).

Вітчизняні педагоги виокремлюють такі ключові компетентності [129, 85]:

1) *навчальні компетентності* (інтелектуальний розвиток особистості та здатність вчитися протягом всього життя);

2) *соціальні компетентності* (володіння сукупністю засобів, що надають можливість особистості взаємодіяти з різними соціальними групами та соціальними інститутами суспільства);

3) *загальнокультурні компетентності* (здатність жити та взаємодіяти з іншими в умовах полікультурного суспільства, керуючись національними та загальнолюдськими духовними цінностями);

4) *здоров'язберігаючі компетентності* (спрямованість на збереження фізичного, соціального, психічного та духовного здоров'я – свого та оточення);

5) *ІКТ-компетентності* (здатність орієнтуватись в інформаційному просторі, володіти й оперувати даними відповідно до потреб ринку праці);

6) *громадянські компетентності* (здатність захищати та піклуватися про відповідальність, права, інтереси та потреби людини і громадянина української держави і суспільства);

7) *підприємницькі компетентності* (володіння засобами, що дають особі можливість ефективно організувати особисту та колективну трудову й підприємницьку діяльність).

1.1.2 Загальнопрофесійні компетентності вчителя. *Компетентності вчителя* – комплекс педагогічних здібностей і можливостей, вмотивована спрямованість на навчально-виховний процес, система необхідних знань, умінь, навичок і досвіду, які постійно вдосконалюються і реалізуються на практиці [138, 146].

Н. Ф. Радіонова та А. П. Тряпіцина [239] *професійні компетентності вчителя* визначають як інтегральну характеристику, яка визначає здатність вирішувати професійні проблеми та типові професійні задачі, які виникають в реальних ситуаціях професійної педагогічної діяльності, з використанням знань, професійного та життєвого досвіду, цінностей та схильностей. Компетентність завжди «виявляється» в дії, діяльності, поведінці та вчинках, не можна побачити «невиявлену» компетентність. Професійні компетентності вчителя виявляються при вирішенні професійних задач.

Розуміючи професійну підготовку як процес професійного розвитку, оволодіння досвідом майбутньої професійної діяльності, можна визначити, що компе-

тентний спеціаліст завжди орієнтується на майбутнє, передбачає зміни, орієнтований на самоосвіту. Особливістю професійних компетентностей є те, що вони реалізуються в сьогоденні, а орієнтовані на майбутнє.

Професійні компетентності – це сукупність ключових, базових та спеціальних компетентностей [239].

Ключові компетентності – це компетентності, необхідні для будь-якої професійної діяльності, сприяють успіху людини у сучасному мінливому світі.

Базові компетентності відображають специфіку визначеної професійної діяльності (педагогічної, медичної, технічної та ін.).

Для професійної педагогічної діяльності базовими є компетентності, необхідні для організації професійної діяльності в контексті вимог до системи освіти на визначеному етапі розвитку суспільства.

Спеціальні компетентності відображають специфіку конкретної предметної чи надпредметної сфери професійної діяльності. Спеціальні компетентності можна розглядати як реалізацію ключових та базових компетентностей в галузі навчального предмету, конкретної галузі професійної діяльності.

Всі три види компетентностей взаємопов’язані та розвиваються одночасно, що забезпечує становлення професійних компетентностей як визначеної цілісної, інтегративної особистісної характеристики фахівця.

На нашу думку, для означення компетентностей, які відображають загальну специфіку професійної діяльності, краще підходить характеристика «загальнопрофесійні». Адже, зазвичай, характеризуючи певні властивості, як базові, ми маємо на увазі не лише найзагальніші, але й найпростіші.

На думку В. О. Адольфа «професійні компетентності (педагога)– складне утворення, що містить комплекс знань, вмінь, властивостей і якостей особистості, які забезпечують варіативність, оптимальність та ефективність побудови навчально-виховного процесу» [30, 118].

З досліджень В. Ю Стрельнікова: «професійні компетентності вчителя – це знання навчально-виховного процесу, сучасних проблем педагогіки та психології, а також уміння застосувати ці знання у повсякденній практичній роботі» [276, 44].

Як зазначає Н. Г. Ничкало, професійні компетентності вчителя – це «гармонійне поєднання знань навчальної дисципліни, методики і дидактики викладання, а також умінь і навичок культури педагогічного спілкування» [206, 8]. У структурі професійних компетентностей вчителя вона вирізняє дві підструктури: діяльнісну (знання, уміння, навички і здібності особистості, необхідні для здійснення педагогічної діяльності) і комунікативну (знання, уміння, навички, необхідні особистості для здійснення педагогічного спілкування). Таким чином, у структурі загальнопрофесійної компетентності вчителя Н. Г. Ничкало виокремлює такі компетентності: предметні, методичні, дидактичні, комунікативні.

В. М. Гриньова вважає, що професійні компетентності педагога є чинником підвищення якості освіти і включають професійно-змістовий, технологічний і професійно-особистісний компоненти [83, 25]. Професійно-змістовий компонент передбачає наявність у викладача цінностей – знань з предмету, який він викладає, суміжних дисциплін, з дисциплін, що виражають квінтесенцію спеціальності, якими має оволодіти студент, теоретичних знань з основ наук, які вивчають особистість людини, що забезпечує усвідомленість при визначенні педагогом змісту його професійної діяльності з виховання, навчання та освіти студентів. Технологічний компонент включає професійні цінності – знання, апробовані в дії, тобто цінності-вміння. Забезпечують цей компонент інформаційно-інноваційні технології, які ґрунтуються на комплексному діагностико-дослідному осмисленні педагогічної ситуації і перспективному її прогнозуванні. Професійно-особистісний компонент включає особистісні здібності-цінності [83, 25]. Отже, структура загальнопрофесійних компетентностей вчителя за В. М. Гриньовою наступна: предметні; психологічні; методичні; ІКТ-компетентності.

Ґрунтовно розглядаються професійні компетентності педагога і в роботах С. О. Дружилова: «професійні компетентності педагога – це багатофакторне явище, що містить в собі систему теоретичних знань учителя та способів їх застосування в конкретних педагогічних ситуаціях, ціннісні орієнтації педагога, а також інтегративні показники його культури (мова, стиль спілкування, відношення до себе та своєї діяльності, до суміжних галузей знань та ін.)» [92, 28]. В професій-

них компетентностях педагога науковець виокремлює такі компоненти: мотиваційно-вольовий, функціональний, комунікативний та рефлексивний.

Мотиваційно-вольовий компонент включає в себе: мотиви, цілі, потреби, ціннісні установки, стимулює творчий прояв особистості в професії; передбачає наявність інтересу до професійної діяльності.

Функціональний компонент в загальному випадку проявляється у вигляді знань про способи педагогічної діяльності, необхідних вчителю для проектування та реалізації тієї чи іншої педагогічної технології.

Комунікативний компонент компетентностей включає вміння ясно і чітко викладати думки, переконувати, аргументувати, будувати докази, аналізувати, висловлювати судження, передавати раціональну і емоційну інформацію, встановлювати міжособистісні зв'язки, погоджувати свої дії з діями колег, вибирати оптимальний стиль спілкування в різних ділових ситуаціях, організовувати і підтримувати діалог.

Рефлексивний компонент виявляється в умінні свідомо контролювати результати своєї діяльності і рівень власного розвитку, особистісних досягнень; сформованість таких якостей і властивостей, як креативність, ініціативність, націленість на співробітництво, співтворчість, схильність до самоаналізу. Рефлексивний компонент є регулятором особистісних досягнень, пошуку особистого сенсу у спілкуванні з людьми, самоврядування, а також збудником самопізнання, професійного зростання, вдосконалення майстерності та формування індивідуального стилю роботи [92, 28].

Карпова Л. Г. професійні компетентності учителя визначає як «інтегративне особистісне утворення на засадах теоретичних знань, практичних умінь, значущих особистісних якостей та досвіду, що зумовлює готовність учителя до виконання педагогічної діяльності та забезпечує високий рівень її самоорганізації». Професійні компетентності вчителя не мають вузько професійних меж, оскільки від нього вимагається постійне осмислення розмаїття соціальних, психологічних, педагогічних та інших проблем, які пов'язані з освітою [121, 28].

В. О. Сластьонін стосовно визначення професійних компетентностей педа-

гога зауважує: «поняття професійних компетентностей педагога виражає єдність його теоретичної і практичної готовності до здійснення педагогічної діяльності і характеризує його професіоналізм» [258, 30].

Структура професійних компетентностей учителя, згідно з В. О. Сластьоніним, може бути розкрита через педагогічні вміння. Педагогічні вміння він об'єднує в чотири групи.

1. Уміння «переводити» зміст об'єктивного процесу виховання в конкретні педагогічні завдання: вивчення особистості і колективу для визначення рівня їх підготовленості до активного оволодіння новими знаннями і проектування на цій основі розвитку колективу й окремих учнів; виділення комплексу освітніх, виховних і розвиваючих завдань, їх конкретизація і визначення домінуючого завдання (психолого-педагогічні компетентності).

2. Уміння побудувати і привести в рух логічно завершену педагогічну систему: комплексне планування освітньо-виховних завдань; обґрунтований відбір змісту освітнього процесу; оптимальний вибір форм, методів і засобів його організації (дидактичні компетентності).

3. Уміння виокремлювати і встановлювати взаємозв'язки між компонентами і факторами виховання, приводити їх у дію: створення необхідних умов (матеріальних, морально-психологічних, організаційних, гігієнічних та ін); активізація особистості школяра, розвиток його діяльності, що перетворює його із об'єкта в суб'єкт виховання, організація і розвиток спільної діяльності, забезпечення зв'язку школи із середовищем, регулювання зовнішніх, не програмованих впливів (психолого-педагогічні, організаційні компетентності).

4. Уміння обліку та оцінки результатів педагогічної діяльності: самоаналіз і аналіз освітнього процесу і результатів діяльності вчителя; визначення нового комплексу домінуючих і підпорядкованих педагогічних завдань (психолого-педагогічні компетентності).

Теоретична діяльність, у свою чергу, виявляється в узагальненому умінні педагогічно мислити, що передбачає наявність у вчителя аналітичних, прогностичних, проективних, а також рефлексивних умінь.

Зміст практичної готовності виражається у зовнішніх (предметних) уміннях, тобто в діях, які можна спостерігати. До них належать організаторські і комунікативні уміння [258, 30].

Таким чином, В. О. Сластьонін виокремлює у загальнопрофесійних компетентностях вчителя психолого-педагогічні, дидактичні, організаційні та комунікативні компетентності.

Розглядаючи питання професійних компетентностей І. О. Зімняя [116, 16] зазначає, що в результаті навчання в людини мають бути сформовані деякі соціально-професійні якості, які надають можливість їй успішно виконувати виробничі задачі та взаємодіяти з іншими людьми. Ці якості можуть бути визначені як соціально-професійні компетентності людини.

В такому розумінні соціально-професійні компетентності людини є його особистісні, інтегративні, сформовані якості, які проявляються в адекватності розв’язання (стандартних та особливо нестандартних, творчих) задач у всьому розмаїтті соціальних та професійних ситуацій.

Колективом авторів [100, 8] запропоновано таку структуру соціально-професійних компетентностей вчителя інформатики:

1) соціально-значущі компетентності;

2) професійні компетентності:

– загальнопрофесійні:

• дидактико-методичні;

• організаційно-управлінські;

• психолого-педагогічні;

• комунікативні;

• дослідницькі;

• природничо-математичні;

– предметні:

• інформологічно-методологічні;

• інформаційно-технологічні;

• у галузі комп’ютерної інженерії;

• у галузі моделювання;

• у галузі алгоритмізації і програмування.

О. Г. Ларіонова [145, 32] в структурі професійних компетентностей вчителя виокремлює інформаційно-методологічні, теоретичні, методичні, соціально-комунікативні та особистісно-валеологічні.

Приводом для такої класифікації стали провідні сфери життя людини в сучасному суспільстві та сфери діяльності вчителя. Кожна з груп компетентностей передбачає сформованість базових компетентностей, які складаються з менш узагальнених компонентів, які розкривають специфіку групи в цілому.

О. В. Лебедєва визначає таку структуру професійних компетентностей вчителя [147]:

1) науково-теоретичні компетентності:

– спеціальні компетентності (фундаментально-наукова підготовка) – знання й уміння у галузі фахового предмета, наукові основи шкільного курсу відповідної шкільної дисципліни;

– методологічні компетентності – знання філософії науки як методологічної основи пізнавальної діяльності (методи наукового пізнання у фаховій галузі);

– інформаційні компетентності – навички й уміння орієнтуватися в інформаційному просторі, використовувати комп’ютерні технології на різних етапах навчально-виховного процесу;

2) методичні компетентності:

– загальнопедагогічні;

– дидактичні;

– конкретно-методична підготовка, яка представлена через специфічні методи і прийоми навчання;

3) психолого-педагогічні компетентності:

– комунікативні компетентності – знання й уміння щодо побудови сприятливого психологічного клімату, відносин співробітництва в системах «учитель-учень», «учень-учень», продуктивних стосунків з колегами;

– диференціально-психологічні компетентності, які пов’язані з виявленням

особистісних якостей і спрямованості учнів, з формуванням і розвитком їхньої пізнавальної активності.

В даному випадку, вважаємо, що загальнопрофесійні компетентності вище описаної структури складаються з таких компонентів: інформаційні, загальнопедагогічні, дидактичні, методичні, комунікативні та психологічні компетентності.

Л. М. Мітіна [165] професійні компетентності вчителя розуміє як гармонійне поєднання знання предмету, методики та дидактики викладання, а також умінь та навичок (культури) педагогічного спілкування. Тобто, можна вважати, що в структурі загальнопрофесійних компетентностей вона виокремлює методичні, дидактичні та комунікативні компетентності.

О. М. Соловова [267] в структурі професійних компетентностей вчителя виокремлює такі компетентності: філологічні, методичні, управлінські, соціальні, психолого-педагогічні та в сфері ІКТ (саме ці компетентності, згідно нашого тлумачення, утворюють систему загальнопрофесійних компетентностей), наукові та предметні.

Згідно з О. М. Спіріним [273, 212], система компетентностей вчителя інформатики наступна:

1) загальні компетентності:

– компетентності щодо індивідуальної ідентифікації й саморозвитку;

– міжособистісні компетентності;

– суспільно-системні компетентності;

2) професійні компетентності:

– загальнопрофесійні компетентності;

– предметно-орієнтовані (профільно-орієнтовані) компетентності:

• науково-предметні компетентності;

• предметно-педагогічні компетентності;

– технологічні компетентності:

• компетентності в галузі педагогічних технологій;

• інформаційно-технологічні компетентності;

– професійно-практичні компетентності.

Згідно визначеної нами структури професійних компетентностей вчителя, до загальнопрофесійних компетентностей за О. М. Спіріним віднесемо загальнопрофесійні, предметно-педагогічні та технологічні, а до спеціальних професійних – науково-предметні та професійно-практичні компетентності.

Н. Ф. Радіонова та А. П. Тряпіцина [239] виокремлюють 5 основних груп задач, досвід вирішення яких характеризує базові (в нашому трактуванні – загальнопрофесійні) компетентності вчителя:

– бачити учня в навчальному процесі (психолого-педагогічні компетентності);

– будувати навчальний процес, орієнтований на досягнення мети конкретного ступеню навчання (дидактичні компетентності);

– встановлювати взаємодію з іншими суб'єктами навчального процесу, партнерами школи (організаційно-управлінські компетентності);

– створювати та використовувати з педагогічною метою навчальне середовище (загальнопедагогічні компетентності);

– проектувати та здійснювати професійну самоосвіту (навчальні компетентності).

А. Г. Кіріллов [124, 47] у структурі загальнопрофесійних компетентностей вчителя виокремлює технологічні, когнітивні, психологічні, регулятивні, дослідницькі та методичні компетентності.

Дещо по-іншому підходить до структурування загальнопрофесійних компетентностей вчителя А. К. Маркова [159, 82]. Згідно з нею, до складу загальнопропрофесійних компетентностей входять процесуальні та результативні. У процесуальних вона виокремлює педагогічну діяльність, педагогічне спілкування та особистість вчителя; в результативних – навченість та вихованість учнів.

Для оцінки рівня сформованості компетентностей, А. К. Маркова для кожної з складових пропонує розглядати сукупність необхідних знань, вмінь та психологічні вимоги до їх виконання.

Н. В. Кузьміна [137] структуру професійно-педагогічних (загальнопрофе-

сійних – у нашому трактуванні) компетентностей вчителя визначає наступним чином:

– методичні компетентності щодо способів формування знань, вмінь і навичок учнів;

– соціально-педагогічні компетентності у сфері процесів спілкування;

– диференціально-психологічні компетентності у сфері мотивів, здатностей і спрямувань учнів;

– аутопсихологічні компетентності у галузі переваг і недоліків власної діяльності й особливостей власних особистісних якостей;

– спеціальні і професійні компетентності у сфері тієї дисципліни, що викладається.

Указом Президента України від 4 липня 2005 року № 1013/2005 (п. 7) «Про невідкладні заходи щодо забезпечення функціонування та розвитку освіти в Україні» визначені заходи, спрямовані на реалізацію в Україні положень Болонської декларації, зокрема, з розроблення та затвердження нових ГСВО [230].

В основі розробки нових галузевих стандартів вищої освіти України покладено компетентнісний підхід, у відповідності до якого одним із ключових моментів оцінки якості процесу навчання є результат формування системи компетентностей.

Згідно ГСВО систему компетентностей майбутнього фахівця з інформаційних технологій та викладача-стажиста складають соціально-особистісні, загальнонаукові, інструментальні, загально-професійні та спеціалізовано-професійні компетентності [68, 18].

Процес розробки ГСВО для напрямків інформатичного профілю – складний поетапний процес, в якому доцільно враховувати досвід побудови аналогічних систем як у вітчизняній, так й у зарубіжній вищій школі.

Експерти Міжнародної організації ЮНЕСКО також активно працюють над розробкою професійних компетентностей учителів. Особливу увагу вони приділяють ІКТ-компетентностям [3]. Вони зазначають, що для більш успішного життя, навчання та роботи в інформаційному суспільстві студенти та викладачі по-

винні використовувати існуючі технології.

В рамках освітнього процесу використання ІКТ надають студентам можливість:

– здійснювати пошук даних, їх аналіз; виконувати обчислення;

– вирішувати проблеми та приймати рішення;

– творчо та ефективно використовувати всі можливі інструменти для підвищення власної продуктивності;

– стати інформованими, відповідальними, свідомими громадянами.

Завдяки ефективному використанню ІКТ в процесі навчання студенти можуть значно підвищити свої можливості. Ключовою фігурою в допомозі студентам з розвитку власних можливостей має стати викладач.

Тому сучасним викладачам необхідно бути готовими, щоб забезпечити підтримку студентам щодо вивчення ІКТ, ознайомити їх з перевагами, які вони можуть надати та бути готовими використовувати ІКТ. Традиційні освітні методи вже більше не забезпечують в повній мірі майбутніх викладачів всіма необхідними навичками.

Проект ЮНЕСКО «ICT Competency Standards for Teachers (ICT-CST)» («Стандартні ІКТ-компетентності для викладачів») [3] створений з метою:

а) розробки повної структури компетентностей з ІКТ;

б) проектування освітніх стандартів існуючого навчання і освітніх програм на ICT-CST;

в) здійснення спроби прискорення глобальних змін в цій області.

Цей проект створений за підтримки таких організацій: Microsoft (Майкрософт), Intel (Інтел), Cisco (Ціско), the International Society for Technology in Education (ISTE) (Міжнародна організація технологій в навчанні), Virginia Polytechnic Institute (Політехнічний інститут Вірджинії), State University (Virginia Tech) (Державний технічний університет Вірджинії).

Запропонований навчальний план з курсу, задачею якого є формування ІКТ-компетентностей у вчителів розвиває три підходи: освіченість, поглиблення і створення знань; з шістьма компонентами освітньої системи: спрямованість, нав-

чальний план, педагогіка, ІКТ, організація й управління, розвиток професіоналізму вчителя.

Для більшої наочності подамо проаналізовані структури загальнопрофесійних компетентностей вчителя інформатики у вигляді таблиці (табл. 1.3).

Таблиця 1.3

Структура загальнопрофесійної компетентності вчителя інформатики

№	Автор	Структура загальнопрофесійних компетентностей вчителя інформатики
1.	В. М. Гриньова [83, 25]	– предметні; – психологічні; – методичні; – ІКТ-компетентності
2.	С. О. Дружилов [92, 28]	компоненти: – мотиваційно-вольовий; – функціональний; – комунікативний; – рефлексивний
3.	М. І. Жалдак, Ю. С. Рамський, М. В. Рафальська [100, 8]	– дидактико-методичні; – організаційно-управлінські; – психолого-педагогічні; – дослідницькі; – комунікативні; – природничо-математичні
4.	А. Г. Кіріллов [124, 47]	– технологічні; – когнітивні; – психологічні; – регулятивні; – дослідницькі; – методичні
5.	Н. В. Кузьміна [137]	– методичні;

№	Автор	Структура загальнопрофесійних компетентностей вчителя інформатики
		– соціально-педагогічні; – диференціально-психологічні; – ауто психологічні
6.	О. Г. Ларіонова [145]	– інформаційно-методологічні; – методичні; – соціально-комунікативні; – особистісно-валеологічні
7.	О. В. Лебедєва [147]	– інформаційні; – загальнопедагогічні; – дидактичні; – методичні; – комунікативні; – психологічні
8.	А. К. Маркова [159, 82]	– педагогічна діяльність; – педагогічне спілкування; – особистість вчителя
9.	Л. М. Мітіна [165]	– методичні; – дидактичні; – комунікативні
10.	Н. Г. Ничкало [206, 8].	– методичні; – дидактичні; – комунікативні; – предметні
11.	Н. Ф. Радіонова, А. П. Тряпіцина [239]	– психологічні; – дидактичні; – комунікативні; – загальнопедагогічні;

№	Автор	Структура загальнопрофесійних компетентностей вчителя інформатики
		– навчальні
12.	В. О. Сластьонін [258, 30]	– психолого-педагогічні; – дидактичні; – організаційні; – комунікативні
13.	О. М. Соловова [267]	– методичні; – управлінські; – соціальні; – психолого-педагогічні; – ІКТ-компетентності
14.	О. М. Спірін [273, 212]	– загальнопрофесійні; – предметно-педагогічні; – технологічні (компетентності в галузі педагогічних технологій та інформаційно-технологічні компетентності)

Отже, підсумовуючи, в системі професійних компетентностей вчителя інформатики можна виокремити такі складові:

– ключові компетентності (адже сформованість ключових компетентностей надасть можливість майбутньому вчителю успішно діяти й жити в сучасному світі) [129]:

- навчальні;
- соціальні;
- загальнокультурні;
- здоров'язберігаючі;
- ІКТ-компетентності;
- громадянські;
- підприємницькі;

– загальнопрофесійні:

• методичні;

• науково-дослідницькі;

• психолого-педагогічні;

• організаційно-управлінські;

• комунікативні;

• ІКТ-компетентності вчителя;

– спеціальні професійні (інформатичні).

Розглянемо більш детально складові загальнопрофесійних компетентностей.

Методичні компетентності передбачають розуміння місця і значення методики навчання в професійній підготовці вчителя інформатики; знання основних компонентів методичної системи навчання інформатики в школі та їх взаємозв'язків у навчальному процесі; знання основних компонентів концепції навчання інформатики, а також програм і підручників, розроблених на їх основі; розуміння суті й призначення освітніх стандартів навчання; знання змісту стандартів з інформатики; володіння методикою навчання окремих тем і питань шкільного курсу інформатики; уміння використовувати програмну підтримку курсу і оцінювати її методичну доцільність; знання принципів диференціації навчання інформатики, володіння методикою навчання кількох профільних курсів інформатики, що відповідають спеціалізації освіти на старшому ступені в конкретній школі; уміння планувати навчальний процес з інформатики, вибирати форми організації і методи навчання, адекватні змістові матеріалу, що вивчається; знання функцій, видів контролю і оцінки результатів навчання, уміння розробляти і використовувати засоби перевірки, об'єктивно оцінювати знання і вміння учнів, коригувати методику навчання за результатами різних видів контролю; знання сучасних тенденцій у навчанні інформатики [197, 7].

Науково-дослідницькі компетентності передбачають готовність до науково-дослідницької діяльності; вміння формулювати питання, що виникають на практиці, у вигляді наукової проблеми, проводити цілеспрямоване спостереження, педагогічний експеримент; уміння прогнозувати результати дослідження,

вміння чітко, логічно викласти власні думки, вміння переконувати, аргументувати власну точку зору; здатність аналізувати психолого-педагогічну, наукову та методичну літературу з метою підвищення власного професійного рівня; створення власних педагогічних програмних засобів; написання статей, використання передового педагогічного досвіду.

Психолого-педагогічні компетентності передбачають наявність вмінь постановки та рішення педагогічних задач, вивчення педагогічної ситуації, об'єднання навчальних, розвиваючих та виховних задач, здійснення прогнозування, володіння педагогічними технологіями, виявлення зони найближчого розвитку, передбачення можливих та врахування типових ускладнень, врахування мотивації дітей при плануванні навчально-виховного процесу, робота як зі слабкими, так і з обдарованими дітьми, застосування диференційованого та індивідуального підходу до учнів, здійснення педагогічного самоаналізу; наявність таких якостей, як педагогічна ерудиція, педагогічне цілепокладання, педагогічне мислення, педагогічна інтуїція, педагогічна імпровізація, педагогічний оптимізм, педагогічна рефлексія; керування власним емоційним станом, надання йому конструктивного, а не деструктивного характеру; визначення особливостей власного стилю та використання позитивних природних даних; здійснення творчого пошуку; вміння бачити свою роботу в цілому, розуміння причинно-наслідкових відношень [159, 83].

Організаційно-управлінські компетентності передбачають організацію вчителем учнівського колективу, активну участь у роботі вчительського колективу, участь у роботі методичних семінарів, організацію позакласної (гуртки, екскурсії, вікторини, олімпіади та ін.) та самостійної роботи учнів, дотримання правил і норм охорони здоров'я в кабінеті інформатики, створення баз даних педагогічних матеріалів, бібліотечних та Інтернет-ресурсів, здійснення документообігу [100, 7].

Комунікативні компетентності передбачають демократичний стиль спілкування, вміння слухати, впливати не прямо, а опосередковано; виявляти інтерес до особистості співбесідника; наявність таких здатностей, як комунікабельність, тактовність; пошук нових задач та способів спілкування, взаємообмін відомостями, взаємопізнання один одного, взаємокорекція поведінки, володіння засобами

невербального спілкування, вміння приймати незалежну позицію при вирішенні конфліктів, виховання культури спілкування; риторичні вміння.

Незважаючи на те, що ІКТ-компетентності було виокремлено як ключові, в професійній діяльності педагога вони набувають нового змісту та особливостей, тому їх включено і до складу загальнопрофесійних компетентностей.

ІКТ-компетентності вчителя передбачають використання ІКТ в процесі навчання: для організації та управління навчальним процесом: використання технологій дистанційного навчання, сучасних методів і засобів комп'ютеризованого контролю знань; розробка електронних курсів; використання ІКТ для організації індивідуальної, самостійної роботи; професійного вдосконалення.

1.2 Спеціальні професійні компетентності вчителя інформатики

1.2.1 Структура спеціальних професійних компетентностей учителя інформатики. *Спеціальні професійні компетентності* відображають специфіку конкретної предметної, галузевої чи надпредметної професійної діяльності. Спеціальні компетентності можна розглядати як реалізацію ключових та базових компетентностей в області навчального предмету, конкретної галузі професійної діяльності [239].

В. О. Калінін визначає спеціальні професійні (предметні) компетентності як «освіченість та авторитетність учителя в галузі науки, представником якої він є, яка акумульована в цьому предметі і яким має оволодіти учень, щоб у самостійній діяльності ефективно вирішувати творчі завдання» [234, 136].

Визначення спеціальних професійних компетентностей вчителя інформатики, як бачимо, безпосередньо залежить від змісту навчання інформатики.

Враховуючи досить «молодий вік» (порівняно з іншими науками) та надзвичайно швидкий розвиток, зміст навчальної дисципліни «інформатика» до цих пір породжує численні дискусії. Тому питання про зміст навчання інформатики залишається відкритим.

В. Г. Кінельов зазначає, що в рамках інформатизації освіти відбувається зміщення акцентів на отримання фундаментальних знань, найбільш стабільних та

універсальних, а, відповідно, перше місце в підготовці майбутніх учителів інформатики мають зайняти загальнотеоретичні знання, які відрізняються множиною внутрішніх та зовнішніх зв'язків, які розкривають структуру змісту і визначають методологічну базу предметної області «інформатика», а саме – проблеми теоретичної інформатики [123].

Теоретична інформатика – це математична дисципліна, яка використовує методи математичного моделювання для обробки, передачі та використання повідомлень і створює цим самим «фундамент» інформатики [123, 23].

За М. В. Швецьким основними розділами фундаментальної підготовки в галузі інформатики є наступні: алгоритми; програмування; структури даних; комп'ютерна графіка і обчислювальна геометрія; архітектура ЕОМ; мови програмування: парадигми та організація; конструювання інтерпретаторів та компіляторів; бази даних та інформаційний пошук; штучний інтелект (програмно-прагматичний підхід); дискретна математика та теорія алгоритмів [142, 157].

Н. В. Морзе до змісту фундаментальної підготовки вчителя інформатики запропоновано внести такі розділи: теоретичні основи інформатики, теорія алгоритмів, структури даних, технологія розробки програмного забезпечення, архітектура комп'ютерних систем, парадигми програмування (функціональне, продукційне, хорновське, об'єктно-орієнтоване), комп'ютерна графіка, операційні системи, інформаційні системи, теоретичні основи баз даних, бази даних і інформаційний пошук, системи штучного інтелекту, комп'ютерне моделювання, аналіз і моделювання систем, дискретна математика, теоретичне програмування, соціальна інформатика, комп'ютерні комунікації і мережі, глобальна мережа Інтернет, гіпермедійний дизайн, програмна інженерія [202, 17].

У роботах [5, 16; 6, 14] в інформатиці виокремлюють такі галузі знань: дискретні структури, основи програмування, алгоритми і теорія складності, архітектура й організація ЕОМ, операційні системи, мови програмування, людино-машинна взаємодія, графіка і візуалізація, інтелектуальні системи, керування інформацією, соціальні та професійні питання програмування, програмна інженерія, методи обчислень, захист та безпека інформації, розподілені обчислення.

В документі [4, 7] до цього списку додано ще 5 нових галузей: мережі та зв'язок, платформено-зорієнтовані розробки, основи розробки програмного забезпечення, паралельні і розподілені обчислення, фундаментальні принципи комп'ютерних систем.

М. П. Лапчик відзначає, що «школі потрібен вчитель інформатики з фундаментальними знаннями в галузі інформатики. Причому об'єм цих знань несподівано … не лише став достатньо відчутним, але й має тенденцію до постійного (і досить енергійного) зростання» [144, 10].

Розглядаючи проблему структури та методичної системи підготовки кадрів інформатизації школи в педагогічних ВНЗ, М. П. Лапчик особливу увагу звертає на необхідність посилення профільної підготовки вчителя інформатики за рахунок математичної компоненти фундаментальної освіти, яка передбачає отримання освіти в галузі основ інформатики, математичного моделювання та формування фундаментальних основ теоретичної (математичної) інформатики [144, 11].

Таким чином, фундаменталізація інформатичної освіти зводиться до посилення математичної складової.

Тому, розглядаючи питання спеціальних професійних компетентностей учителя інформатики, необхідно включити до їх складу не лише інформатичні, але й компетентності з математичної інформатики та фундаментальних дисциплін.

Колектив авторів [100] у спеціальних професійних компетентностях учителя інформатики виокремлює:

– інформологічно-методологічні;

– інформаційно-технологічні;

– у галузі комп'ютерної інженерії;

– у галузі моделювання;

– та у галузі алгоритмізації і програмування.

Формування *системи інформатичних компетентностей майбутніх учителів інформатики* полягає в опануванні на достатньо високому рівні змістом фундаментальних та прикладних розділів інформатики, її основними методами з вра-

хуванням майбутньої професійної діяльності, набутті досвіду розв'язування задач професійного спрямування, опануванні методологією здійснення дослідницької діяльності у відповідній предметній галузі. Набуття спеціальних професійних компетентностей необхідних рівнів дасть змогу вчителям інформатики ефективно здійснювати свою педагогічну діяльність, продовжити навчання у магістратурі, аспірантурі, здобути освіту у галузі інформатики за іншою спеціальністю (наприклад, за спеціальністю інженера-програміста) [246, 52].

А. Г. Кіріллов [124, 49] в структурі інформатичних компетентностей учителя інформатики виокремлює компетентності:

– математичні;

– користувача;

– інформаційно-системні;

– технічні;

– з програмування.

Проблема розкриття змісту спеціальних професійних компетентностей розглядається в дослідженні В. С. Шелудька [307], який зауважує, що «характер видів діяльності вчителя інформатики і аналіз стандарту вчителя інформатики дозволяє визначити відповідні компетентності: користувацькі, технічні, математичні і програмувальні». Відповідно користувацькі компетентності відтворюють навички роботи з пристроями введення-виведення інформації, прикладним програмним забезпеченням загального і навчального призначення, роботу в глобальній мережі та інше. Технічні компетентності визначають вчителя, як комп'ютерного координатора, що очолює всю роботу по налагодженню і використанню комп'ютерів в навчальному процесі. Математичні компетентності відтворюють знання вчителя інформатики в галузі математики. Програмувальні компетентності характеризується широким спектром вмінь: логічного, алгоритмічного мислення, розробки програмного продукту, його налагодження, аналізу і тестування.

Згідно з О. М. Спіріним спеціальні професійні компетентності включають в себе:

– предметно-орієнтовані або профільно-орієнтовані компетентності:

• науково-предметні;

• предметно-педагогічні;

– професійно-практичні (ці компетентності слід розуміти як такі, якими повинен володіти випускник з позицій роботодавця – це є визначенням ступеня готовності випускника виконувати конкретні практичні роботи) [273, 221].

М. С. Головань інформатичні компетентності визначає як «інтегративне утворення особистості, яке інтегрує знання про основні методи інформатики та інформаційних технологій, уміння використовувати наявні знання для розв'язання прикладних задач, навички використання комп'ютера і технологій зв'язку, здатності представляти повідомлення і дані у зрозумілій для усіх формі і виявляється у прагненні, здатності і готовності до ефективного застосування сучасних засобів інформаційних та комп'ютерних технологій для розв'язання завдань у професійній діяльності і повсякденному житті, усвідомлюючи при цьому значущість предмету і результату діяльності» [72, 65]. В структурі інформатичних компетентностей він розглядає мотиваційний, когнітивний, діяльнісний, ціннісно-рефлексивний, емоційно-вольовий компоненти.

Мотиваційний компонент: прагнення і здатність (готовність) до отримання знань, умінь і навичок у галузі інформатики, комп'ютерної техніки та ІКТ; прагнення самостійно використовувати можливості комп'ютера як засобу інформаційної діяльності у навчальній та позанавчальній діяльності; прагнення вивчати нові досягнення в галузі інформатики; націленість на досягнення високого рівня інформатичної компетентності; мотивація досягнення успіху в професійній діяльності на основі використання інформаційних технологій, прагнення отримати визнання у своїх однокурсників, колег тощо [71, 320].

Когнітивний компонент: сукупність знань, що відображають систему сучасного інформаційного суспільства; знання, які складають інформативну основу пошукової пізнавальної діяльності; теоретичні знання про основні поняття та методи інформатики як наукової дисципліни; знання інформаційних технологій, їх можливостей для розв'язання професійних задач; здатність аналізувати інформаційні ресурси і виявляти їх можливості для розв'язання задач професійної діяль-

ності; проявляти креативність, гнучкість, критичність, системність, мобільність, оперативність мислення в ситуаціях пошуку та перетворення необхідних даних [71, 320].

Діяльнісний компонент: досвід пізнавальної діяльності, зафіксований у формі його результатів – знань у галузі інформатики; досвід здійснення відомих способів діяльності у формі умінь діяти за зразком; досвід творчої діяльності у формі умінь приймати ефективні рішення в проблемних ситуаціях; досвід здійснення емоційно-ціннісних відношень у галузі особистісних орієнтацій; уміння працювати з апаратним та програмним забезпеченням на рівні кваліфікованого користувача; уміння спілкуватися з використанням інформаційних засобів і технологій; уміння орієнтуватися в інформаційному середовищі [71, 320].

Ціннісно-рефлексивний компонент: сукупність особисто значущих і цінних прагнень, ідеалів, переконань, поглядів, ставлень до продукту і предмету діяльності у сфері інформаційних процесів і стосунків; розуміння інформатичної компетентності як однієї з провідних професійних і соціальних цінностей; адекватна самооцінка власних можливостей у використанні інформаційних технологій, інформаційних ресурсів, упевненість у їх виборі та реалізації; наявність власної позиції щодо застосування інформаційних технологій у професійній діяльності для розв’язання економічних задач; прагнення до самоактуалізації, саморозвитку, постійної роботи над собою у сфері інформаційних технологій; прагнення до професійного самовдосконалення на основі інформаційних технологій здатність адекватно орієнтуватися в інформаційних інноваціях; здатність брати на себе відповідальність за інформатизацію професійної діяльності; здатність до рефлексії у сфері пошуку та перетворення інформації, в опануванні та використанні інформаційних технологій; самоаналіз і самооцінка професійної діяльності на основі інформаційних технологій; здатність адекватно оцінювати власні досягнення в галузі інформатики, свій рівень інформатичних компетентностей; уміння визначати переваги і недоліки власних компетентностей в галузі інформатики та інформаційних технологій; уміння визначати резерви свого подальшого професійного зростання; уміння регулювати свою інформатичну діяльність і ставлення до неї [71, 321].

Емоційно-вольовий компонент: здатність розуміти власний емоційний стан в ситуації пошуку та перетворення потрібної інформації; здатність достойно переживати відсутність результату, технічні та інші збої у процесі роботи в інформаційному середовищі; здатність відкрито ділитися своїми почуттями і переживаннями щодо використання інформаційних технологій; цілеспрямованість дій в інформаційному середовищі; терпіння і володіння собою в ситуаціях пошуку та перетворення інформації за допомогою інформаційних технологій; наполегливість в опануванні знань у галузі інформатики і умінь у використанні нових інформаційних технологій у професійній сфері; наполегливість у досягненні цілей самоактуалізації та саморозвитку; прояв вольових зусиль у розв'язанні навчальних і професійних проблем; прояв ініціативності, сміливості, принциповості в розробці і здійсненні навчальних і професійних проектів на основі інформаційних технологій [71, 321].

Згідно документу [16] інформатичні компетентності діляться на чотири групи: інформаційні служби, мережні системи, програмування та розробка програмного забезпечення, засоби взаємодії. Основну частину документу [16] складають 49 модулів, які містять компетентності та їх складові з наступних галузей: основи інформаційних технологій, передавання даних, теорія програмування, мови прикладного програмування, розробка програмного забезпечення, основні графічні принципи дизайну, фото-, відео та звукова продукція, Інтернет, дизайн web-сторінки, мультимедіа продукція, апаратний дизайн та обслуговування, операційні системи, мережні архітектури, мережні операційні системи, глобальні мережі, управління мережею, основи систем управління баз даних, адміністрування баз даних, теорія інформаційних систем, управління інформаційними системами, інсталяція системи та її обслуговування, системне адміністрування та управління, управління проектом, спілкування, технічна документація, обслуговування клієнтів, економічні та ділові поняття, перевірка якості, статистика та ін. Причому для кожної з компетентностей визначено є вона обов'язковою або ж рекомендованою для будь-якої з чотирьох вищевказаних груп.

У відповідності до документу [1] інформатичні компетентності діляться на

дві групи: компоненти технічного вивчення та компоненти, що є для них необхідною основою. Для кожної з компетентностей документом надано її складові та показники, які визначають її сформованість. В якості показників можуть виступати завдання на виконання певної дії, що демонструє сформованість здатностей, які складають компетентності.

До компонентів технічного вивчення відносяться такі компетентності: комп'ютерні тенденції в бізнесі та суспільстві, принципи роботи персонального комп'ютера, Windows, бази даних, електронна пошта, графічне програмне забезпечення, Інтернет, електронні таблиці, текстовий редактор, редактор презентацій, встановлення та налаштування програмного забезпечення, встановлення та налаштування апаратних засобів, мережні технології, програмування.

Для зручності компетентності, що входять до складу компонент технічного вивчення, можна об'єднати в такі групи: компетентності користувача (або ж ІКТ-компетентності), програміста та адміністратора комп'ютерної системи.

Необхідною основою для вивчення технічних компонентів є такі компетентності: аналіз, дизайн/розробка, ділове спілкування, обслуговування клієнтів, подання матеріалу, розв'язування проблем, управління проектом, дослідження, самоосвіта, управління задачею, робота в групі, тестування/перевірка.

Згідно ГСВО фахівець з інформаційних технологій та викладач-стажист має володіти такими спеціалізовано-професійними компетентностями [68, 20]:

– знати та розуміти методи системного аналізу та теоретичної кібернетики щодо побудови інформаційних моделей об'єктів та процесів різної природи;

– знати математичні методи системного аналізу та кібернетики, методи математичного моделювання для побудови та аналітичного дослідження детермінованих та стохастичних моделей об'єктів і процесів інформатизації, моделі оптимізації, прогнозування, оптимального керування та прийняття рішень;

– знати сучасні методи розробки та оптимізації концепцій комп'ютерної реалізації моделей об'єктів і процесів інформатизації;

– знати математичні методи розробки та дослідження алгоритмів розв'язування задач моделювання об'єктів і процесів інформатизації, алгоритмів

функціонування інформаційних систем та методик оцінювання складових ефективності даних алгоритмів;

– знати методи побудови та верифікації абстрактної архітектури комп'ютеризованої системи та знати апаратні платформи та програмні середовища, що відповідають побудованій архітектурі;

– знати методи виявлення, формулювання, специфікації, аналізу та трасування вимог до комп'ютеризованих систем на етапі їх проектування, методи проектування та верифікації абстрактної архітектури комп'ютеризованих систем;

– знати основні парадигми проектування та мови моделювання програмного забезпечення комп'ютеризованих систем, методи планування життєвого циклу програмного забезпечення та розроблення моделі керування ресурсами;

– знати методи побудови концептуальної, логічної та фізичної моделей проектування систем керування базами даних;

– знати моделі подання знань, методи добування та структурування знань, логічного виведення для розроблення баз знань та інтелектуальних систем;

– знати основні протоколи Інтернет, моделі та структури Інтернет-серверів проектування інформаційних web-ресурсів з інтеграцією зовнішніх даних і програмних продуктів, з використанням методів захисту інформації;

– знати методи розробки проекту локальної комп'ютерної мережі на основі стандартних протоколів і інтерфейсів, планування мережної інфраструктури, програмного та апаратного забезпечення, розроблення логічної та фізичної моделей локальної комп'ютерної мережі, топологію структурованих кабельних систем, використовуючи методи захисту інформації;

– знати методи цифрового подання та обробки графічної, звукової та відео інформації, основи комп'ютерної графіки, методи проектування динамічних графічних об'єктів для програмних систем;

– знати методи, нормативи, державні стандарти та чинне законодавство стосовно організації, планування, контролю та управління роботами з проектування та розроблення комп'ютеризованих систем колективом розробників;

– знати базові методики викладання основ інформатики та математики для

професійно-технічної освіти нижчого рівня, ніж вища освіта;

– знати операційні системи (Windows, Unix тощо), системне програмне забезпечення, найбільш розповсюджені пакети прикладних програм, інформаційні портали Інтернет, програмні методи захисту інформації в комп'ютеризованих системах та мережах;

– знати базові та спеціалізовані технології розроблення програмного забезпечення комп'ютеризованих систем;

– знати методи, методики контролю та тестування правильності роботи програмного забезпечення комп'ютеризованих систем;

– знати методи та правила експлуатації та обслуговування системного та прикладного програмного забезпечення комп'ютеризованих систем.

Таким чином, у структурі спеціальних професійних компетентностей учителя інформатики можна виокремити компетентності:

– з теоретичної (математичної) інформатики;

– з програмування;

– з інформаційних технологій;

– з фундаментальних природничо-математичних дисциплін.

Причому ці компетентності складають інформатичні компетентності не просто сумарно, а мають певні взаємозв'язки. Так, набуття компетентностей з математичної інформатики приводить до якісних змін у рівні компетентностей з програмування, що, в свою чергу, приводить до змін у компетентностях з інформаційних технологій, і навпаки. Таким чином, компетентності з програмування відображають зв'язок між компетентностями з математичної інформатики та інформаційних технологій. Компетентності з фундаментальних природничо-математичних дисциплін мають взаємозв'язок з усіма іншими інформатичними компетентностями.

Концептуальну структуру системи професійних компетентностей учителя інформатики зображено на рис. 1.1.

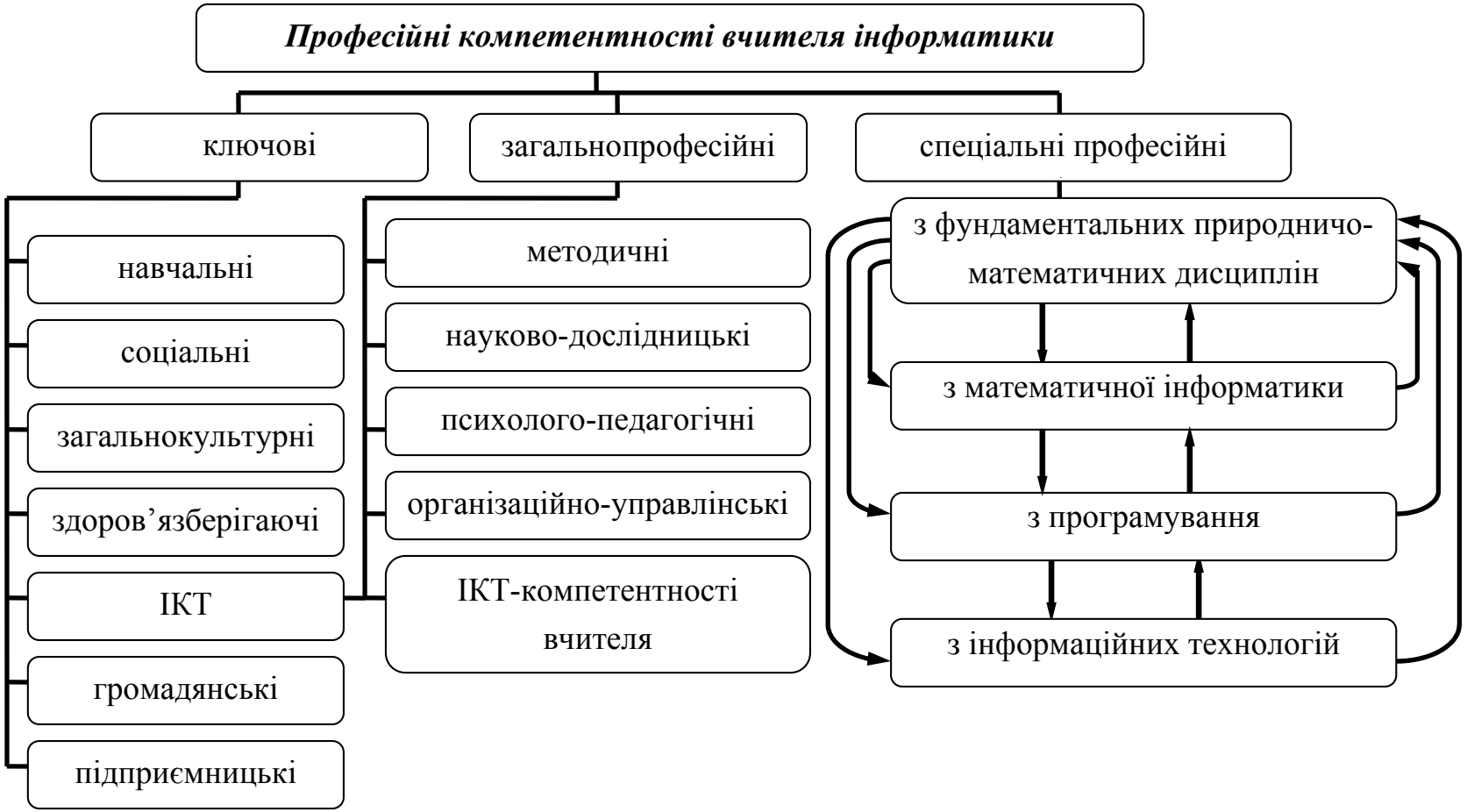


Рис. 1.1. Структура системи професійних компетентностей учителя інформатики

1.2.2 Компетентності з програмування. Згідно документу [1, 47] компетентності з програмування мають такі складові:

– уміння пояснити призначення та функції комп'ютерних програм;

– уміння пояснити термін «мови програмування» та навести приклади для кожної з різних парадигм програмування;

– уміння пояснити фактори, які необхідно розглянути при виборі мови програмування для розв'язування задачі;

– уміння описати етапи розробки програм;

– уміння пояснити та застосувати концепції програмування та інструментальні засоби, що використовуються в структурному програмуванні;

– уміння пояснити та проілюструвати лінійні, з розгалуженням та циклічні конструкції, що використовуються в структурному програмуванні;

– уміння пояснити відмінності використання різних парадигм програмування: подіє-орієнтованого програмування, об'єктно-орієнтованого програмування та імперативного програмування;

– уміння створити внутрішню та зовнішню документацію до програми;

– уміння спроектувати, написати, перевірити та дослідити результати виконання програм.

Як бачимо, в цьому переліку присутні лише елементи, що, згідно нашого визначення компетентності, утворюють тільки її праксеологічну складову. Тобто, існує необхідність виокремлення й інших складових компетентностей з програмування.

Компетентності з програмування, як і будь-які компетентності, мають такі взаємопов'язані складові (згідно нашого визначення):

– *когнітивно-змістову* (гносеологічну) – знання основних форм для керування виконанням програми; знання простих типів даних та функцій для роботи з ними; знання похідних типів даних, способів їх утворення з простих типів даних, функцій для роботи з ними та пріоритетних напрямів їх використання; знання основних етапів розв'язування прикладних задач; знання основних етапів проектування програм; знання складових мови програмування;

– *операційно-технологічну* (праксеологічну) – вміння пояснити призначення та функції існуючої програми, знайти помилки в логіці розв'язання задачі, описати етапи розробки програм, розробити функції та обґрунтувати пріоритетність використання того чи іншого виразу для їх створення, створити документацію до програми, пояснити та продемонструвати процес створення похідних типів даних, спроектувати, описати, перевірити та проаналізувати результати виконання програми; оцінити переваги різних способів розв'язання однієї задачі; вміння обирати засоби для розв'язання задачі та обґрунтовувати свій вибір; уміння використовувати можливості обраних засобів (довідка, налагодження програми, налаштування необхідних параметрів та ін.);

– *ціннісно-мотиваційну* (*аксіологічну*) – емоційно-ціннісне ставлення до процесу розробки, опису, налагодження, тестування програм та аналізу результатів їх роботи; уміння знаходити нові, нестандартні рішення задачі; внутрішня мотивація до опанування програмування; готовність до активного застосування гносеологічних та праксеологічних складових у практичній діяльності; прагнення до самовдосконалення, потреба у саморозвитку гносеологічних та праксеологічних складових; уміння самостійно приймати рішення, критично ставитись до чужих впливів, здатності за власним почином організовувати діяльність, ставити мету, в разі необхідності вносити в поведінку зміни; вміння постійно і тривало домагатися мети; наполегливість у досягненні мети, прагнення до поліпшення отриманих результатів, незадоволеність досягнутим, намагання домогтися успіху; внутрішня потреба у створенні програмних продуктів;

– *соціально-поведінкову* – здатність до співпраці у процесі розробки, опису, налагодження, тестування програм та аналізу результатів їх роботи, використання засобів для організації спільної роботи над проектом; відповідальність за власну поведінку, за виконання завдань; комунікабельність; здатність до адаптації; схильність до дискусії.

Кожна складова має різні «вагові внески» в загальний рівень сформованості компетентностей з програмування (визначалась методом експертних оцінок, описаним в п. 2.4).

Процес формування тієї чи іншої компетентності може бути досить тривалий і здійснюватися під впливом різних факторів (навчання у закладах освіти, професійна діяльність, міжособистісне спілкування), тому говорити про наявність у студентів певної компетентності некоректно. Далі, говорячи про набуття студентами певних компетентностей, будемо розуміти їх сформованість на *певному* рівні [246, 24].

Таким чином, процес формування компетентностей з програмування є перехід від одного рівня (нижчого) до іншого (вищого), а для оцінювання рівня їх сформованості необхідно виокремити рівні їх сформованості та критерії, які б надали можливість здійснювати перевірку рівня сформованості компетентностей.

Аналіз літератури надав можливість зробити висновок, що не існує одностайної думки стосовно кількості рівнів сформованості компетентності.

Так, О. М. Спірін виокремлює шість основних рівнів (початковий, мінімально-базовий, базовий, підвищений, поглиблений та дослідницький) [272]. Схожої думки і В. А. Сухомлін, він також виокремлює шість рівнів (0 – рівень відсутності знань, 1 – рівень ознайомлення, 2 – рівень технічної грамотності, 3 – рівень розуміння концепцій/здатності використання, 4 – поглиблені знання/використання в додатках, 5 – рівень експерта) [278]. Ряд вітчизняних дослідників (О. Є. Антонова, О. А. Дубасенюк, В. М. Єремєєва, М. В. Левківський, Н. Г. Сидорчук та ін.) співвідносять запропоновані Б. Блумом категорії цілей з рівнями сформованості компетентностей: знання – репродуктивний рівень, розуміння – адаптивний рівень, застосування – конструктивний рівень, аналіз – творчий рівень, синтез – дослідницький рівень, оцінювання – оцінно-узагальнюючий рівень [273, 210].

Є. М. Смирнова-Трибульска [264, 130] виділяє дев’ять рівнів, що згруповані у три блоки: 0-2 – базовий (елементарний) рівень, 3-5 – середній (функціональний) рівень, 6-8 – просунутий (системний) рівень.

М. С. Головань розвиток компетентності у студентів розглядає як «незворотну, закономірну, цілеспрямовану зміну внутрішньої структури інформатичної компетентності і зовнішніх форм її прояву, у результаті чого виникають нові багаторівневі якісні її стани, основою яких є діалектична єдність можливого і дійс-

ного, а також як саморегульований процес, тобто внутрішньо необхідний рух, «саморух» від наявного рівня компетентності до вищого відповідно до стадій цього процесу. Розвиток компетентності як системи забезпечується кількісними, якісними і структурними перетвореннями її елементів у процесі зміни стадій руху – становлення, активного розвитку, саморозвитку» [76, 89]. У формуванні компетентності студентів М. С. Головань виділяє три рівні (низький, середній, високий) та три стадії (становлення, активного розвитку, саморозвитку). Стадії «горизонтального» просування (становлення, активного розвитку, саморозвитку) відбивають кількісне накопичення «критичної маси» суб'єктивних характеристик компетентності в кожного студента. «Вертикальне» просування – це якісний стрибок як перехід на вищий рівень розвитку [76, 91].

На стадії становлення відбувається засвоєння студентами знань, вироблення умінь на репродуктивному рівні, формування мотивації до вивчення предмету та позитивного ставлення до програмування. На стадії активного розвитку студенти осмислено оперують уміннями та знаннями, мають потребу в особистій самореалізації в інформаційному середовищі, мають такі розвинені якості, як рефлективність, креативність, критичність мислення, мають сформовані навички саморегуляції діяльності. Основна мета стадії саморозвитку – розвиток самостійності, творчої активності, самоорганізації та самоуправління, актуалізація потреби у саморозвитку [75, 19].

О. С. Меняйленко, Г. В. Монастирна [164], як і М. В. Жукова [95, 9] виділяють чотири рівні сформованості компетентності (низький, середній, достатній та високий).

Схожої думки і Я. Б. Сікора – вона розглядає такі рівні сформованості компетентності: адаптивний (низький), алгоритмічний (середній), частково-пошуковий (достатній), творчий (високий) [255, 11].

Адаптивний рівень характеризується недостатньою сформованістю професійних намірів, відсутністю необхідних знань та вмінь, репродуктивним виконанням діяльності, неадекватною самооцінкою.

Для алгоритмічного рівня характерним є епізодичний інтерес до професії,

недостатнє вміння використовувати наявні знання, нестійка потреба у самовдосконаленні.

Частково-пошуковий рівень відрізняється розвиненою суб'єктною позицією, яка виявляється в усвідомленості своїх дій та можливостей, прагненні до прийняття рішень, внесенні змін при використанні запозиченого досвіду, наявністю інтересу до професії, розумінням її значущості, проте недостатньою чіткістю у визначенні цілей формування компетентності.

Творчий рівень характеризується сформованістю стійкого інтересу до професії; здатністю до нестандартного розв'язання завдань, поглиблення знань та прийняттям усвідомлених рішень з урахуванням прогнозування наслідків своїх дій; прагненням до самовираження, самовдосконалення, об'єктивної самооцінки в професійній діяльності; володінням способами самодіагностики і саморозвитку [255, 11].

В. П. Беспалько пропонує впорядковану структуру, що відображає процес оволодіння певною діяльністю, і також виділяє чотири рівні засвоєння: ознайомлення, репродукції, умінь і навичок, трансформації. Кожному рівню відповідають певні дії, які повинен виконувати студент: 1) розпізнання, розрізнення; 2) відтворення, порівняння й аналіз; 3) використання на практиці до заданого класу явищ і об'єктів для завдань, що потребують буквального застосування знань і дій (дія за прикладом); 4) використання засвоєної інформації для практичних завдань поза тим класом явищ і об'єктів, на яких формувалися знання (дія з широким перенесенням) [48, 46].

С. О. Дружилов у загальному випадку в моделі навчання виділяє чотири стадії сформованості компетентності, що характеризують процес навчання, починаючи від стадії початкового знайомства з новим матеріалом (знаннями, концепціями, навичками) і закінчуючи стадією сформованої компетентності [91, 33].

Перша стадія: несвідома некомпетентність – у людини немає необхідних знань, умінь, навичок, і вона не знає про їх відсутність або взагалі про можливі вимоги щодо них для успішної діяльності. Ця стадія характеризується наступною професійної самооцінкою: «Я не знаю, що я не знаю». Коли людина усвідомлює

відсутність знань, умінь, навичок, необхідних для даної діяльності, вона переходить на другу стадію.

Друга стадія: свідома некомпетентність – людина усвідомлює, що їй не вистачає професійних знань, умінь, навичок. Тут можливі два результати а) конструктивний (як форма прояву особистісної та професійної активності) і б) деструктивний (форма соціальної пасивності). Конструктивний шлях означає, що усвідомлення суб'єктом своєї професійної некомпетентності сприяє підвищенню його мотивації на здобуття відсутніх професійних знань, умінь, навичок. Деструктивний результат може призводити до виникнення почуття невпевненості у своїх силах, психологічного дискомфорту, підвищеної тривожності та ін., що заважає подальшому професійному навчанню. Для другої стадії характерна наступна професійна рефлексія суб'єкта: «Я знаю, що я не знаю».

Третя стадія: свідома компетентність – людина знає, що входить в структуру і становить зміст її професійних знань, умінь і навичок і може їх ефективно застосовувати. Для третьої стадії характерна професійна самооцінка суб'єкта в такій формі: «Я знаю, що я знаю».

Четверта стадія: несвідома компетентність – коли професійні навички повністю інтегровані, вбудовані в поведінку; професіоналізм є частиною особистості. Несвідома компетентність характеризує рівень майстерності [91, 33].

Таким чином, дослідники виділяють, як правило, 4 рівні сформованості компетентності: низький, середній, достатній та високий. У дослідженні ми розглядатимемо процес формування компетентностей з програмування як перехід між цими рівнями: низький → середній → достатній → високий.

Низький рівень характеризується негативним ставленням до процесу розробки, опису, налагодження, тестування програм та аналізу результатів їх роботи; поверхневими, несистемними знаннями з програмування, відсутністю вмінь. *Середній рівень* відзначається індиферентним ставленням до процесу розробки, опису, налагодження, тестування програм та аналізу результатів їх роботи; слабкою мотивацією до опанування програмуванням; середніми за об'ємом, фрагментарними знаннями, наявністю окремих, розрізнених вмінь. *Достатній рівень* перед-

бачає виявлення інтересу до процесу розробки, опису, налагодження, тестування програм та аналізу результатів їх роботи; упорядкованими, структурованими знаннями, достатніми вміннями; проявленням здатності до співпраці у процесі програмування, використанням засобів для організації спільної роботи над проектом; здатністю до самонавчання. *Високий рівень* характеризується позитивним ставленням до процесу розробки, опису, налагодження, тестування програм та аналізу результатів їх роботи; стійкими, ґрунтовними знаннями, творчим підходом, уміннями до нестандартного розв'язання завдань, умінням відстоювати власну думку, постійною здатністю до співпраці у процесі програмування, використанням засобів для організації спільної роботи; здатністю до самонавчання.

1.3 Функціональний підхід до формування компетентностей з програмування

1.3.1 Основні підходи до навчання програмування. Для початку визначимо основні поняття, які будуть використані при розгляді даного питання: програмування, мова програмування, синтаксис та семантика мови програмування, парадигма програмування, технологія програмування та підходи до навчання програмування.

Програмування – 1) розробка програм розв'язування різноманітних задач на ЕОМ. В широкому розумінні програмування – весь процес створення програми (аналіз вимог, розробка проекту та реалізація програми), у вузькому розумінні – складання та налагодження програми.

2) Наука, що займається розробкою методів і засобів формалізації процесів обробки інформації, подання їх у вигляді алгоритмів та отримання програм для ЕОМ [130, 420].

Мова – це сукупність засобів для фіксації повідомлень і передавання їх від джерела інформації до споживача. Алгоритмічні мови – мови, призначені для фіксації алгоритмів у вигляді деяких повідомлень чи послідовностей повідомлень споживачеві інформації (виконавцеві алгоритму).

Мова програмування – алгоритмічна мова, призначена для фіксації алгоритмів, розрахованих на виконання ЕОМ [97, 129].

Окреме повідомлення про деяку операцію, яку повинна виконати машина, називають *вказівкою*, *командою* чи *виразом*.

Сукупність знаків і правил, за допомогою яких описуються алгоритми, утворюють алгоритмічну мову.

При конструюванні алгоритмічної мови виходять з деякого набору операцій, які вважаються елементарними і типовими для об'єктів певного вигляду. При цьому важливо вибрати форму, зручну як для того, хто складає правила обробки повідомлень, так і для того, хто буде читати і розуміти це правило, тобто форму, яка відповідає людським поняттям і уявленням. Найбільш універсальні структурні й операційні поняття намагаються використовувати в якомога меншій кількості. Ці поняття становлять змістовну частину (*семантику*) алгоритмічної мови. Сукупність зовнішніх законів форми опису алгоритмів утворює *синтаксис* алгоритмічної мови [97, 129].

Парадигма – 1) сукупність філософських, загальнотеоретичних і метатеоретичних основ науки; 2) той чи інший приклад або показовий випадок концепції чи теоретичного підходу [58, 704].

Парадигма програмування – це сукупність ідей та понять, через які визначається стиль написання програми. Парадигма, в першу чергу, визначається базовою програмною одиницею. В сучасній індустрії програмування дуже часто парадигма програмування визначається набором інструментів програміста, а точніше, мовою програмування і бібліотеками, що використовуються [80, 83].

Відомо кілька основних парадигм програмування, найважливішими з яких на даний момент є парадигми імперативного, декларативного (функціонально-логічного), об'єктно-орієнтованого та подіє-орієнтованого програмування.

Імперативне (інколи визначають як директивне) програмування – це парадигма програмування, згідно з якою процес обчислень описується у вигляді інструкцій, що змінюють стан програми. Імперативна програма – це послідовність команд, які повинен виконати комп'ютер. Мови програмування імперативної парадигми: C, Pascal, Basic, Fortran, Algol, Cobol та ін.

На противагу імперативній парадигмі в *декларативній* (функціонально-

логічній) парадигмі програмування програма подається не у вигляді інструкцій, не в описі того, як треба знайти результат, а в описі того, що треба знайти. Так, у *функціональній* парадигмі програмування програма – визначення функцій (або через інші функції, або ж рекурсивно). ФМП: Lisp, CommonLisp, Scheme, Haskell, ML, Hope, Miranda, R та ін. В *логічній* парадигмі програмування програма – опис задачі і основні правила її розв'язання. Найвідоміша мова логічного програмування – Prolog.

В *об'єктно-орієнтованій* парадигмі первинним поняттям є поняття об'єкту, що об'єднує в собі поля (дані) та методи (дії, що виконуються об'єктом). Для розв'язування задачі програміст створює такі об'єкти, взаємодія яких після запуску програми призведе до досягнення результату. Найвідоміші мови ООП: C++, Java, Objective-C, C#, Delphi, VB.NET, Smalltalk, PHP та ін.

В *подіє-орієнтованій* парадигмі програмування виконання програми визначається подіями – діями користувача (клавіатура, миша та ін.), повідомленнями інших програм, подіями операційної системи. Різні мови програмування підтримують подіє-орієнтоване програмування в різній мірі. Найбільш повна його підтримка у мовах програмування Smalltalk, Java, ActionScript, C#.

Парадигмою програмування визначається, в яких термінах програміст описує логіку програми. Слід зазначити, що парадигма програмування не визначає однозначно мову програмування. Багато сучасних мов програмування є мультипарадигменними – в них допускається використання різних парадигм [80, 85].

Крім парадигми програмування важливою є технологія програмування.

Технологія програмування – це система методів, засобів і прийомів розробки і налагодження програм. Технологія програмування розуміється як технологія розробки програмних засобів, включаючи всі процеси від моменту зародження ідеї програмного засобу [80, 85].

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

В основі *структурної технології* програмування лежить подання програми у вигляді ієрархічної структури блоків. Згідно з даною технологією:

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

– фрагменти програми, що повторюються, можуть оформлюватися у вигляді підпрограм;

– розробка програми відбувається покроково, за методом «згори донизу».

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

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

Підхід до навчання програмування визначається за педагогічними аспектами, що висуваються на перший план: «перш за все – програмні інструкції», або «перш за все – об'єкти», «перш за все – функції» і т. д.

Існують різні реалізації вступних курсів з програмування [6], здебільшого виокремлюють такі 6 підходів:

– імперативний;

– об'єктний;

– функціональний;

– з максимальним охопленням матеріалу;

– алгоритмічний;

– апаратний.

Автори [6] наголошують, що критерієм для виокремлення саме цих підходів слугувало те, що підхід має бути підтриманий людьми, які з успіхом використовували цей підхід і при цьому не є його творцями, адже завдяки енергії та ентузіазму авторів, будь-які «освітні експерименти приречені на успіх» – справжня ж перевірка полягає в відтворенні успіху при використанні цього підходу іншими.

Імперативний, об'єктний та функціональний підходи можна об'єднати в

один – *з орієнтацією на програмування* [6, 22]. Цей підхід має спільні як недоліки, так і переваги.

Недоліки:

– концентрація на програмуванні дає студентам обмежене розуміння дисципліни (за рахунок виключення інших питань);

– теоретичні теми, які повинні закріплювати розуміння студентами практичного матеріалу, відкладаються на пізніші етапи навчання, коли вони вже не мають такого безпосереднього значення;

– курси з програмування досить часто сконцентровані на синтаксисі та особливостях мови програмування. Це зумовлює концентрацію уваги студентів на відносно неважливих деталях, ніж на базових алгоритмічних навичках;

– вхідні курси з програмування досить часто занадто спрощують процес програмування, щоб зробити його доступним для початківців, як результат – проектуванню, аналізу та тестуванню приділяється недостатньо уваги;

– курси інтенсивного навчання програмування створюють у студентів, які не мали великого досвіду роботи з комп'ютером, що ті, хто раніше мав комп'ютерні навички, знають значно більше за них. Як результат – студенти-новачки просто не справляються з обсягом відомостей, в той час, як студенти з деякими попередніми знаннями досить часто працюють в півсили.

– використання такого підходу може переконати студентів, що написання програми – єдиний підхід до вирішення проблем з використанням комп'ютера. Однак останнім часом потужність та функціональність прикладних програм суттєво збільшились, і студентам необхідно усвідомлювати, що ці засоби можуть бути ефективними інструментами розв'язування задач без використання класичного програмування.

Переваги:

– знання програмування є необхідною умовою для багатьох поглиблених курсів з інформатики;

– багатьом студентам програмування подобається більше, ніж інші аспекти інформатики.

Розглянемо детальніше виокремлені підходи.

Імперативний підхід є найбільш традиційним з усіх перелічених підходів, він передбачає вивчення однієї із процедурних мов програмування [6, 29]. При цьому підході можна використовувати і одну з об'єктно-орієнтованих мов для навчання основ програмування. В такому випадку, відмінність цього підходу від об'єктно-орієнтованого, полягає в акцентуванні та порядку проходження тем: якщо навчання ведеться з використанням об'єктно-орієнтованої мови, то на першому етапі увага повинна концентруватися на імперативних аспектах цієї мови, а саме: виразах, структурах керування, процедурах і функціях, а також інших ключових елементах традиційної процедурної моделі. Технології об'єктно-орієнтованого проектування вивчаються на наступному етапі.

До недоліків імперативного підходу відносять те, що студенти одержать менше досвіду з використання технології ООП, ніж це можливо при використанні моделі «з орієнтацією на ООП». Водночас, студентам необхідне розуміння традиційного імперативного (процедурного) стилю програмування, який є невід'ємною частиною ООП.

Об'єктний підхід також концентрується на програмуванні, але при цьому із самого початку робиться акцент на принципах об'єктно-орієнтованого проектування і програмування [6, 30].

Головною перевагою «об'єктного підходу» до вступних курсів є раннє ознайомлення з ООП, яке стало надзвичайно важливим як для освіти, так і для промисловості. Але водночас цей підхід не позбавлений недоліків, спільних для підходів «з орієнтацією на програмування». І в цьому випадку вони можуть лише посилитись, оскільки більшість мов, які використовуються для ООП в індустрії (зокрема, C++ і Java) набагато складніші, ніж класичні мови. І якщо викладачі не докладуть особливих зусиль для обмеження складності матеріалу, то деталі обраної мови можуть затінити суть ООП.

Функціональний підхід вперше був використаний в Массачусетському технологічному інституті в 1980-х роках і характеризується використанням ФМП (Lisp, Scheme) [6, 30]. Порівняно з іншими, цей підхід має такі переваги:

– використання мови, що не вивчається в середніх навчальних закладах, зменшує негативний ефект від різниці в початковій підготовці студентів з інформатики;

– нескладний синтаксис функціональних мов надає можливість викладачу акцентувати увагу на фундаментальних принципах і питаннях програмування;

– використання рекурсії, функцій як даних і зв'язаних структур даних природним чином вписуються в такий підхід, що дає можливість вивчати ці питання на самому початку навчання основ програмування.

Проте і в цьому підході є певні недоліки. Зокрема, цей підхід, як правило, вимагає від студентів достатньо високого рівня абстрактного мислення на більш ранній стадії навчання в порівнянні з використанням традиційних мов програмування.

У Додатку А наведено список навчальних закладів та назви курсів, які використовують ФП до навчання основ програмування (мова програмування – Scheme).

Підхід із максимальним охопленням матеріалу дає студентам більш цілісний погляд на дисципліну [6, 31].

Найбільш часто даний підхід реалізується у вигляді вступного оглядового курсу, в якому студенти отримують уявлення про цілий ряд цікавих та важливих тем. Після цього курсу студенти починають «звичайний» вступний курс. Перевагою використання такого курсу є те, що студенти одержують більш широке уявлення про програмування та інформатику.

Недоліки підходу з максимальним охопленням матеріалу:

– розгляд широкого спектру теоретичних питань з програмування на ранній стадії навчання, який передбачає такий підхід, ускладнює засвоєння навчального матеріалу студентами молодших курсів;

– до вступних профільних курсів додається іще один;

– вступний курс з програмування відкладається на наступний семестр.

Алгоритмічний підхід при вивченні основних концепцій програмування передбачає використання псевдокоду замість реальної мови програмування [6, 32].

Цей підхід мінімізує зусилля, що витрачаються на вивчення специфічних синтаксичних конструкцій конкретної мови програмування. Натомість, від студентів вимагається обґрунтування і роз'яснення алгоритмів, які вони будують. Після того, як студенти опанують основні типи алгоритмів і типи даних, вони можуть починати використовувати одну з мов програмування. Завдяки виключенню із навчальної програми часу, відведеного на вивчення синтаксису й особливостей певного середовища програмування, вступні курси, побудовані на основі цього підходу, можуть містити додаткові теоретичні теми. В результаті студенти починають знайомитися з відповідними аспектами теорії програмування з найперших днів навчання.

Водночас, підхід з орієнтацією на алгоритми має ряд недоліків: курси, що зводяться до конструювання алгоритмів у псевдокоді, викликають у студентів розчарування і знижують мотивацію навчання; орієнтація на псевдокод не припускає необхідності демонструвати роботу налагоджених текстів програм, хоча оволодіння навичками налагоджування програм необхідне для подальшої успішної роботи студентів і тому бажано, щоб вони практикувалися в цьому при навчанні якомога раніше.

Апаратний підхід передбачає навчання основ інформатики і, зокрема, основ програмування, починаючи з машинного рівня [6, 33]. Зазначається, що таким чином студенти можуть вивчити інформатику більш глибоко і послідовно. Тільки після створення у студентів розуміння структурних особливостей апаратної складової, машинної логіки і математики, курс переходить до розгляду програмування мовами високого рівня. З погляду розробників, такий підхід більш придатний для студентів, що вважають за краще розуміти процес обчислення на ЕОМ у всіх його деталях.

Як вважають розробники документа [6], цей підхід мало ефективний для того, щоб заохочувати студентів бачити цілісні концепції, що стоять за механізмом реалізації. Підхід з орієнтацією на апаратну складову також не дуже добре узгоджується із сучасною тенденцією все більшого вдосконалення віртуальних машин, що відділяють процес програмування від апаратних засобів. Цей підхід мо-

же бути використаний при підготовці фахівців із проектування комп'ютерної техніки, коли необхідно досягнути раннього ознайомлення з апаратною частиною обчислювальної системи.

Проведемо аналіз документів СС 2001 [6], СС 2008 [5] та СС 2013 [4]. У документах [5; 6] вивчення тем, що пов'язані з програмуванням, передбачалося в галузях «Вступ до програмування» (38 год. у 2001 році та 47 год. у 2008 році) та «Мови програмування» (21 год. і в 2001 році, і в 2008 році), а навчальні теми поділено на обов'язкові та за вибором. Вивчення теми «Функціональне програмування» у 2001 та 2008 роках передбачалося лише за вибором [5, 40; 6, 17].

В документі [4, 25] теми розділені на рівень-1, рівень-2 та за вибором. Причому зазначається, що всі теми рівня-1 є обов'язковими для студентів всіх інформатичних спеціальностей та повинні бути опрацьовані протягом перших двох років навчання, оскільки переважна більшість інших тем з інформатики передбачає наявність знань з цих тем в якості вхідних знань. Щонайменше 80-90 % тем рівня-2 також мають бути засвоєні усіма студентами інформатичних спеціальностей. Тим не менше, у вхідних курсах повинні розглядатись не лише теми, що зазначені в рівні-1, але й також теми з рівня-2 та за вибором. У 2013 році в галузі «Мови програмування» (28 год.: 8 рівня-1 та 20 рівня-2) на вивчення теми «Функціональне програмування» передбачено 7 год. (3 + 4).

Для порівняння: на вивчення теми «Об'єктно-орієнтоване програмування» у 2001 році передбачалось 10 год. (галузь «Мови програмування»), у 2008 році – 8 год. у галузі «Основи програмування» та 10 год. у галузі «Мови програмування», у 2013 році – в галузі «Мови програмування» 10 год. (4 + 6) [4, 129].

Для більшої наочності подамо ці дані у вигляді таблиці (табл. 1.4).

Таблиця 1.4

Відповідність між темами та роками

Тема	2001	2008	2013	
			рівень-1	рівень-2
Об'єктно-орієнтоване програмування	10	8 + 10	4	6
Функціональне програмування	–	–	3	4

Таким чином, спостерігається тенденція до збільшення кількості годин на вивчення функціонального програмування.

1.3.2 Математичні основи функціонального підходу в програмуванні. Окрім наведених вище переваг, ФП до програмування має фундаментальну математичну основу. З метою активізації пізнавальної діяльності студентів в процесі навчання математичної інформатики було запропоновано використати міжпредметні зв'язки математичної логіки та програмування. Так, в [142, 168] встановлено відповідність між формальними численнями та фундаментальними розділами курсу інформатики (табл. 1.5). Зокрема, функціональне програмування поставлене у відповідність λ-численню та численню комбінаторів.

Починаючи з 2009/2010 н.р., навчання основ програмування студентів спеціальності «Інформатика» на фізико-математичному факультеті КДПУ (КНУ) відбувається із застосування ФМП Scheme, оскільки її математичні основи суттєво покращать розуміння студентами λ-числення та числення комбінаторів. Серед ФМП мову Scheme обрано виключно завдяки її простим синтаксису та семантиці.

Розділ «Формальні системи» зазвичай починається з розгляду питання означення формальної системи та визначення її складових.

Формальна система (числення) – це математична модель, яка задає множину дискретних об'єктів шляхом опису «исходных» об'єктів і правил утворення нових об'єктів з початкових та уже утворених. Об'єкти формальної системи складаються з неподільних елементів різних видів [118, 93].

Згідно з [301, 48] для побудови формальної мови необхідно скористатись якоюсь уже відомою мовою та в термінах цієї мови створити алфавіт і сформулювати правила формальної мови. Алфавіт утворюється шляхом задання символів, що називаються «исходными» символами мови і є неподільними за таких умов:

а) при утворенні мови ніколи не використовуються їх частини;

б) будь-яку скінченну послідовність «исходных» символів можна утворити єдиним способом з елементів множини «исходных» символів.

Скінченна послідовність «исходных» символів називається формулою. Формули утворюються згідно з визначеними правилами. Деякі з формул оголошу-

ються аксіомами. І, нарешті, встановлюються правила виведення (або правила дії, або правила перетворення), згідно з якими з формул безпосередньо виводиться або безпосередньо слідує як висновок деяка формула. Скінченна послідовність, яка складається з однієї чи більшого числа формул називається доведенням, якщо кожна формула в послідовності є або аксіомою, або виводиться безпосередньо згідно з одним із правил виведення з попередньо утворених формул послідовності.

Визначаючи формальну систему, необхідно додати ще вимоги ефективності її визначення [301, 50]:

1) задання «исходных» символів повинно бути ефективним в тому сенсі, що повинен існувати метод, який завжди дозволяє ефективно визначити, чи є деякий символ одним з «исходных» символів послідовності;

2) визначення формули повинно бути ефективним в тому сенсі, що повинен існувати метод, який завжди дозволяє визначити, чи є деяка формула правильно утворена;

3) задання аксіом повинно бути ефективним в тому сенсі, що повинен існувати метод, який завжди дозволяє ефективно визначити, чи є формула аксіомою;

4) правила виведення повинні бути ефективними в тому строгому сенсі, що повинен існувати метод, який завжди дозволяє ефективно визначити, чи виводиться безпосередньо згідно з правилами виведення.

Графічне подання складових формальної системи (відповідно до [142, 168]) показано на рис. 1.2.

В табл. 1.5–1.8 згідно з названими правилами встановлено відповідність між λ-численням і ФМП Scheme.

У ФМП Scheme використовується префіксна нотація (прямий польський запис). Тобто, інфіксний запис 3 + 4 в префіксній нотації матиме вигляд (+ 3 4). Scheme, як і більшість ФМП, працює зі списками, що позначаються круглими дужками. Крайній зліва елемент списку – функція, інші елементи – аргументи функції. Тобто, звичний запис f(x, y) має вигляд (f x y). Особливості префіксної нотації:

– префіксна форма запису виразу може містити більше, ніж 2 аргументи:

```
(+ 2 3 4); (* 5 6 9);
```

– вона природно розширюється, в результаті чого отримуються вкладені списки (композиції функцій), елементами яких також є списки:

```
(+ (– 6 3 4)
   (* 4 5 2)).
```

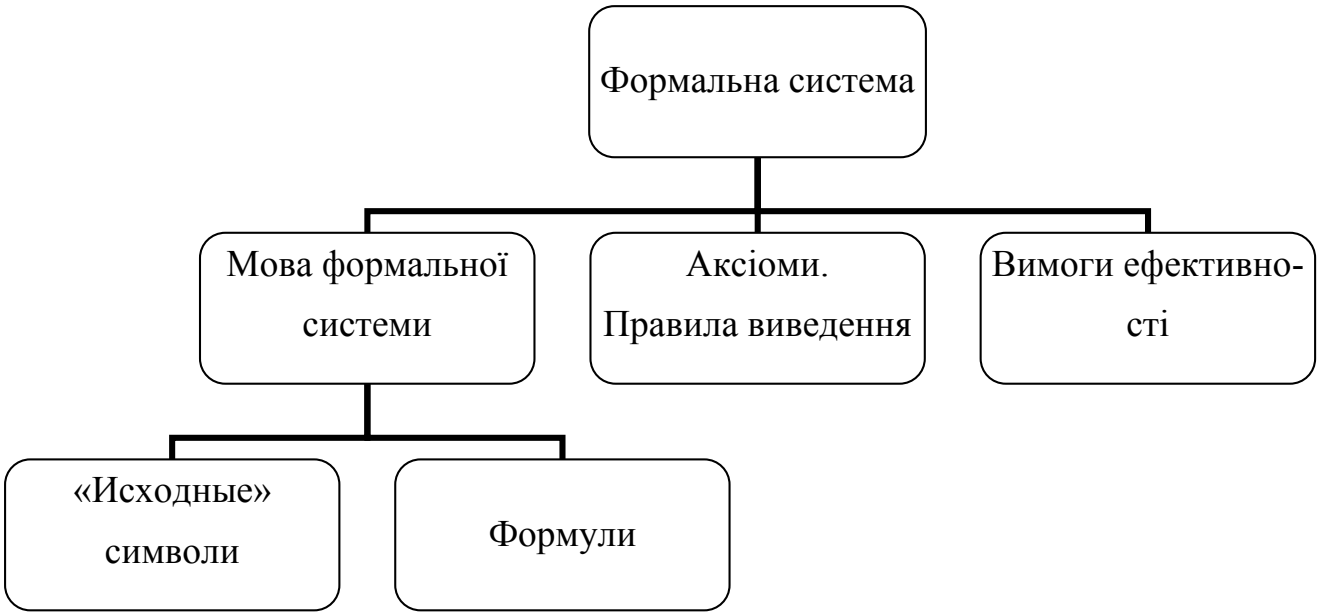


Рис. 1.2. Складові формальної системи

На додачу до символів елементарних співвідношень існують операції логічної композиції, які дозволяють утворювати складені умови:

– `(and (вираз1) … (виразn))`;

– `(or (вираз1) … (виразn))`;

– `(not (вираз))`.

Таблиця 1.5

Порівняння алфавітів λ-числення та ФМП Scheme

λ-числення	**ФМП Scheme**
константи та змінні (позначаються малими буквами латинського алфавіту, можливе використання індексів; напри-	константи та змінні (можуть складатися з великих та малих букв, цифр та спеціальних символів (за виключенням дужок), не повинні складатися лише з цифр та містити пробіли; наприклад, `c1`, `c2`, `x`, `квадрат`, `Д`…)

λ-числення	**ФМП Scheme**
клад, c_1, c_2, x, y…)	
вирази або формули (або, інакше, – терми) (позначаються великими літерами латинського алфавіту, можливе використання індексів; наприклад, M_1, N, ...)	елементарні функції для виконання основних арифметичних операцій: +, –, *, / (функції «+» і «*» можуть містити декілька параметрів (в тому числі і більше 2); якщо функція «*» задана без параметрів, вона повертає 1, функція «+» без параметрів повертає 0; при використанні функції «–» всі параметри, окрім першого, віднімаються від першого; ділення схоже на віднімання)
спеціальні символи («(», «)», «.»)	спеціальні символи: «(», «)», «'», «.» та символи елементарних співвідношень «=», «>», «<», «>=», «<=».

Так, наприклад, умова належності числа x відрізку $[a, b]$ ($x \in [a, b]$) виглядатиме так:

```
(and (>= x a)
     (<= x b)).
```

Таблиця 1.6

Порівняння правил утворення формул λ-числення та ФМП Scheme

λ-числення	**ФМП Scheme**
операція абстракції – визначення функції	

($\lambda x_1.\lambda x_2.\ldots$*тіло*),
де x_1, x_2, … x_n – формальні параметри
	1. `(lambda (формальні параметри) тіло).`
	2. Якщо необхідно назвати функцію, яку описує λ-вираз, треба використати спеціальну форму `define`, що пов'язує між собою ім'я функції та її λ-вираз:
	`(define function`
	`  (lambda (формальні параметри) тіло)).`
	В цьому випадку λ-вираз скорочується: слово `lambda` та дужки початку й кінця списку видаляють. В такій формі функція `define` отримує в якості параметрів дві частини. Перша – прототип виклику функції, який склада-

λ-числення	**ФМП Scheme**

	ється з імені функції та списку її формальних параметрів, а друга – тіло функції:
	`(define (ім'я формальні_параметри)`
	`  тіло)`

Наприклад, функція інкременту (збільшення аргументу на 1):	
(λ*x*. *x*+1)	1. `(lambda (x) (+ x 1))` 2. (`define (inc x)` `(+ x 1))`
операція аплікації – застосування функції до аргументу	
(*MN*)	`(M N)`
застосування функції інкременту до аргументу *x* = 5	
(λ*x*. *x*+1) (5)	1. `((lambda (x) (+ x 1)) 5)` 2. `(inc 5)`

Таблиця 1.7

Порівняння аксіом λ-числення та ФМП Scheme

<table>
<tr><th>λ-числення</th><th colspan="2">ФМП Scheme</th></tr>
<tr><td rowspan="5">можливість підстановки терма y замість всіх входжень змінної x в λ-вираз: a λx.a = λy.[y/x]a</td><td colspan="2">результат застосування функції не залежить від імен формальних параметрів: наприклад, результат функцій <code>подвоїти1</code> і <code>подвоїти2</code>:</td></tr>
<tr><td><code>(define (подвоїти1 x)
  (* 2 x))</code></td><td><code>(define (подвоїти2 y)
  (* 2 y))</code></td></tr>
<tr><td colspan="2">буде однаковим, якщо їх застосувати до одного й того ж числа:</td></tr>
<tr><td><code>(подвоїти1 4)</code>
результат 4</td><td><code>(подвоїти2 4)</code>
результат 4</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>можливість редукції (спрощення вигляду) λ-виразу в лівій частині</td><td colspan="2">розглянемо, наприклад, таку задачу: створити програму, яка повертає суму квадратів двох чисел. Розв'язок можна отримати, створивши функцію <code>сума_квадратів1</code>:</td></tr>
</table>

<table>
<tr><th>λ-числення</th><th colspan="2">ФМП Scheme</th></tr>
<tr><td rowspan="4">шляхом підстановки b замість всіх входжень змінної x в λ-вираз a: (λx.a)b = [b/x]a</td><td colspan="2">(define (сума_квадратів1 x y)
(+ (* x x)
(* y y)))
А можна створити функцію, яка повертає квадрат аргументу, і, використовуючи її, створити функцію сума_квадратів2:
(define (квадрат x)
(* x x))
(define (сума_квадратів2 x y)
(+ (квадрат x)
(квадрат y)))
У другому випадку ми спростили вигляд функції сума_квадратів шляхом підстановки функції квадрат.
Перевірка:</td></tr>
<tr><td>(сума_квадратів1 2 3)
результат 13</td><td>(сума_квадратів2 2 3)
результат 13</td></tr>
</table>

Таблиця 1.8

Порівняння правил виведення формул λ-числення та ФМП Scheme

<table>
<tr><th>λ-числення</th><th colspan="2">ФМП Scheme</th></tr>
<tr><td rowspan="4">якщо a=b, то ca=cb</td><td colspan="2">результат функції, застосованої до однакових аргументів, буде однаковий. Наприклад:</td></tr>
<tr><td>(define a 6)</td><td>(define b 6)</td></tr>
<tr><td colspan="2">В результаті таких дій було створено дві функції-константи a і b з однаковим значенням 6.
Застосуємо функцію інкременту до аргументів a і b</td></tr>
<tr><td>(inc a) → результат 7</td><td>(inc b) → результат 7</td></tr>
<tr><td rowspan="2">якщо a=b, то ac=bc</td><td colspan="2">результати однакових функцій, застосованих до одного й того ж аргументу, будуть однакові:</td></tr>
<tr><td>(define (a x) (+ x 1))</td><td>(define (b x) (+ x 1))</td></tr>
</table>

<table>
<tr><th>λ-числення</th><th colspan="2">ФМП Scheme</th></tr>
<tr><td rowspan="2"></td><td colspan="2">В результаті: створено дві однакові функції <code>a</code> і <code>b</code>. Тоді:</td></tr>
<tr><td><code>(a 3)</code> → результат 4;</td><td><code>(b 3)</code> → результат 4.</td></tr>
<tr><td rowspan="2">якщо $a=b$, то $\lambda x.a=\lambda x.b$</td><td colspan="2">однакові функції мають однакові тіла</td></tr>
<tr><td><pre>(define a
  (lambda (x) (+ x 1)))</pre></td><td><pre>(define b
  (lambda (x) (+ x 1)))</pre></td></tr>
<tr><td colspan="3">$a=a$ (рефлексивність)</td></tr>
<tr><td colspan="3">якщо $a=b$, то $b=a$ (симетричність)</td></tr>
<tr><td colspan="3">якщо $a=b$ і $b=c$, то $a=c$ (транзитивність)</td></tr>
</table>

Наведені у табл. 1.5–1.8 порівняння складових λ-числення та ФМП Scheme надають можливість зробити висновок про доцільність паралельного навчання математичної логіки та функціонального програмування. Так, засобами ФМП Scheme легко проілюструвати правила утворення формул (λ-виразів), пов’язати λ-вирази та визначення функцій, показати виконання аксіом λ-числення, правила виведення термів та інше засобами мови програмування.

Таким чином, ФП до побудови мов програмування надає можливість реалізувати в них основні положення λ-числення, а застосування ФМП надає можливість описати обчислювальний процес у вигляді абстракцій функцій та їх аплікацій.

1.3.3 Реалізація функціонального підходу в мовах програмування. Реалізацією ФП в програмуванні є ФМП.

Одним із центральних понять математики є поняття функції. Математичні функції виражають зв’язок між параметрами (входом) і результатом (виходом) деякого процесу. Оскільки обчислення – це теж процес, що має вхід і вихід, функція – цілком адекватний засіб опису обчислень. Саме цей простий принцип покладений в основу ФМП. Функціональна програма являє собою визначення функцій. Функції визначаються через інші функції або ж рекурсивно – через себе. У процесі виконання програми, функції одержують параметри, обчислюють і повертають результат, у разі потреби обчислюючи значення інших функцій. Програмуючи функціональною мовою, програміст не повинен описувати порядок обчис-

лень. Йому необхідно просто описати бажаний результат у вигляді системи функцій.

Основні властивості функціональних мов:

Відкладені обчислення. В імперативних мовах, наприклад, мові Паскаль при застосуванні функції до аргументу, спочатку обчислюється аргумент, а вже потім він передається до функції. В цьому випадку кажуть, що аргумент передається за значенням, маючи на увазі, що лише його значення передається до функції. Таке правило обчислень називається «виклик за значенням». Переваги виклику за значенням полягають в тому, що його реалізація проста: спочатку обчислюються значення аргументу, а потім обчислюється значення функції. Але недоліком є надмірні обчислення: існують випадки, коли значення аргументу зовсім не потрібне для обчислення значення функції. Альтернативою виклику за значенням є виклик за необхідності, в якому всі аргументи передаються до функції в необчисленому вигляді і обчислюються лише тоді, коли в них виникає необхідність всередині тіла функції.

Стислість. Програми на функціональних мовах зазвичай набагато коротші, ніж ті ж самі програми на імперативних мовах.

Модульність. Механізм модульності надає можливість розділяти програми на декілька порівняно незалежних частин (модулів) з чітко визначеними зв’язками між ними. Тим самим полегшується процес проектування і наступної підтримки великих програмних систем.

Використання функцій як значень. В функціональних мовах функції можуть бути передані іншим функціям як аргумент або повернуті як результат. Функції, що приймають функціональні аргументи, називаються функціями вищих порядків або функціоналами.

Чистота (відсутність побічних ефектів). В імперативних мовах функція в процесі свого виконання може читати і модифікувати значення глобальних змінних і здійснювати операції введення/виведення. Тому, якщо викликати ту саму функцію двічі з тим самим аргументом, може трапитися так, що результатом будуть два різних значення. Така функція називається функцією з побічними ефек-

тами.

У функціональному програмуванні оператор присвоювання відсутній, об’єкти не можна змінювати і знищувати, можна тільки створювати нові шляхом декомпозиції і синтезу існуючих. Завдяки цьому в функціональних мовах усі функції вільні від побічних ефектів.

Стосовно мов функціонального програмування існують декілька поширених хибних уявлень, які здебільшого наводять у списку їх недоліків. Наприклад:

– мови функціонального програмування здебільшого академічні – на противагу цьому твердженню наведемо приклади застосування мов ФП у відомих програмних системах: AutoCAD (AutoLisp), GIMP (Scheme), Mldonkey (OCaml), Jabber (Erlang), Yahoo! Store (Common Lisp);

– функціональне програмування непосильне для початківців – в якості спростування можна розглянути мови програмування, створені спеціально для навчання програмування початківців (Logo, Scheme);

– сфера використання функціонального програмування обмежена розв’язуванням задач штучного інтелекту – заперечення цього твердження надамо в другому розділі при розгляді середовищ програмування.

Теорія, покладена в основу функціонального підходу, народилася в 20-х – 30-х роках XX століття. У витоків математичних основ функціонального програмування стояли вітчизняний математик М. І. Шенфінкель та англієць Хаскелл Каррі, що розробили теорію комбінаторів, а також Алонзо Чьорч (США), творець λ-числення [2].

На початку 1950-х Джон Маккарті розробив мову Lisp, що стала першою функціональною мовою програмування і протягом багатьох років залишалася єдиною [13]. Lisp було створено для підтримки мовних засобів обробки списків, потреба в яких виникла з появою робіт в галузі штучного інтелекту.

У даний час найбільш відомими функціональними мовами програмування є такі: Lisp, CommonLisp, Scheme, Haskell, ML, Hope, Miranda, R.

У вищій школі в навчанні інформатичних дисциплін все більшого застосування набувають *мобільні інтерпретовані мови загального призначення*. Яскра-

вим прикладом такої мови є функціональна мова програмування Scheme, застосування якої надало можливість створити вступний курс з програмування незалежний від операційної системи, що використовується.

Scheme – невеликий і елегантний діалект мови Lisp – інтерпретована функціональна мова програмування високого рівня. Перший варіант мови Scheme був створений Д. Д. Сассманом та Г. Стілом в Массачусетському технологічному інституті у 1975 р. як «простий та конкретний експериментальний інструмент для досліджень в галузі семантики та стилю програмування» [23]. Найбільш інтенсивна розробка ядра мови відбувалась у період з 1975 по 1980 рр.

Серед переваг Scheme можна назвати:

– відкладені обчислення (альтернативою виклику за значенням є виклик за необхідності);

– стислість;

– модульність;

– використання функцій як значень;

– чистота (відсутність побічних ефектів);

– невеликий розмір ядра мови (менше 20 ключових слів);

– мобільність програм (внаслідок інтерпретованої природи мови);

– можливість використання в діалоговому режимі (корисне для експериментування та розв’язування простих задач);

– зручність для розв’язування математичних задач (можливість опрацьовувати числові типи даних, подані як у вигляді десяткового та десяткового періодичного дробу, так і звичайного дробу; також Scheme містить засоби роботи з комплексними числами, цілими числами довільної довжини);

– простота синтаксису (завдяки цьому програми на Scheme зазвичай є коротшими, їх легше писати, читати та розуміти; виникає можливість зосередитись на засвоєнні загальних принципів програмування, а не на вивченні синтаксису):

Програма мовою C	*Програма мовою Scheme*
`int factorial(int x) {`	`(define (factorial x)`
`  if (x == 0) {`	`  (if (= x 0)`

```
    return 1;
  } else {
    return x * factorial(x-1);
  }
}
```

```
    1
    (* x (factorial(- x 1)))))
```

– підтримка багатьох парадигм (функціональної, об'єктно-орієнтованої та імперативної).

Інтерпретатор Scheme та стандартні бібліотеки доступні на всіх основних платформах, тобто сам інтерпретатор є кросплатформенним.

Вказані переваги стали причиною застосування мови Scheme як у вступних курсах інформатики, так і при розгляді загальних принципів програмування у провідних вищих навчальних закладах світу (за матеріалами журналу Times [26]): Гарвардський університет, США; Єльський університет, США; Чиказький університет, США; Массачусетський технологічний університет, США; Каліфорнійський технологічний університет, США; Токійський технологічний інститут, Японія; Гонконгзький університет науки та технологій, Гонконг; університет Торонто, Канада; Національний університет Сінгапуру, Сінгапур та інших (Додаток А). На рівні середніх навчальних закладів Scheme також активно впроваджується, витісняючи, таким чином, традиційні мови програмування Basic, Pascal, або C [20].

Саме тому при розробці курсу «Вступ до програмування» для студентів молодших курсів напряму підготовки «Інформатика*» вищих педагогічних навчальних закладів в якості мови програмування було обрано Scheme.

1.4 Психолого-педагогічні особливості формування у студентів молодших курсів компетентностей з програмування на основі функціонального підходу

Одним із основних принципів дидактики вищої школи є врахування вікових та індивідуальних особливостей студентів.

Віковий період 17-20 років (а саме такий вік має більшість студентів першого та другого курсів ВНЗ) більшість психологів називають юністю (або пізньою юністю) [36, 107], [110], [140, 344], [305, 264], ранньою дорослістю чи молодістю

[305, 287] або ж студентським віком [215, 238].

Розвиток в юності в меншій мірі пов'язаний з фізичним ростом, а більше – з швидким когнітивним вдосконаленням, відбувається посилений розвиток психічних функцій, особливо пам'яті та мислення, довгострокової пам'яті [215, 242]. Пам'ять людини в юності характеризується високим рівнем оволодіння складною системою узагальнених умінь розуміння та запам'ятовування, відбувається подальший розвиток абстрактно-логічної пам'яті, оволодіння прийомами, що допомагають запам'ятовувати складні системи фактів, понять, законів.

Зростання самостійності в навчальному процесі студентів збільшує роль довільного запам'ятовування абстрактно-логічної пам'яті. Відставання в цей період мислення від розвитку пам'яті знижує ефективність запам'ятовування і відтворення, бо матеріал недостатньо опрацьовується. Зменшити цю розбіжність можна шляхом осмислення та структурування матеріалу. Пам'ять потрібна для засвоєння нових знань, а їх набуття сприяє вдосконаленню пам'яті [110, 202].

Процес поліпшення пам'яті продовжується завдяки навчанню. Особливе значення для цього має уміння працювати з книгою, складати плани й реферувати матеріали, вести дослідження. Чим стійкіші мотиви досягнення мети супроводжують діяльність, тим продуктивніша пам'ять. Лише усвідомлена діяльність сприяє розвиткові пам'яті [110, 203].

Найпотужнішим чинником для розвитку уваги є пізнавальний інтерес, інтерес не лише до знань, а й до самого процесу учіння. Саме цей інтерес здійснює вирішальний вплив на всю психічну діяльність. Він сприяє переростанню довільної уваги у післядовільну: воля відходить на задній план, а її функції замінює інтерес [110, 258]. Оскільки шлях до післядовільної уваги лежить через захопленість своєю справою важливо підбирати «змістові» завдання.

Наприклад, при вивченні теми «Циклічні вирази» можна запропонувати студентам таке завдання: кредитний відділ деякого банку хоче обрати систему нумерації для нової кредитної картки таким чином, щоб можна було легко відрізнити справжній номер картки від підробленого. Дизайнерський відділ запропонував наступну перевірку: кредитна картка дійсна, якщо її номер без остачі ділиться

на 13, але не ділиться на 2, 3, 5, 7 та 11. Наприклад, номер 1000051 є дійсним, а 1000571 – ні (оскільки 1000571 = 11 * 90961). Визначте та протестуйте функцію `номер-дійсний?`, яка перевіряє, чи є задане ціле число допустимим для номеру кредитної картки відповідно до цієї умови.

Наприклад:

```
> (номер-дійсний? 1000051)
#t
> (номер-дійсний? 1000751)
#f
```

Маркетингове відділення цього ж банку занепокоєне: чи вистачить номерів для карток, адже, за їх розрахунками, близько 180000 клієнтів захочуть отримати цю картку, а номер картки обмежений семи цифрами. Створіть та протестуйте функцію `(число-дійсних-карток початок кінець)`, яка знаходить кількість дійсних номерів для кредитних карт у вказаному діапазоні (початок кінець).

Після перевірки функції `число-дійсних-карток` для невеликих діапазонів, з'ясуйте, чи вистачить-таки номерів для кредитних карт?

Саме для юнацького віку характерний інтенсивний розвиток вольових якостей (наполегливості, дисциплінованості, самостійності, цілеспрямованості, ініціативності, організованості) та відносна завершеність їх формування [110, 389]. Вольові якості стають компонентами й рисами характеру особистості.

В юності вдосконалюється володіння складними інтелектуальними операціями аналізу й синтезу, теоретичного узагальнення й абстрагування, аргументування й доведення, порівняння й класифікації та ін. Характерним стає встановлення причинно-наслідкових зв'язків, вдосконалюються такі якості мислення як широта, глибина, ясність думки, послідовність, самостійність, критичність, звільнення від шаблонних способів розв'язування завдань, здатність змінювати способи відповідно до конкретної ситуації, швидко зосереджувати сили на нових завданнях і активно, енергійно працювати над їх розв'язанням, відбувається перехід від формальної до діалектичної логіки [110, 358]. Виникає тенденція до узагальненого розуміння світу, до цілісної і абсолютної оцінки тих чи інших явищ дійсності [305, 271].

Юність – перший період дорослого, самостійного життя, час вибору життєвого шляху. Відповідальність за свою долю, за все наступне життя визначає специфіку цього вікового періоду. В психологічному плані юність розв'язує задачі кінцевого, дійсного самовизначення та інтеграції в суспільство дорослих людей, відбувається формування науково-теоретичних, філософських, моральних та естетичних ціннісних орієнтацій, в яких виявляється сама сутність людини. Формується світогляд як система узагальнених уявлень про світ в цілому, про оточуючу дійсність та інших людей і самого себе, готовність керуватись ним в дійсності. Формується усвідомлене відношення до життя.

Студенти першого курсу переживають кризу 17 років (вона виникає на рубежі звичного шкільного та нового дорослого життя) – різка зміна образу життя, включення в нові види діяльності, спілкування з новими людьми викликають значну напругу. Тому, серед психолого-педагогічних особливостей формування у студентів молодших курсів компетентностей з програмування необхідно враховувати особливості початкового етапу навчання, які пов'язані з *адаптацією* студентів у вузі.

У педагогічному словнику [266, 8] адаптація до навчання у вузі визначається як «пристосування особистості або групи до змінених зовнішніх умов» та «перебудова пізнавальної, мотиваційної та емоційно-вольової сфер» учня при переході до систематично організованого вузівського навчання. На початковому етапі навчання адаптація може бути визначена, як процес подолання труднощів при включенні особистості першокурсника в навчальний процес у вузі та входження в новий колектив [117, 89].

Період адаптації студентів-першокурсників достатньо складний, багатогранний і потребує допомоги педагогів, спрямованої на з'ясування студентами цілей та змісту навчальної діяльності у вузі, оволодіння новими для молодих людей методами пізнавальної діяльності та формування їх взаємовідносин у студентській групі та з викладачами.

Розрізняють такі форми адаптації студентів:

– професійна адаптація – пристосування до структури вищої школи, загаль-

ного змісту й окремих компонентів навчального процесу, особливостей обраної професії;

– дидактична адаптація, що забезпечує наступність у системі «школа – ВНЗ», поступове входження у сферу навчання у ВНЗ;

– соціально-психологічна адаптація, яка виражається у формуванні стосунків із однокурсниками та мірі вдоволення цими відносинами [63, 56].

Професійна адаптація обумовлена іншими, відмінними від школи, формами та методами навчальної діяльності. Відмінність навчання у ВНЗ полягає не тільки у спеціальній спрямованості знань (професійній спеціалізації), а й більшими складністю та обсягом навчального матеріалу, зростанням кількості годин самостійної роботи. Як наслідок – студентам необхідні вміння обирати цілі навчання та встановлювати їхню пріоритетність; планувати та організовувати свою навчальну діяльність; здійснювати контроль за нею; аналізувати результати власної роботи, розв’язувати проблеми, що виникають у процесі навчання та організовувати пошук і обробку необхідних навчальних матеріалів з використанням різноманітних засобів сучасних ІКТ.

Тому важливо з перших занять ознайомити студентів з організаційними питаннями, пов’язаними з вивченням даної дисципліни та надати їм наступні відомості: кількість лекційних та лабораторних годин на тиждень, завдання курсу та його зв’язок з іншими дисциплінами, форми контролю і критерії оцінки, поверховий огляд структури та змісту дисципліни, форми організації та оцінювання самостійної роботи, навчально-методичні матеріали курсу. У зв’язку з цим обов’язковим питанням першої (вступної) лекції є огляд основних положень робочої програми курсу (для подальшої роботи ці відомості також розміщені у вступній частині розробленого НМК «Вступ до програмування»).

Другою формою адаптації у вузі є дидактична адаптація, яка пов’язана з низьким рівнем знань, отриманих у школі, та з великим обсягом навчального матеріалу. Тому для формування компетентностей з програмування у студентів молодших курсів необхідним є вибір мови програмування з малим синтаксичним ядром та можливістю створення особистісно значущих програм уже на початковому

етапі навчання, без детального ознайомлення з синтаксисом мови. Саме такими є мови ФП. Тому для розробленого курсу «Вступ до програмування» було обрано ФМП Scheme.

Неабиякою проблемою є соціально-психологічна адаптація першокурсників, проявами якої можуть бути: розвиток депресій, наявність конфліктів із однокурсниками та викладачами, погіршення пам'яті, неможливість зосередитися на навчанні, виникнення підвищеної збудженості та стомленості.

Створенню позитивної соціально-психологічної атмосфери під час вивчення курсу «Вступ до програмування» сприяє використання засобів організації спільної роботи, використання групових та проектних форм організації навчання.

У результаті досліджень психолого-педагогічних факторів, які впливають на ефективність формування у студентів молодших курсів компетентностей з програмування, можна зазначити, що до основних педагогічних факторів відносяться: зміст навчального матеріалу; різноманітні методи, форми та засоби навчально-виховної роботи; компетентності викладача; стан матеріально-технічної бази ВНЗ; готовність студентів до навчальної діяльності, їх психологічний стан та наявність *мотивації* [117, 95].

Мотивація – сукупність причин, що спонукають людину до активної діяльності та надають їй осмисленість. Не знаючи мотивів, не можна зрозуміти, чому людина прагне до однієї, а не до іншої цілі, не можна зрозуміти справжню суть її дій.

У сучасних умовах навчання у ВНЗ проблема розвитку мотивів і потреб людини є однією з найбільш актуальних. Це визначається як вітчизняними, так і закордонними дослідженнями [125, 15].

Мотивація навчання – це стимулювання студентів до навчально-пізнавальної діяльності, яка спрямована на досягнення конкретних цілей та розв'язування завдань. Основою мотивації учіння є різноманітні потреби та інтереси суб'єктів навчання, врахування та задоволення яких суттєво покращує не лише якісні показники у навчанні та розвитку, а й полегшує процес управління всіма компонентами навчально-пізнавальної діяльності суб'єктів навчання [125,

16].

Існують різні класифікації мотивів. За усвідомленістю розрізняють усвідомлювані й неусвідомлювані мотиви. Неусвідомлювані мотиви – установки і потяги, усвідомлювані – інтереси, переконання, прагнення [110, 304].

Інтерес – це емоційний вияв пізнавальних потреб особистості. Суб'єктивно інтереси розкриваються на позитивному емоційному тлі, в бажанні глибше пізнати об'єкт, зрозуміти його. Інтереси є спонукальним механізмом пізнання, змушують особистість шукати шляхів, засобів задоволення того чи іншого бажання. Безпосередні інтереси зумовлює емоційна привабливість об'єкта (інтерес до розв'язання кросвордів, завдання-ігри). Саме тому в розробленому НМК «Вступ до програмування» створено різноманітні ігри: «Мільйонер», «Кросворд», «Шибениця», «Змії та сходи», «Криптекст» (див. п. 2.2.2).

Опосередкований інтерес виникає щодо результату діяльності, хоча сам процес не завжди цікавить суб'єкта. Виникненню інтенсивного і стійкого інтересу до навчання сприяє залучення студентів до пошукової та дослідницької діяльності. З метою формування опосередкованого інтересу в розробленому курсі «Вступ до програмування» при вивченні третього та четвертого модулів використовується проектна форма організації роботи.

Розглянемо деякі умови, що викликають інтерес студентів до навчання [125, 20]:

1. Спосіб подання навчального матеріалу. Здебільшого навчальний предмет має певну структуру і пропонується студентам у вигляді окремих розділів або тем. У цьому випадку необхідно показувати взаємозв'язок всіх розділів, важливо надавати не готові способи роботи з навчальним матеріалом, а сприяти розумінню студентами суті досліджуваних явищ. Так, наприклад, на початку вивчення теми «Циклічні вирази» можливо запропонувати студентам розв'язати задачу знаходження факторіалу числа n, що визначається рівнянням:

$$n! = n \cdot (n - 1) \cdot (n - 2) \cdot \ldots \cdot 3 \cdot 2 \cdot 1.$$

Це рівняння можна переписати в такому вигляді:

$$n! = n \cdot (n - 1) \cdot (n - 2) \cdot \ldots \cdot 3 \cdot 2 \cdot 1 = n \cdot (n - 1)!$$

Оскільки дана тема є наступною після умовних виразів, студенти мають можливість інтуїтивно прийти до вирішення цієї задачі. Враховуючи, що 1! = 1, отримуємо:

```
> (define (факторіал n)
    (if (= n 1)
        1
        (* n (факторіал (- n 1)))))
> (факторіал 3)
6
```

Обчислення факторіалу можна записати і інакше: спочатку множимо 1 на 2, далі результат множимо на 3, потім на 4 і так доти, доки не досягнемо n. Для опису цього обчислення необхідно записати правила, за якими змінюються `лічильник` (який набуватиме значень від 1 до n) та `результат`:

```
результат = результат · лічильник
лічильник = лічильник + 1
```

Умова закінчення цього процесу: `лічильник > n`.

```
> (define (факторіал n)
    (define (цикл результат лічильник)
        (if (> лічильник n)
            результат
            (цикл (* результат лічильник)
                  (+ лічильник 1))))
    (цикл 1 1))
> (факторіал 4)
24
```

Далі можна ввести поняття самозастосовних функцій. І вже потім ознайомити із спеціальними конструкціями для організації циклічних обчислень.

2. Організація роботи над вивченням предмету малими групами. При об’єднанні студентів у групи, окрім побажань студентів, необхідно враховувати, що, при включенні студентів з нейтральною мотивацією до вивчення предмету до групи студентів, які не люблять даний предмет, після спільної роботи перші суттєво підвищать свій інтерес до вивчення даного предмету. Даний принцип діє і навпаки: при об’єднанні студентів з нейтральним ставленням до предмету в одну

групу зі студентами, які люблять предмет, ставлення у перших не зміниться. Найбільш повно дане твердження реалізується у практику в третьому модулі курсу «Вступ до програмування» при розробці проектів.

3. Співвідношення між мотивом і метою. Метою вивчення даного курсу у студентів має бути не отримання бажаної оцінки (або ж страх перед учителем, батьками чи ін.), а формування компетентностей з даного предмету: мета, поставлена викладачем, повинна стати метою студента. Тому важливо показувати взаємозв'язок даної дисципліни із тими, що будуть вивчатись у подальшому. Також важливо, щоб мета була «досяжною». Для цього кожна лабораторна робота має містити пункт «мета», а у висновках до лабораторної роботи студенти мають описати, що вони досягли в результаті її виконання, тобто необхідно сприяти розумінню студентами власних досягнень.

4. Проблемність навчання. Для формування стійкого інтересу до навчального предмету варто використовувати проблемні методи навчання, створювати проблемні ситуації та завдання.

5. Зміст навчання. Для формування зацікавленого ставлення необхідно показувати взаємозв'язок теорії й практики, підбирати змістові завдання, сприяти усвідомленню студентами навчального матеріалу.

На формування змістових мотивів під час навчання інформатики, перш за все, впливають структура й зміст даної дисципліни, які повинні повною мірою відповідати сучасному стану та тенденціям розвитку інформатики як науки. При цьому зміст навчання повинен відповідати наступним вимогам [134, 184]:

– у студентів повинно виникати відчуття невдоволення тими знаннями, уміннями та навичками, якими вони володіють. Тому вивчення нової теми необхідно починати з постановки завдання, для розв'язання якого у студентів недостатньо знань чи вмінь (наприклад: тема «Умовні вирази», завдання: визначити функцію, яка повертає модуль введеного числа);

– студенти повинні чітко бачити зміст нових понять: новий матеріал необхідно подавати чітко та логічно;

– студенти повинні бути готовими до встановлення зв'язку між новими та

попередніми поняттями (тобто, для кожної теми по можливості показати зв'язок з попередніми: тема «Циклічні вирази» логічно пов'язується з темою «Умовні вирази», а якщо її вивчення розпочинати з рекурсії, то студенти мають змогу без будь-яких нових відомостей отримати бажаний результат – циклічну функцію);

– нові представлення для студентів повинні бути кориснішими та перспективнішими від старих (наприклад, під час вивчення теми «Графічний інтерфейс» студенти отримують завдання створити графічний інтерфейс до ЕС, розробленої попередньо).

Крім того, під час навчання дисципліни «Вступ до програмування» завдяки використанню КОЗН (див. п. 2.2) підвищується мотивація студентів за рахунок [134, 185]:

– можливості вивчати та закріплювати навчальний матеріал за допомогою засобів, які надають можливість працювати у власному темпі навчальної діяльності, самостійно планувати хід навчання з урахуванням пропозицій викладача та власного досвіду, рівня самостійної пізнавальної діяльності, загальних здібностей (цьому сприяє як самостійна робота в НМК із опрацювання теоретичного матеріалу, так і виконання лабораторних та індивідуальних робіт, робота з глосарієм, wiki та ін.);

– можливості проведення об'єктивного та оперативного контролю знань, умінь та навичок студентів (використовуючи розроблені в НМК тестові завдання);

– можливості проведення дослідницької роботи в комп'ютерних лабораторіях;

– можливості організації спільної роботи.

Зацікавленому відношенню студентів до навчання та розвитку мотивації також сприяє використання методу доцільно дібраних задач, в яких відображається практичний зміст вивчення предмета.

Залежно від того, що є в основі мотивації – спонука чи потреба пізнання, у навчальній діяльності виокремлюють такі мотиви:

– безпосередньо спонукальні: виникають у студентів за рахунок компетентності викладача, що формує інтерес до даного предмету. І хоча в цьому випадку

виникає швидше зацікавленість, а не мотивація пізнавальної діяльності, проте для майбутнього педагога така зацікавленість може перерости у мотивацію, оскільки він бачить ніби еталон для своєї майбутньої діяльності.

– перспективно спонукальні: пізнавальна діяльність є тільки засобом досягнення мети, яка знаходиться власне поза пізнавальною діяльністю. Наприклад, вимогливість викладачів, вимогливість до себе, значущість предмету для майбутньої діяльності, прагнення оправдати довіру та надії батьків, меркантильні інтереси (бажання скласти сесію, отримати стипендію, бути краще за всіх).

– пізнавально-спонукальні мотиви безкорисливого пошуку знань, істини. Інтерес до навчання виникає у зв'язку з проблемою і розвивається у процесі розумової праці, пов'язаної з пошуком та знаходженням розв'язання проблемної задачі або групи задач, у зв'язку з чим студент відчуває ні з чим незрівнянне задоволення, і на цій основі виникає внутрішня зацікавленість. З виникненням пізнавально-спонукальних мотивів відбувається перебудова сприйняття, пам'яті, мислення, переорієнтація інтересів, активізація здібностей студента, створюються передумови для успішної навчальної діяльності, до якої у нього є інтерес [125, 30].

За спрямуванням та виникненням вирізняють внутрішні і зовнішні мотиви. Внутрішні мотиви визначаються власними потребами, інтересами та переконаннями студента, а зовнішні – вимогами, які ставлять перед студентами суспільство, викладачі, батьки, колектив, ситуація навчання тощо. Спочатку мотиви навчальної діяльності формуються під впливом зовнішніх по відношенню до неї факторів, які пізніше повинні замінюватися внутрішніми. Сформованість внутрішніх мотивів виступає необхідною умовою активної навчально-пізнавальної діяльності студентів. Оптимальною є та навчальна мотивація, в якій переважають позитивні установки, стійкі і збалансовані соціальні, професійні, пізнавальні мотиви, а провідними є мотиви, орієнтовані на перспективу [277, 32].

Таким чином, серед психолого-педагогічних особливостей формування у студентів молодших курсів компетентностей з програмування на основі ФП слід відзначити:

1. Кризовий віковий період студентів: для адаптації необхідна допомога пе-

дагогів;

2. Швидкий розвиток усіх психічних функцій (уваги, мислення, пам'яті): необхідність формування у студентів умінь самостійної, творчої, дослідницької роботи;

3. Необхідність розробки всіх компонентів методичної системи навчання, що сприятимуть формуванню позитивної навчально-пізнавальної мотивації:

– мета навчання має усвідомлюватись студентами як їх внутрішня потреба;

– зміст навчання повинен сприяти виникненню стійкого інтересу, як до результату, так і до процесу навчання;

– методи активного навчання сприяють формуванню у студентів вмінь наукової-дослідницької роботи;

– групові форми організації навчання надають можливість педагогу сприяти як соціально-психологічній адаптації студентів, так і створенню позитивної мотивації;

– КОЗН: особливості мов ФП (простота синтаксису, можливість створення коротких особистісно значущих задач);

4. Неабияке значення мають *компетентності викладача*, як загальнопрофесійні, так і галузеві.

1.5 Функціональний підхід у формуванні мислительних операцій

Для формування компетентностей з програмування насамперед важливо сформувати у студента здатність «мислити». Здебільшого, в цьому контексті вимагається сформованість алгоритмічного, операційного чи об'єктного стилів мислення [67]. Найбільш детально мислительні операції розглянуті у роботах С. Л. Рубінштейна. Так, він зазначає, що «для розв'язання поставлених задач мислення йде шляхом різноманітних операцій, які складають різні взаємопов'язані та такі, що переходять один в одного, сторони мислительного процесу. Такими операціями є порівняння, аналіз, синтез, абстракція та узагальнення [248, 324].

Аналіз – це мисленне розчленування навчальної задачі і виявлення її складових елементів. Розв'язування будь-якої задачі розпочинається з її аналізу. Від-

новлення розділеної аналізом навчальної задачі, з викриттям більш або менш суттєвих зв'язків і відношень елементів є операцією синтезу. Аналіз розчленовує проблему; синтез по-новому об'єднує дані для її вирішення.

Можливість виявлення схожості чи розбіжності надає операція порівняння. Абстракція – це виокремлення, вичленування однієї якої-небудь сторони, властивості предмету, в якому-небудь співвідношенні суттєвої та відволікання від інших. Узагальнення – це виділення суттєвих спільних властивостей, у яких предмети або явища схожі між собою [248, 327].

Розглянемо особливості формування мислительних операцій з використанням ФМП на прикладі знаходження сум математичних послідовностей – традиційної задачі в курсі з навчання основ програмування.

У ФМП розв'язування будь-якої задачі полягає у написанні функцій, аргументами яких є інші функції або їх результати. Таким чином, для розв'язання задачі необхідно встановити, які функції нам необхідні, та які між ними взаємозв'язки (операція аналізу й абстракції – визначення необхідних функцій, синтезу й абстракції – встановлення взаємозв'язків між цими функціями).

У ФМП задачу знаходження сум математичних послідовностей розглядають, вивчаючи рекурсію. В цих випадках важливо зосередити увагу студентів на:

– виборі «граничних» (тривіальних) умов;

– визначенні рекурсивної вітки.

Так, спочатку студентам пропонується задача про знаходження суми цілих чисел від 0 до n:

$S(n)= 1 + 2 + \ldots + (n – 2) + (n – 1) + n.$

Якщо цю формулу переписати у вигляді

$S(n)= n + (n – 1) + (n – 2) + \ldots + 2 + 1$

і позначити, що $S(n – 1) = (n – 1) + (n – 2) + \ldots + 2 + 1$, то можна записати наступну формулу: $S(n)= n + S(n – 1)$.

Міркуючи аналогічно, отримуємо: $S(n – 1)= (n – 1) + S(n – 2)$ і т.д.

У найпростішому випадку (якщо $n = 0$) $S(0) = 0$.

Після цього студенти приходять до висновку, що сума чисел у випадку, ко-

ли n = 0, теж рівна 0, інакше результатом буде сума числа n та суми чисел від 0 до зменшеного на 1 n.

Таким чином, під час побудови моделі задачі формується мислительна операція аналізу, а з метою побудови алгоритму та програми, що реалізує дану модель задачі – операція синтезу. Абстракція формується і під час побудови алгоритму та програми, і під час використання вже визначених функцій для визначення інших функцій.

```
> (define (сума-цілих-чисел n)
    (if (< n 0)
        0
        (+ n (сума-цілих-чисел (- n 1)))))
```

Далі доцільно запропонувати студентам задачу знаходження суми цілих чисел від a до b (включно).

Розв'язування задачі знову розпочинається з її аналізу. Розглянемо випадки:

1) якщо a = b, то S(a, b) = a;

2) якщо a > b, то S(a, b) = 0;

3) якщо a < b, то розв'язування цієї задачі схоже на розв'язування попередньої: запишемо

S (a, b)= a + (a + 1) + … + (b – 1) + b.

Або ж в іншому вигляді:

S (a, b) = b + (b – 1) + … + (a + 1) + a =

= b + S(a, (b – 1)) = b + ((b – 1) + S(a, (b – 2)))

У результаті таких перетворень зрештою можна перейти до випадку 2, коли S(a, b) = 0. Таким чином, об'єднуючи ці випадки, отримуємо:

```
> (define (сума-цілих-чисел a b)
    (if (> a b)
        0
        (+ b
           (сума-цілих-чисел a (- b 1)))))
```

Для закріплення варто розв'язати ще одну подібну задачу – про суму квадратів цілих чисел від a до b (функція `квадрат` визначена попередньо):

```
> (define (квадрат x)
    (* x x))
```

Розв'язування задачі розпочинаємо з її порівняння з попередньою задачею.

Запишемо формулу для знаходження суми:

$S(a, b) = a^2 + (a + 1)^2 + \ldots (b - 1)^2 + b^2$

Відмінність цих задач полягає лише у тому, що сумувати потрібно не просто числа, а їх квадрати:

```
> (define (сума-квадратів a b)
    (if (> a b)
        0
        (+ (квадрат b)
           (сума-квадратів a (- b 1)))))
```

Порівнюючи між собою задачі про знаходження сум чисел від 0 до n, від a до b та квадратів чисел від a до b, стає зрозуміла загальна форма функції для знаходження сум послідовностей виду $\sum_{n=a}^{b} f(n) = f(a) + \ldots + f(b)$:

```
> (define (сума терм a b)
    (if (> a b)
        0
        (+ (терм b)
           (сума терм a (- b 1)))))
```

Так в процесі розв'язування подібних задач формується операція порівняння. Маючи функцію `сума` розв'язок задачі про суму квадратів цілих чисел від a до b виглядатиме так:

```
> (define (сума-квадратів a b)
    (сума квадрат a b))
```

Щоб показати, що визначена функція `сума` не є універсальною для обчислення сум, студентам пропонується задача про знаходження суми парних чисел від a до b. Оскільки в задачі необхідно сумувати кожне друге число функція `сума` для даного випадку не підходить.

Аналіз задачі. Можливі випадки:

1) якщо b – парне число, то задача зводиться до задачі про обчислення суми

чисел від a до b, відмінність полягає лише у тому, що b треба зменшувати на 2;

2) якщо число b – непарне, то зменшити його на 1 і перейти до пункту 1).

```
> (define (сума-парних-чисел a b)
    (if (odd? b)
        (сума-парних-чисел a (- b 1))
        (if (> a b)
            0
            (+ b
               (сума-парних-чисел a (- b 2))))))
```

Для того, щоб була можливість визначити функцію `сума-парних-чисел`, використовуючи функцію `сума`, необхідно, щоб вона містила ще один аргумент – функцію, яка визначатиме наступне значення аргументів a і b:

```
> (define (сума терм next a b)
    (if (> a b)
        0
        (+ (терм b)
           (сума терм next a (next b)))))
```

Визначимо функцію `dec2`, що зменшує аргумент на 2 та функцію ідентичності `identity`, що повертає свій аргумент:

```
> (define (dec2 x)
    (- x 2))
> (define (identity x)
    x)
```

Тепер можна визначити функцію `сума-парних-чисел`, використовуючи функцію `сума`:

```
> (define (сума-парних-чисел a b)
    (if (odd? b)
        (сума-парних-чисел a (- b 1))
        (сума identity dec2 a b)))
```

Маючи функцію для обчислення сум в загальному випадку, розв’язок задачі про суму квадратів цілих чисел від a до b виглядатиме так:

```
> (define (inc x)
    (+ x 1))
```

```
> (define (сума-квадратів a b)
    (сума квадрат inc a b))
```

Зобразимо ієрархію абстракцій у прикладах із визначенням функцій `сума-цілих-чисел`, `сума-квадратів` та `сума` (рис. 1.3).

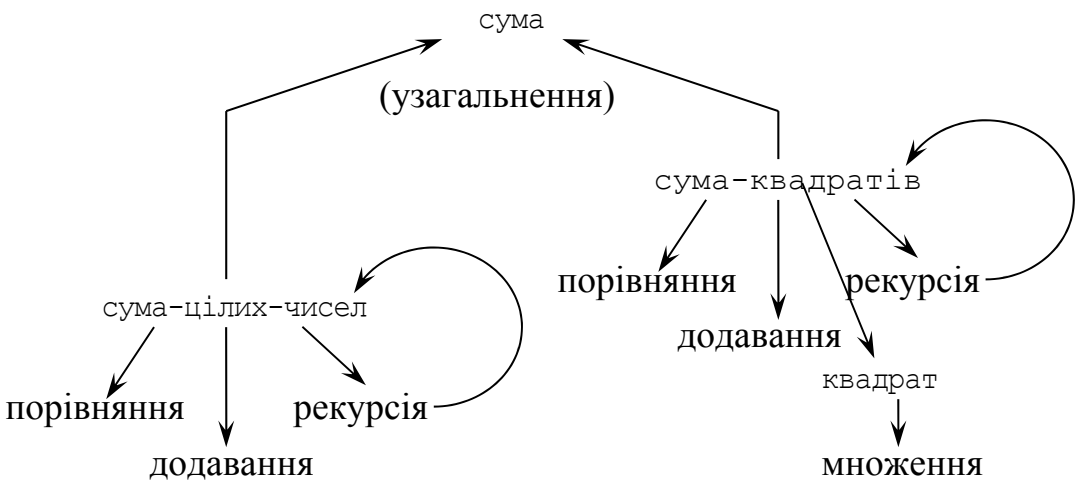


Рис. 1.3. Ієрархія абстракцій

Розглянемо також можливість формування мислительних операцій на прикладі знаходження коренів рівняння $f(x) = 0$ методом половинного ділення [106, 92].

Нехай задано неперервну функцію $f(x)$ і потрібно знайти x^* – корінь рівняння $f(x) = 0$, що знаходиться на проміжку $[a, b]$. На цьому проміжку функція $f(x)$ неперервна і на кінцях проміжку набуває різних значень, тобто $f(a) \cdot f(b) < 0$. Це означає, що на вказаному проміжку $[a, b]$ рівняння $f(x) = 0$ має принаймні один корінь x^*. Задача полягає в тому, щоб визначити цей корінь із наперед заданою точністю ε.

Метод половинного ділення полягає в тому, що проміжок $[a, b]$ поступово звужують, зменшуючи його щоразу удвічі (звідси і назва методу), доки не буде досягнуто необхідної точності.

Запишемо алгоритм розв’язування задачі (операція аналізу й абстракції):

1) точкою $x = \dfrac{a+b}{2}$ ділимо проміжок $[a, b]$ навпіл;

2) якщо $f(a) \cdot f(x) < 0$, то корінь знаходиться на проміжку $[a, x]$, інакше – на проміжку $[x, b]$;

3) дії 1-2 необхідно продовжувати доти, доки не виконається умова $\Delta < \varepsilon$ ($\Delta = \frac{b-a}{2}$), тобто корінь рівняння $f(x) = 0$ знайдено з необхідною точністю і за x^* можна взяти точку $x = \frac{a+b}{2}$.

Виразимо ці міркування у вигляді програмного коду (операція синтезу й абстракції):

```
> (define (метод_дихотомії f a b eps)
    (let ((x (/ (+ a b) 2)))
    (if (точність-досягнута? a b eps)
       x
       (if (> (* (f a) (f x)) 0)
           (метод_дихотомії f x b eps)
           (метод_дихотомії f a x eps)))))
```

Функція `точність-досягнута? x y` перевіряє, чи досягнута необхідна точність для знаходження кореня x^* на проміжку $[a, b]$:

```
> (define (точність-досягнута? a b eps)
  (< (abs (- a b)) eps))
```

Завдяки використанню функцій як значень, можна отримати компактний розв'язок для будь-якої функції, що задовольняє умову задачі:

```
> (метод_дихотомії sin 2.0 4.0 0.0001)
3.141571044921875
```

Таким чином, перевагами ФМП підходу для формування мислительних операцій є:

1) можливість подати розв'язок будь-якої задачі у вигляді композиції функцій (аналізуючи умову задачі та визначаючи необхідні функції, формується здатність до аналізу й абстракції, об'єднуючи ці функції в одну – до синтезу й абстракції);

2) можливість використовувати функції як значення (у випадку розв'язування задачі на знаходження узагальненої суми та знаходження кореня

рівняння методом дихотомії формується здатність до абстракції й узагальнення).

Висновки до розділу 1

1. Сучасна освітня парадигма передбачає розробку методичних систем навчання всіх дисциплін на основі компетентнісного підходу, впровадження якого у процес навчання надає можливість його гуманізувати, підвищити професійну мобільність майбутніх фахівців та створити умови для включення особистості спеціалістів в систему неперервної освіти. Для досягнення цієї мети мають бути сформовані такі складові компетентності, як гносеологічна, праксеологічна, аксіологічна та соціально-комунікативна.

2. Систему компетентностей, формування яких передбачається у майбутнього вчителя інформатики в процесі навчання у педагогічному університеті, утворюють такі 3 групи компетентностей: ключові, загально-професійні та спеціальні професійні.

Аналіз психолого-педагогічної, методичної, спеціальної літератури з проблеми дослідження надав можливість узагальнити різні підходи до визначення переліку загальнопрофесійних компетентностей у такий спосіб: методичні, науково-дослідницькі, психолого-педагогічні, організаційно-управлінські, комунікативні, ІКТ-компетентності вчителя.

При цьому ключові ІКТ-компетентності набувають свого подальшого розвитку у підготовці майбутнього вчителя як загальнопрофесійні, а у підготовці майбутнього вчителя інформатики вони розвиваються у спеціальних професійних компетентностях.

3. В структурі блоку спеціальних професійних компетентностей учителя інформатики одними з найбільш значущих є компетентності з програмування, формування яких можливе на основі різних підходів.

Формування у студентів напряму підготовки «Інформатика*» педагогічних університетів компетентностей з програмування на основі функціонального підходу надає можливості фундаменталізації процесу навчання на основі широкого застосування моделей та методів математичної інформатики.

4. З метою оцінювання у студентів рівня сформованості компетентностей з програмування були виділені 4 рівні (низький, достатній, середній та високий). Рівень сформованості компетентностей з програмування у студентів визначається через сформованість кожної з їх складових, що вимагає виділення показників їх розвиненості, незалежних від використовуваного підходу до навчання програмування. Це надає можливість на основі єдиних критеріїв порівнювати ефективність різних підходів до формування компетентностей з програмування.

5. Серед психолого-педагогічних особливостей формування у студентів молодших курсів компетентностей з програмування на основі ФП слід відзначити: кризовий віковий період студентів, швидкий розвиток всіх психічних функцій; необхідність розробки всіх компонентів методичної системи навчання, що сприятимуть формуванню позитивної навчально-пізнавальної мотивації.

Основні результати першого розділу опубліковано у роботах [169; 173; 174; 175; 176; 177; 178; 184; 185; 186; 189; 190; 193; 194].

РОЗДІЛ 2
МЕТОДИКА ФОРМУВАННЯ КОМПЕТЕНТНОСТЕЙ З ПРОГРАМУВАННЯ НА ОСНОВІ ФУНКЦІОНАЛЬНОГО ПІДХОДУ У СТУДЕНТІВ НАПРЯМУ ПІДГОТОВКИ «ІНФОРМАТИКА*» ПЕДАГОГІЧНИХ УНІВЕРСИТЕТІВ

2.1 Мета та зміст курсу «Вступ до програмування» для студентів напряму підготовки «Інформатика*» молодших курсів педагогічних університетів

Традиційна структура методичної системи навчання запропонована А. М. Пишкало [236]. В ній використовується системний підхід стосовно компонентів процесу навчання (всі компоненти утворюють єдине ціле із визначеними внутрішніми зв'язками). Згідно з цією структурою, методична система навчання – це сукупність ієрархічно пов'язаних компонентів: цілей навчання, змісту, методів, засобів і форм організації навчання (рис. 2.1).

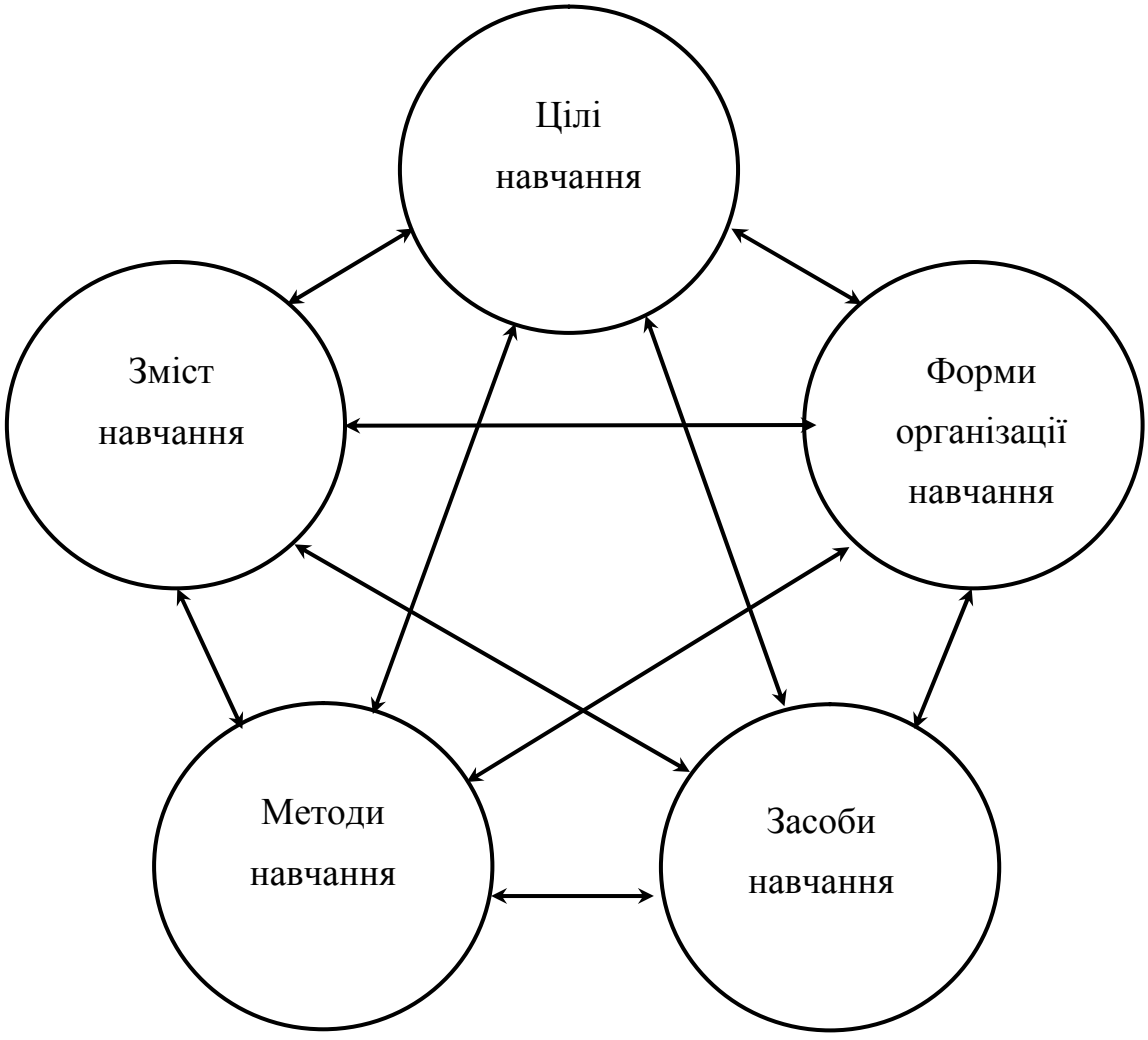


Рис. 2.1. Структура методичної системи навчання (за А. М. Пишкало)

Враховуючи особливості науки інформатики (а, відповідно, і навчального предмету інформатики) – нестабільність і стрімкий розвиток, висока чутливість до змін, що відбуваються в науково-технічній та соціально-економічній сферах – рядом дослідників [201; 251; 300] було обґрунтовано необхідність виокремлення технологічної підсистеми в структурі методичної системи навчання, що надало можливість максимально відобразити взаємовпливи всіх компонентів: цільового (відповідає на питання «Навіщо навчати?»), змістового (питання «Чому навчати?») та технологічного («Як навчати?») (рис. 2.2).

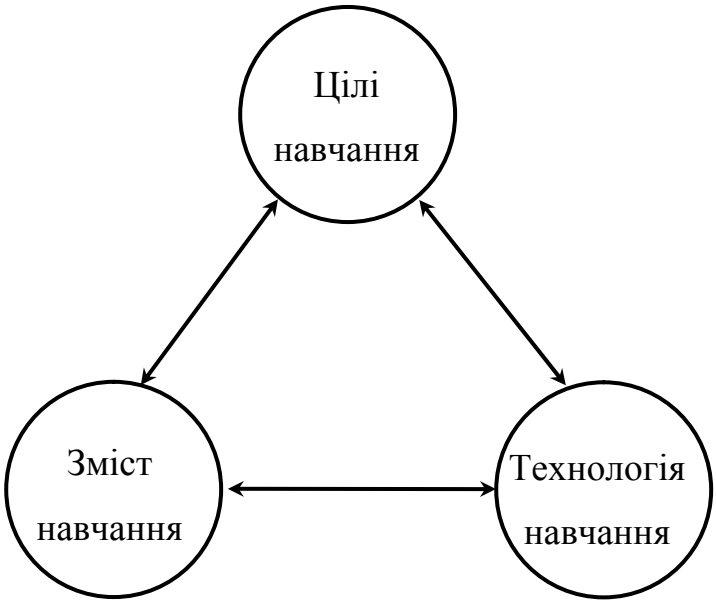


Рис. 2.2. Структура методичної системи навчання з виділеною підсистемою «технологія навчання»

Виходячи з такої структури, визначають цільовий, змістовний та технологічний компоненти методичної системи навчання.

Мета (*ціль*) навчання – ідеальне передбачення кінцевих результатів навчання; те, до чого прагнуть студенти, викладачі. За традиційним підходом до визначення процесу навчання через знання, уміння та навички, він передбачає три основні групи взаємопов’язаних цілей: 1) освітня – формування у студентів наукових знань, спеціальних й загально-навчальних умінь і навичок; 2) розвиваюча – розвиток мови, мислення, пам’яті, творчих здібностей, рухової та сенсорних систем; 3) виховна – формування світогляду, моралі, естетичної культури тощо [284,

229].

У зв’язку з швидкими темпами оновлення засобів сучасних ІКТ, зменшенням періоду застарівання фактичного матеріалу, вчитель інформатики повинен бути готовий до самонавчання та саморозвитку впродовж всього життя. Тому, одним з необхідних сьогодні результатів здобуття освіти у педагогічному університеті є набуття майбутніми вчителями інформатики інформатичних компетентностей на основі здобутих знань з фундаментальних та прикладних розділів інформатики, що дасть їм змогу у майбутньому швидко опановувати нові технології навчання, засоби сучасних ІКТ для розв’язування професійних та прикладних задач [246, 49].

Таким чином, *метою* навчання програмування майбутніх учителів за компетентнісного підходу є формування у студентів компетентностей з програмування.

Особливістю формування компетентностей з програмування є їх взаємозв’язок з іншими, як спеціальними професійними (компетентностями з математичної інформатики, з інформаційних технологій та з фундаментальних природничо-математичних дисциплін), так і з загальнопрофесійними (ІКТ-компетентностями) та ключовими (навчальними, соціальними, ІКТ та підприємницькими).

Формування компетентностей з програмування має акумулюючий характер: нові знання, що здобувають студенти, доповнюють їхню систему знань; уміння, навички, способи діяльності, яких набувають студенти, інтегруються з уже сформованими. Застосування здобутих знань, сформованих умінь, навичок, досвіду навчально-пізнавальної діяльності при розв’язуванні навчальних задач, задач майбутньої професійної діяльності та рефлексування цього процесу приводить до формування компетентностей певних рівнів. Тому, розглядаючи процес набуття майбутніми вчителями інформатики компетентностей з програмування при вивченні деякої дисципліни слід спочатку проаналізувати попередній етап формування їх основних складових та визначити шляхи подальшого їх формування [246, 62].

Формування компетентностей з програмування у майбутніх учителів інформатики в педагогічному університеті розпочинається в процесі навчання курсу «Шкільний курс інформатики», «Основи інформатики та обчислювальної техніки» або «Вступ до програмування».

Стосовно терміну «компетентність» будемо використовувати поняття «формування компетентності» – діяльність вчителя щодо організації засвоєння певного елемента соціального досвіду (поняття, дії) студентом [279, 25], та поняття «набуття компетентності», підкреслюючи таким чином те, що лише шляхом власної діяльності щодо пізнання соціального досвіду і його критичного осмислення, особистої творчості можна досягти компетентностей.

Наступним компонентом методичної системи навчання є зміст навчання. Спершу дамо визначення цього поняття згідно законодавчих документів.

Зміст вищої освіти – обумовлена цілями та потребами суспільства система знань, умінь і навичок, професійних, світоглядних і громадянських якостей, що має бути сформована в процесі навчання з урахуванням перспектив розвитку суспільства, науки, техніки, технологій, культури та мистецтва [111].

Зміст навчання у широкому розумінні – структура, зміст і обсяг навчальної інформації, засвоєння якої забезпечує особі можливість здобуття вищої освіти і певної кваліфікації [111].

Зміст навчання у вузькому розумінні (на рівні навчального предмету) – система знань з певної наукової галузі, практичних вмінь і навичок та способів діяльності, якими повинен оволодіти студент у процесі навчання [284, 230].

Зміст навчання з програмування визначається основними державними документами про освіту: Законами України «Про освіту» [113], «Про вищу освіту» [111], «Про національну програму інформатизації» [112], Державною національною програмою «Освіта. Україна XXI століття» [90], специфікою освітньої системи, освітнього закладу; навчальними планами та програмами.

Зміст освіти є одним із факторів економічного і соціального прогресу суспільства і повинен бути орієнтований на забезпечення самовизначення особистості, створення умов для її самореалізації; розвиток суспільства; посилення та вдоско-

налення правової держави.

Визначення змісту курсу з початків програмування необхідно здійснювати з врахуванням принципів, спільних як для будь-якого навчального курсу, так і властивих для курсів з програмування [33, 430; 84, 78; 85, 44; 144, 62; 197, 60; 246, 83]:

1. *Принцип відповідності навчальним цілям.* Ціллю навчання основ програмування є формування компетентностей студента, причому не лише з програмування, а й інших спеціальних професійних, загальнопрофесійних та ключових компетентностей.

2. *Принцип науковості і посильної складності.* Вимога науковості передбачає взаємозв'язок теорії, розробки, аналізу і оцінювання ефективності, реалізації і застосування алгоритмів. Зміст курсу повинен складатися з тих розділів і тем, які важливі для практики програмування незалежно від обраного підходу до навчання програмування. В той же час зміст курсу мусить містити всі необхідні компоненти для його засвоєння, тобто не містити «білих плям», а всі завдання, що пропонуються студентам мають або бути посильними, або ж орієнтованими на зону найближчого розвитку.

3. *Принцип фундаментальності.* За твердженням Мері Шоу навчання інформатики має організовуватись «навколо ідей, а не навколо артефактів…. Машинобудівні інститути не викладають проектування бойлера – вони викладають термодинаміку» [21]. Оскільки компетентнісний підхід є одним із напрямів фундаменталізації, принцип фундаментальності покладено в основу навчання.

З іншого боку, оскільки «основна ідея концепції фундаментального навчання програмування – виокремлення та поєднання у змісті навчання кожної теми відповідних математичних теорій, абстракцій (класичних алгоритмів і структур даних) та їх реалізацій на обраній мові програмування» [306], реалізація принципу фундаментальності досягається за рахунок фундаментальної математичної основи ФП (λ-числення та числення комбінаторів) (див. п. 1.3.2).

4. *Принцип відкритості.* Цей принцип передбачає можливість корекції змісту курсу залежно від освітнього напряму підготовки, без порушення цілісності

фундаментального ядра дисципліни. У розробленому курсі можливість реалізації даного принципу досягається через добір тематики проектів.

5. *Принцип сучасності.* Швидкий розвиток ІКТ вимагає постійного оновлення навчальної програми, що для учителів інформатики є особливо актуальним з огляду на особливості їхньої майбутньої професійної діяльності в умовах широкого впровадження засобів сучасних ІКТ у процес навчання школи.

6. *Принцип перспективності.* Цей принцип передбачає формування у студентів готовності до подальшого навчання протягом всього життя, що надасть можливість їм бути здатними розв'язувати професійні проблеми у майбутньому. У зв'язку із впровадженням компетентнісного підходу в навчанні реалізація цього принципу є безперечною, оскільки навчальні компетентності визначені одними з ключових як вітчизняними, так і європейськими вченими.

7. *Принцип вирівнювання знань.* Зміст курсу з початків програмування повинен включати модуль, під час вивчення якого буде здійснено початкове опанування мови програмування. Як вже зазначалось, особливістю мов ФП є їх нескладний синтаксис та семантика, тому даний принцип у розробленому курсі реалізується за незначний час: у модулі «Основи синтаксису» під час першої теми «Основи Scheme» студенти отримують відомості про основні складові мови програмування, вчаться записувати вирази у Scheme та визначати змінні і функції.

8. *Принцип послідовності і систематичності навчання.* Кожна наступна складова курсу повинна спиратися на попередній матеріал. На рис. 2.3 наведено послідовність тем курсу «Вступ до програмування». Для реалізації принципу систематичності обов'язковою є формулювання теми та мети на аудиторних заняттях, демонстрація можливості і способів досягнення мети заняття на конкретних прикладах і задачах.

10. *Принцип наочності змісту і діяльності*. Цей принцип для програмування (а особливо ФМП) реалізувати досить просто (порівняно з іншими навчальними дисциплінами та іншими підходами до програмування): адже обов'язковим етапом розв'язування будь-якої задачі є реалізація алгоритму за допомогою комп'ютера. Оскільки при ФП програмування функцій можливе ще на початкових

етапах вивчення курсу, то практично будь-який фрагмент коду можна визначити як функцію та переконатись у його дієвості. Особливо актуально це при програмуванні графічного інтерфейсу програми: для візуалізації роботи з деякими графічними елементами досить кількох рядків програмного коду (див. п. 2.3.3).

Рис. 2.3. Послідовність тем курсу «Вступ до програмування»

11. *Принцип створення відповідних умов для функціонування процесу навчання*. Особливістю курсів з інформатики є їх переважне проведення у комп'ютерних аудиторіях, що зумовлює підвищені вимоги до навколишнього середовища – необхідно дотримуватись відповідних санітарно-гігієнічних вимог. Але окрім цього, важливе значення має і створення сприятливої морально-психологічної атмосфери, що цілком залежить від рівня методичних, психолого-педагогічних, організаційно-управлінських та комунікативних компетентностей учителя.

12. *Принцип забезпечення оперативного контролю і самоконтролю в навчанні*. Для виявлення результативності та можливої корекції подальшого процесу навчання необхідно вчасно діагностувати рівень сформованості компетентностей з програмування. Важливе значення має різноманітність форм контролю: в розробленому курсі це і усне опитування на лабораторних заняттях, і проходження тес-

тів в ЕСПН Moodle, і виконання та захист лабораторних та індивідуальних завдань, модульний контроль. Для попередження виникнення стресових станів, що можуть виникнути на занятті під час тестування, студенти мають можливість самостійно пройти попереднє тестування в ЕСПН Moodle.

13. *Принцип індивідуалізації і колективності навчання*. Цей принцип передбачає необхідність врахування індивідуальних особливостей кожного студента: як фізичних (наприклад, студенту з вадами зору доцільним буде при роботі в парі доручити опис алгоритму розв'язування задачі, а реалізацію алгоритму покласти на іншого студента, або ж надавати матеріал у роздрукованому вигляді та ін.), так і психологічних (особливості сприйняття, уваги, мислення).

Принцип колективності у навчанні досягається шляхом використання групових форм організації роботи, однією з яких є і проектна форма. Завдяки реалізації цього принципу відбувається формування соціально-поведінкової та ціннісно-мотиваційної складових компетентностей з програмування.

14. *Принцип свідомості, активності і самостійності студентів*. Для реалізації принципу свідомості необхідно сприяти розумінню студентами сфери застосувань здобутих знань та вмінь, вмінню пояснити переваги того чи іншого методу при розв'язуванні задачі, вмінню обирати засоби для вирішення поставлених задач та обґрунтовувати вибір, тобто сприяти формуванню праксеологічної та аксіологічної складових компетентностей з програмування.

Реалізація принципу активності та самостійності студентів відбувається переважно в процесі формування елементів ціннісно-мотиваційної складової компетентностей з програмування: такі властивості студентів, як наполегливість, критичність мислення, здатність до цілепокладання, внутрішня мотивація до опанування програмуванням, прагнення до самовдосконалення, самостійність та ін.

Окрім розглянутого, добір змісту навчального матеріалу має здійснюватися з врахуванням і інших дидактичних принципів навчання: розвиваючого і виховного характеру навчання, врахування вікових та індивідуальних особливостей, міцності і системності знань, зв'язку змісту навчання з життям, практикою та ін.

Згідно документу [6, 65] програма будь-якого курсу з інформатики має від-

повідати таким вимогам:

– застосування методик навчання, що підкреслюють різницю між викладанням і навчанням (самоосвітою) та стимулюють студентів незалежно мислити;

– навчання студентів на творчих задачах та вправах, що стимулюють їх ініціативність;

– постійне оновлення обладнання та програмного забезпечення;

– ознайомлення студентів з інформаційними ресурсами та стратегіями оновлення своїх знань;

– заохочення колективного навчання та використання ІКТ для забезпечення взаємодії груп учнів;

– переконання студентів у необхідності продовження професійного розвитку та самовдосконалення протягом всього життя.

Як бачимо, ці вимоги сприяють формуванню у студентів ціннісно-мотиваційної та соціально-поведінкової складових компетентності.

Зміст курсу, спрямованого на формування у студентів напряму підготовки «Інформатика*» педагогічних університетів компетентностей з програмування – «Вступ до програмування», відображений у навчальному посібнику «Схематичне програмування (початки програмування: функціональний підхід)» [191]. Посібник складається з двох частин: *основи програмування мовою Scheme*: опис мови, запис виразів у Scheme, визначення змінних та функцій, прості типи даних: числовий тип (number), логічний тип (boolean), тип знак (character), символьний тип (symbol); керування виконанням програми: умовні вирази, циклічні вирази, рекурсивні функції, виконання блоку дій; похідні типи даних: рядок (string), пара (pair), список (list), вектор (array); введення/виведення, робота з файлами та *практика програмування мовою Scheme* (проекти із застосування моделей та методів математичної інформатики): «Дилема ув'язненого», «Психотерапевт», «Мінімальна система комп'ютерної алгебри», «Експертна система», «Розробка графічного інтерфейсу проекту».

2.2 Методи, засоби та форми організації навчання у процесі формування компетентностей з програмування на основі функціонального підходу

2.2.1 Методи навчання для формування у студентів напряму підготовки «Інформатика*» педагогічних університетів компетентностей з програмування. Окрім цільового та змістового компонентів, методична система навчання містить ще технологічний, до складу якого входять: методи, засоби та форми організації навчання.

Методи навчання (гр. methodos – шлях пізнання, спосіб знаходження істини) – це впорядковані способи взаємопов'язаної, цілеспрямованої діяльності педагога й студентів, спрямовані на ефективне розв'язання навчально-виховних завдань [139, 251].

Оскільки методи навчання … «є категорією історичною, … змінюються зі зміною цілей та змісту навчання» [142, 86], тому у зв'язку з переорієнтацією освіти зі знаннєвого підходу на компетентнісний, виникає необхідність переглянути і методи навчання в методичній системі формування у студентів педагогічних університетів компетентностей з програмування.

Існують різні підходи до класифікації методів навчання.

Якщо в якості критерію обрати *джерело знань*, розрізняють словесні (розповідь, пояснення, бесіда, лекція, інструктаж), наочні (ілюстрація, демонстрація, спостереження) та практичні (лабораторна робота, практична робота, вправи, робота на виробництві) методи навчання.

За *характером навчання* (ступенем самостійності та творчості) методи розділяють на:

– пояснювально-демонстраційні (студенти сприймають знання в «готовому» вигляді);

– репродуктивні методи (застосування вивченого матеріалу на основі зразка або правила);

– метод проблемного навчання (педагог ставить проблему, формулює пізнавальне завдання, а потім, розкриваючи систему доведень, порівнюючи погляди, різноманітні підходи, показує способи розв'язання, поставленого завдання, при

цьому студенти стають свідками і співучасниками наукового пошуку);

– частково-пошуковий або евристичний (організація активного пошуку розв'язання поставлених або самостійно сформульованих пізнавальних завдань, над якими студенти працюють самостійно або під керівництвом педагога на основі евристичних програм та вказівок; такий метод – надійний спосіб активізації мислення, пробудження інтересу до пізнання);

– дослідницький (після аналізу матеріалу, постановки проблеми і визначення завдань студенти самостійно опрацьовують наукові джерела, проводять спостереження та виконують інші дії пошукового характеру). Використання дослідницьких методів навчання сприяє формуванню не лише гносеологічної та праксеологічної складових компетентностей з програмування, а й аксіологічної (ініціативність, самостійність, відповідальність, творчий пошук та ін.).

М. П. Лапчик [144], О. І. Бочкін [56], В. В. Лаптєв, Н. І. Рижова, М. В. Швецький [142, 91], Н. В. Морзе [197, 84] виділяють ще *спеціальні* (частково-дидактичні) *методи навчання інформатики*, до яких відносять метод доцільно дібраних задач, метод демонстраційних прикладів, обчислювальний експеримент та програмування.

Метод доцільно дібраних задач. Розв'язування задач в методиці навчання інформатики розглядається як метод навчання, як засіб навчання і як мета навчання. Задачі використовуються і для мотивації певної діяльності студентів, і для закріплення теоретичного матеріалу, і для вивчення нового теоретичного матеріалу.

Суть *методу доцільно дібраних задач*:

– діяльність викладача полягає в побудові системи задач, причому виконання кожної задачі базується на виконанні попередньої і спрямовано на вирішення сформульованої проблемної ситуації;

– діяльність студента полягає у вирішенні деякої задачі, сформульованої викладачем;

– взаємодія викладача зі студентом полягає в тому, що викладач у разі необхідності може «втручатися» в діяльність студента при формулюванні кожної

задачі або в процесі її розв'язування [142, 106].

В проекті «Мінімальна система комп'ютерної алгебри» окрім визначення стандартних функцій (піднесення до степеня `^`, ділення без остачі `remainder`, залишку від ділення `quotient`) студенти отримують завдання реалізувати функції, результат яких їм ще невідомий («порозрядне і» `and`, «порозрядне або» `or`, «виняткове або» `xor`, «арифметичний зсув вліво» `shl`,«арифметичний зсув вправо» `shr`). В процесі розв'язування цих завдань у студентів, окрім праксеологічної складової компетентностей з програмування, відбувається внесок і у формування гносеологічної складової.

У науковців немає одностайності стосовно розуміння терміну «*програмування*»: *у широкому сенсі* воно охоплює *всі етапи розв'язування задачі на комп'ютері* (від формулювання мети проекту до тестування й налагодження програми та аналізу результатів) – в такому разі програмування *співпадає з обчислювальним експериментом*, *у вузькому сенсі* програмування розглядається лише як *процес кодування алгоритму мовою програмування* і *є складовою частиною обчислювального експерименту з використанням комп'ютера* [142, 101].

Властивості *методу програмування і методу обчислювального експерименту :*

– є практичними методами навчання (за джерелом знань);

– діяльністю викладача є керування практичною діяльністю студентів;

– навчальною діяльністю студентів є практична діяльність;

– характер розумової активності і самостійності студентів може бути як репродуктивним, так і частково-пошуковим або дослідницьким;

– логіка навчально-пізнавальної діяльності студентів може бути індуктивна, дедуктивна, індуктивно-дедуктивна;

– характер керування діяльністю студентів з боку викладача – можливе пряме або опосередковане керування [142, 103].

Процес програмування містить і репродуктивну, і творчу діяльність студентів, оскільки без відтворення певних знань про мову програмування (репродуктивна діяльність) неможливо написати навіть найпростішу програму (творча діяль-

ність студентів) [142, 103].

У процесі формування компетентностей з програмування метод програмування і метод обчислювального експерименту є основними методами навчання, без них важко уявити жодне заняття.

Метод демонстраційних прикладів.

Програмування – творча, дослідницька діяльність, мистецтво конструювання. Його неможливо звести до використання готових рецептів. Тому одним із найадекватніших методів навчання програмування є ретельний відбір та розгляд характерних прикладів [60, 11].

Використання методу демонстраційних прикладів надає можливість студентам на початкових етапах навчання долучитися до процесу створення функціонально довершених програмних продуктів. І хоча спочатку завданням студентів буде аналіз текстів та передбачення результатів програм, знаходження заздалегідь спланованих викладачем помилок, модифікація та вдосконалення проектів, надалі завдання полягатимуть і у самостійній розробці таких продуктів.

У розробленому курсі використовуються такі види демонстраційних прикладів: перший модуль – ілюстрація певних аспектів синтаксису і семантики мови програмування Scheme та демонстрація реалізації простих типів даних; у другому модулі, окрім цього, студентам демонструється реалізація класичних алгоритмів (пошук, сортування) та похідних типів даних і функцій для роботи з ними; у четвертому модулі – демонстрація реалізації об’єктів.

У третьому модулі курсу завданням на початковому етапі роботи над будь-якою темою є «візуальне осмислення тексту» програми [142, 98], для якої вже визначено «базовий» набір функцій. Оскільки у мовах ФП будь-яку функцію можна представити як послідовність викликів невеликих за розміром функцій, студенти мають можливість безпосередньо впевнитись у роботі тієї чи іншої функції, викликавши її з відповідними аргументами.

Частина назв форм навчання інформатики (див. п. 2.2.3) виступають і в якості назв методів навчання: це, насамперед, л*екція*, *метод проектів* та *лабораторно-обчислювальний практикум* (за методом «занурення») [252, 209].

Занурення відноситься до методів концентрованого навчання інформатики.

Найпоширеніша форма реалізації занурення в навчанні інформатики – *лабораторно-обчислювальний практикум* (див. п. 2.2.3).

Особливістю компетентнісного підходу є особлива увага до методів та форм активного навчання студентів та контрольних завдань, які мають виключити можливість механічного заучування [303, 108].

Як відзначається в [144, 355], найбільш адекватним методом навчання комп'ютерному моделюванню (а відповідно, і програмуванню) є *метод проектів*, що є одним із основних сучасних інноваційних методів активного навчання. Виділяють індивідуальні й групові, локальні й телекомунікаційні, ІТ-проекти. Характерною ознакою *навчальних ІТ-проектів* є самостійна дослідницька діяльність їх учасників, пов'язана з розв'язанням навчальної проблеми, що має на меті отримання практичного результату у вигляді програмного продукту та спирається на більшості або на кожному своєму етапі на використання інформаційних технологій [207, 168].

Використання методу проектів висуває певні вимоги як до викладачів, так і до студентів. Так, з боку викладача, необхідний високий рівень методичних компетентностей. Адже для методу проектів характерним є наявність організаційного етапу: після вибору теми проекту, викладач повинен до найменших дрібниць прорахувати всі складнощі (пошук необхідних відомостей, вибір засобів реалізації та ін.), що можуть виникнути під час виконання проекту, щоб, у разі потреби, спрямувати студентів у правильному напрямі.

Стосовно студентів, то у них має бути певний рівень знань та вмінь. Тому у розробленому курсі передбачається використання методу проетів у третьому та четвертому модулях – після того, як у студентів сформовані основні компоненти гносеологічної та праксеологічної складових компетентностей з програмування:

– знання основних етапів розв'язування задач;

– знання складових мови програмування;

– знання основних форм для керування виконанням програми;

– знання простих і похідних типів даних, функцій для роботи з ними;

– уміння пояснити призначення та функції існуючої програми;

– уміння знайти помилки в логіці розв’язання задачі;

– уміння описати етапи розробки програм;

– уміння розробити функції та обґрунтувати пріоритетність використання того чи іншого виразу для їх створення;

– уміння пояснити та продемонструвати процес створення похідних типів даних;

– уміння спроектувати, описати, перевірити та проаналізувати результати виконання програми.

Кожен проект повинен відповідати основним показникам якісного програмного забезпечення [84, 113]:

– ефективність: швидкість виконання, мінімальний об’єм необхідної пам’яті;

– зручність у використанні: наявність дружнього інтерфейсу і необхідної документації;

– надійність: безвідмовність, захищеність і безпека;

– зручність супроводу: можливість удосконалення і модифікації продукту при внесенні змін до проекту.

Тематику проектів (на прикладі проекту «ЕС») див. у п. 2.3.2. Викладач, у разі обрання студентами власної теми проекту, оцінюючи рівень його складності, може корегувати цей вибір, при цьому бажано, щоб корегування носило виключно рекомендаційний характер [84, 113].

Використання методу проектів сприяє формуванню у студентів здатностей складати план виконання проекту, об’єднуватися в групи і розподіляти обов’язки, визначати та знаходити необхідні для виконання проекту матеріали, узагальнювати отримані відомості, представляти результати та здійснювати публічні виступи.

Оцінка розроблених проектів здійснюється у 2 етапи. На обох етапах викладач оцінює проект за такими критеріями: алгоритмічна складність, завершеність проекту, об’єм виконаної роботи, якість призначеного для користувача інтерфейсу та супровідна документація [84, 115].

Перший етап – захист проектів без розробленого графічного інтерфейсу (до виконання проекту «Графічний інтерфейс»). Для кожного із проектів розроблено кінцеві терміни здачі (через 2 тижні після початку роботи над проектом). На цьому етапі студенти мають можливість отримати за кожен проект по 3 бали (отримують усі учасники групи) залежно від відповідності проекту встановленим критеріям оцінювання. В залежності від участі кожного учасника групи у розробці проекту на цьому етапі студенти також можуть додатково отримати ще до 2 балів.

Другий етап – публічний захист програмних продуктівз графічним інтерфейсом. Це оцінювання відбувається після опрацювання всього курсу. На цьому етапі студенти мають можливість отримати ще по 3 бали кожен та до 2 балів в залежності від участі кожного учасника групи у розробці проекту.

Тобто, в сумі за всі проекти цього модуля студент має можливість отримати 40 балів (за 100-бальною шкалою). При цьому на першому етапі оцінювання він може отримати 4 * 5 = 20 балів, і на другому 4 * 5 = 20 балів.

Прикладом комбінованого методу навчання є *учіння через навчання*. Це метод навчання, при якому студенти самі – за допомогою викладача – готують і проводять заняття (це може стосуватися і його окремих частин). Викладач повинен подбати про те, щоб студенти інтенсивно спілкувалися й створювали довгострокові, пов’язані з матеріалом контакти, тобто, щоб студенти колективно продукували знання. Це відбувається найкраще в рамках невеликих дослідницьких проектів, в тому числі – телекомунікаційних (саме тому даний метод відноситься до комбінованих).

Перед розглядом нової теми викладач розподіляє матеріал малими дозами між групами студентів (максимально три студенти); кожна група одержує окрему частину матеріалу, а також завдання повідомити цей зміст всім іншим. Студенти, котрі одержали завдання, дидактично готують матеріал. Під час такої підготовки, що відбувається на занятті, викладач підтримує окремі групи та надає поради.

Необхідно зауважити, що учіння через навчання не є фронтальним заняттям, що проведене студентами: вони повинні постійно відповідними засобами переконуватися, що матеріал зрозумілий тим, кому він адресований (коротко запи-

тувати, узагальнювати, залучати до партнерської роботи). Тут викладач повинен втручатися, якщо він бачить, що комунікація не вдається або що застосовувані студентами прийоми мотивації не спрацьовують.

Переваги методу:

– інтенсивніше опрацювання матеріалу;

– набуття студентами, окрім гносеологічних та праксеологічних складових компетентностей з програмування, таких вмінь: здатність працювати в команді (соціально-комунікативна складова); активність, здатність до планування, цілеспрямованість, презентація, самосвідомість (аксіологічну складова).

До недоліків методу відносять більші часові витрати (у порівнянні з іншими методами навчання).

Учіння через навчання можна застосовувати і у великих групах (з кількістю учасників від 15 до 35) [252, 215].

Використання технічних засобів навчання (комп'ютера, проектора, мультимедійної дошки та ін.) в процесі навчання програмування сприяє використанню такого наочного методу як *демонстрація*.

Використовуючи демонстраційний екран, викладач показує різні навчальні елементи змісту курсу (елементи інтерфейсу, фрагменти програм, схеми, тексти й т.п.). При цьому викладач сам працює на комп'ютері, а студенти спостерігають за його діями або відтворюють їх [157, 58]. Існує можливість і самостійної роботи з розробленими матеріалами. Серед дидактичних функцій демонстрації основною є інформаційна (повідомлення студентам нового навчального матеріалу).

При виборі методів навчання мають враховуватись:

– мета навчання;

– спеціалізація навчального закладу;

– особливості навчальної дисципліни в цілому та конкретного заняття;

– індивідуальні особливості та компетентності студентів (фізичні особливості, риси характеру, рівень спеціальних предметних компетентностей), особливості групи;

– матеріально-технічна база навчального закладу;

– індивідуальні особливості та компетентності викладача (фізичні особливості, риси характеру, стосунки з групою, рівень загальнопрофесійних та спеціальних професійних компетентностей);

– ергономічні умови (час за розкладом, наповнюваність аудиторії, тривалість роботи за комп'ютером і т.д.) [157, 63].

2.2.2 Комп'ютерно-орієнтовані засоби навчання для формування у студентів напряму підготовки «Інформатика*» педагогічних університетів компетентностей з програмування на основі функціонального підходу. *Засоби навчання* – матеріальні й ідеальні об'єкти, які використовуються в освітньому процесі як носії відомостей (інформаційних ресурсів) та інструменти діяльності вчителя (викладача) й учнів (студентів), що застосовуються ними як окремо, так і спільно [284, 230].

До засобів навчання належать: природне і соціальне оточення, обладнання, підручники, книги, наукові видання, комп'ютери і комп'ютерні мережі з відповідним програмним забезпеченням та інформаційними ресурсами, зокрема електронні підручники, довідники, енциклопедії, електронні бібліотеки.

Існують різні класифікації засобів навчання. Одна з них – класифікація за *дидактичною функцією* [201, 158]:

– інформаційні засоби (підручники і навчальні допомоги);

– дидактичні засоби (таблиці, плакати, відеофільми, програмні засоби навчального призначення, демонстраційні приклади);

– технічні засоби навчання (аудіовізуальні засоби, комп'ютери, засоби телекомунікацій, відеокомп'ютерні системи, мультимедіа).

Згідно досліджень психологів людина запам'ятовує лише 10% прочитаного, 20% – почутого, 30% – побаченого. Якщо людина чує та бачить, рівень запам'ятовування підвищується до 50%, а якщо чує, бачить, а потім обговорює, то до 70%. Використання аудіовізуальних засобів скорочує на 40% необхідного часу для навчання і на 20% збільшує об'єм засвоєного матеріалу [119]. Таким чином, необхідне розумне поєдання різних видів методів та засобів навчання.

Серед засобів навчання виділяють (умовно) традиційні і комп'ютерно орієн-

нтовані засоби (КОЗН), які потрібно гармонійно поєднувати і взаємодоповнювати в процесі навчально-пізнавальної діяльності. Під КОЗН розуміють програмно-апаратні засоби й пристрої, що функціонують на базі комп'ютерної техніки, а також сучасних засобів і систем інформаційного обміну, забезпечення операцій щодо пошуку, збирання, накопичення, зберігання, опрацювання, подання і передавання повідомлень [201, 162].

Використання КОЗН сприяє [89, 51]:

– підвищенню мотивації, посиленню інтересу до навчальної діяльності та способів здобуття знань;

– індивідуалізації та диференціації навчання: індивідуальний темп навчання та методики подання навчального матеріалу;

– створенню позитивної соціально-психологічної атмосфери: відсутність категорично негативної оцінки власної діяльності формує у студентів позитивне ставлення до навчання, надає можливість отримувати інтелектуальну насолоду від нього, можливість самостійно пройти попереднє тестування усуває виникнення стресових ситуацій на заняттях;

– активнішому залученню студентів до інтенсивної, творчої навчальної роботи, самостійному здобуттю знань, опануванню сучасними методами наукового пізнання;

– підвищенню ефективності самостійної роботи;

– розширенню способів подання навчальних матеріалів та підвищенню наочності навчання;

– скороченню терміну вивчення кожного розділу навчального курсу, при цьому набуті знання залишаються у пам'яті значно довше і в подальшій практичній роботі скоріше оновлюються.

Оскільки зазвичай викладачі не мають безпосереднього впливу на оснащеність комп'ютерних аудиторій апаратним забезпеченням, в даній роботі детальну увагу приділено тільки програмному забезпеченню.

У роботах [176; 177; 178; 179; 191; 195] обґрунтовано вибір функціональної мови програмування Scheme в якості мови для початкового навчання програму-

вання. Серед вимог до вибору початкової мови для навчання програмування виокремлюють: простоту у використанні, структуру дизайну, потужні обчислювальні засоби, простий синтаксис, простоту введення/виведення, значимі назви ключових слів, швидкий зворотній зв'язок, наявність хорошого діагностичного інструментарію для тестування й налагодження тощо [250, 36].

Але з питанням вибору мови програмування безпосередньо пов'язане і питання вибору середовища навчання програмування, адже від того, наскільки зручним і природним є середовище стосовно розробки, налагодження та виконання програми, отримання довідки тощо буде залежати як кінцевий результат навчання програмування, так і сам процес навчання.

Для формування компетентностей з програмування на основі ФП необхідним є обґрунтований вибір простого та розширюваного середовища навчання мови функціонального програмування. Найбільш поширені середовища програмування мовою Scheme – DrRacket [18], EdScheme [27], MIT-GNU Scheme [14] та STk-ucb [8].

DrRacket. DrRacket є однією зі складових Racket – середовища програмування, в якому можлива реалізація широкого спектру навчальних та виробничих задач.

Окрім середовища для програмування DrRacket, Racket включає в себе компілятор «на льоту», інструменти для створення автономних виконуваних файлів, web-сервер, різноманітні бібліотеки, основну та допоміжну документацію як для початківців, так і для експертів, та багато іншого.

У середовищі Racket існує можливість вибору однієї з мов програмування з наведеного списку: R5RS, Pretty Big, Swindle (успадковані мови); Lazy Scheme, FrTime, Algol 60 (експериментальні мови).

Переваги DrRacket (у порівнянні з іншими середовищами програмування мовою Scheme – STk-ucb, EdScheme, MIT-GNU Scheme):

– різні модулі реалізації, можливість зміни мови (як вручну, так і автоматично при використанні певних бібліотек, які можуть використовуватись лише в певній мові);

– відкритість та мобільність;

– авторська локалізація доступна в офіційній поставці DrRacket.

Додаткова особливість середовища програмування DrRacket – можливість створення імен змінних рідною мовою.

Виходячи з цього, DrRacket (рис. 2.4) було обрано в якості основного середовища програмування для формування у майбутніх учителів інформатики компетентностей з програмування на основі функціонального підходу.

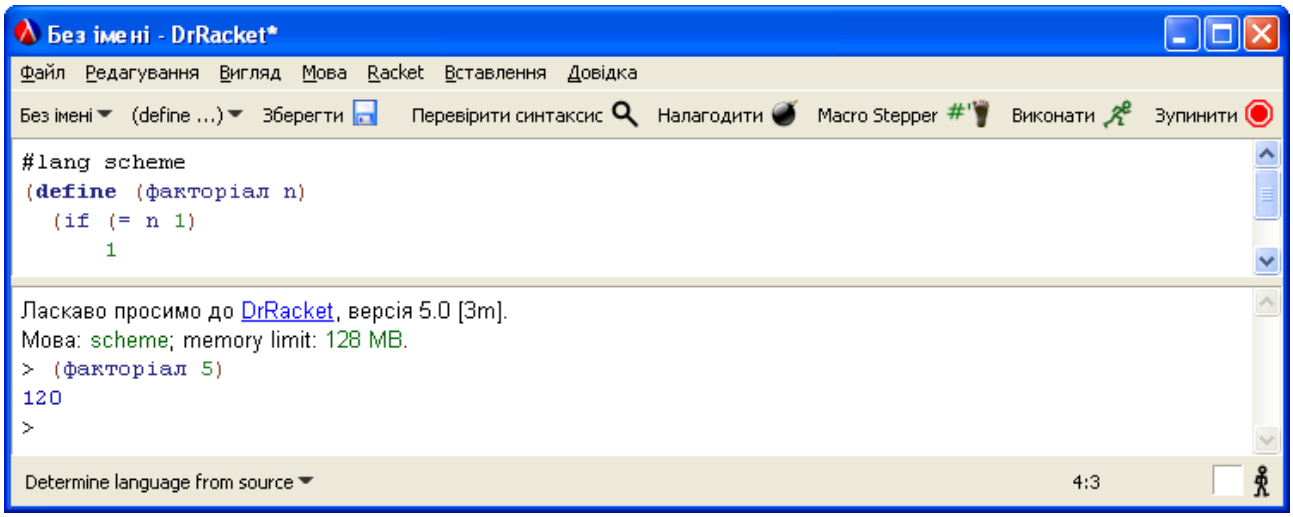


Рис. 2.4. Середовище програмування DrRacket

В додатку до авторського посібника [191] наведено опис локалізованої версії DrRacket та розглянуто її додаткові можливості.

Scheme-аплет. У зв'язку з поширенням мобільних технологій виникає необхідність створення мобільних засобів, які функціонують у web-середовищі. Для реалізації задач формування компетентностей з програмування на основі функціонального підходу в мобільному навчальному середовищі на основі інтерпретатора мови Scheme, реалізованого на Java (автор – Скотт Міллер), було створено 2 аплети, що надають інтерфейс українською та російськими мовами [179, 25] (рис. 2.5). Цей інтерпретатор може бути використано як додатковий компонент для швидкої оцінки S-виразів у процесі виконання тестових завдань у ЕСПН (вбудовується в існуючі web-сторінки).

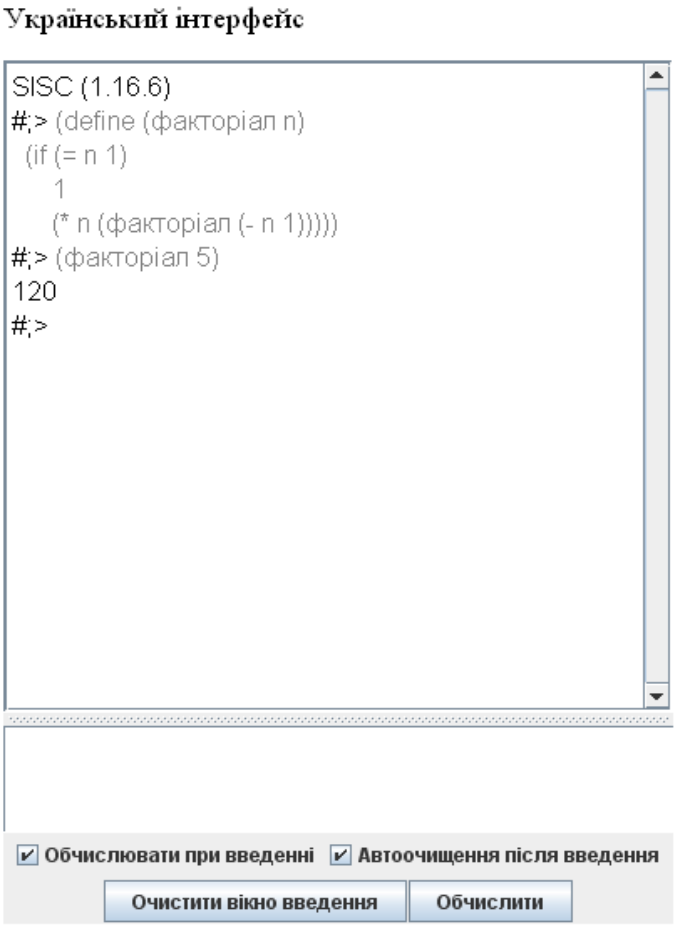


Рис. 2.5. Online-версія Scheme для мобільних пристроїв

ЕСПН Moodle. Серед КОЗН особливе місце належить ЕСПН, що надають можливість раціонально використовувати час та зусилля як студентів, так і викладачів. Сучасні ЕСПН містять необхідний набір засобів не лише для збереження і доставки навчальних ресурсів та організації навчальної діяльності, а й для управління навчальним процесом, обліку та контролю виконання різних видів навчальних робіт, контролю за використанням навчальних ресурсів, адміністрування окремих студентів та груп, організації взаємодії з викладачем, звітності тощо [66, 25].

В Україні найбільшою популярністю користуються такі платформи ЕСПН: Веб-клас ХПІ, Прометей, Херсонський віртуальний університет, ATutor, Claroline, Dokeos, e-Learning, LAMS, Learn Space, Moodle, OLAT, OpenACS, Sakai.

Детальний аналіз та порівняння ЕСПН стосовно засобів організації навчального процесу проведено у роботі К. Р. Колос [128, 41] (табл. 2.1).

Таблиця 2.1

Порівняльний аналіз ЕСПН

стосовно засобів організації навчального процесу

Система / Засоби	ATutor	Claro-line	Dokeos	LAMS	Moodle	OLAT	Open-ACS	Sakai
Рейтинг системи	5	4	4	6	1	6	3	2
Версія	2.0.2	1.10	2.0	2.3.5	2.1	7.0	5.6.0	2.7.1
Голосування	+	–	–	+	+	–	+	+
Опитування	+	–	–	+	+	+	+	+
Анкета	+	–	–	+	+	+	–	+
Урок	–	+	+	+	+	–	–	+
Відеоконференція	–	–	–	+	+	–	–	–
Форум	+	+	+	+	+	+	+	+
Чат	+	+	+	+	+	+	+	+
Глосарій	–	–	+	–	+	+	+	+
Wiki	–	–	+	+	+	+	+	+
Комунікатор (внутрішня система обміну повідомленнями)	+	–	–	+	+	+	–	–
Спілкування між студентами IMS	–	–	–	+	+	+	–	–
Обмін файлами	+	+	+	+	+	+	+	+
Система тестування	+	+	+	+	+	+	+	+
Підтримка зовнішніх тестів	–	–	–	–	+	+	–	+
Управління навчальним курсом	+	+	+	+	+	+	+	+
Організація різних способів представлення навчального матеріалу	±	+	+	+	+	±	±	±

Система / Засоби	ATutor	Claro-line	Dokeos	LAMS	Moodle	OLAT	Open-ACS	Sakai
Організація різних форм діяльності студентів	±	+	+	+	+	+	±	±
Електронна пошта	+	+	+	–	+	+	+	+
Планування (календар)	–	+	+	–	+	+	+	+
Віртуальна дошка	+	–	+	–	–	–	+	+
Пошук	+	–	–	–	+	+	+	–
Робота з групами	–	+	+	+	+	+	–	+
Допомога	+	+	–	+	+	+	–	+
Портфоліо	–	–	–	–	+	–	–	+
Система звітності	Слабко розвинена	Середньо розвинена	Середньо розвинена	Слабко розвинена	Розвинена, постійно розвивається	Слабко розвинена	Слабко розвинена	Розвинена, постійно розвивається
Обмеження на кількість слухачів	Немає	20000	1200000 осіб	Немає	Немає	>700	Немає	200000
Засоби розробки курсу	+	+	+	+	+	+	+	+
Мова програмування	PHP	PHP	PHP	Java	PHP	Java	Tcl	Java
Платформа	Windows, Linux, Unix, MacOS	Windows, Linux, Unix, MacOS	Windows, Linux, Unix, MacOS	Windows, MacOS	Windows, Linux, Unix, MacOS	Linux, Unix	Windows, Linux, Unix, MacOS	Windows, Linux, Unix, MacOS
Багатомовність (підтримка кількості мов)	Так, >50	Так, 37	Так, 39	Так, 19	Так, 70	Так, 16	Ні	Так, 29

Система / Засоби	ATutor	Claro-line	Dokeos	LAMS	Moodle	OLAT	Open-ACS	Sakai
СУБД	MySQL	MySQL	MySQL	MySQL	MySQL	MySQL	MySQL	My-SQL, Oracle, Hsqldb
Відповідність зі стандартом SCORM	+	+	+	–	+	+	–	+
Ліцензія	GNU/ GPL	GNU/ GPL	GNU/ GPL	GNU/ GPL	GNU/ GPL	Apache	GNU/ GPL	Apache 2.0

Згідно з результатами порівняльного аналізу ЕСПН Moodle є найзручнішою для підтримки процесу навчання. Саме тому її було обрано для розробки НМК «Вступ до програмування».

За даними офіційного сайту Moodle [15], станом на лютий 2013 р. у світі зареєстровано 74 975 сайтів (7 005 556 курсів) у 227 країнах, які використовують систему Moodle; в Україні зареєстровано 333 сайти.

Розглянемо детальніше вимоги до ЕСПН в умовах навчального закладу та характеристики ЕСПН Moodle, що відповідають цим вимогам [264, 333] (табл. 2.2).

Таблиця 2.2

Вимоги до ЕСПН в умовах навчального закладу та характеристики ЕСПН Moodle

Вимоги до ЕСПН	Характеристики ЕСПН Moodle
врахування реальних можливостей користувача (просте, інтуїтивне обслуговування на довільному комп'ютері, в будь-якій операційній системі і довільним підключенням до мережі, без необхідності інсталяції спеціального програмного забезпечення та обладнання)	для повноцінного використання ЕСПН Moodle необхідно і достатньо стандартного web-браузера, не передбачається ніяких спеціальних вимог до обладнання та операційної системи

Вимоги до ЕСПН	Характеристики ЕСПН Moodle
врахування потреб і можливостей викладача (просте керування змістом та навчально-пізнавальною діяльністю користувачів, легка комунікація з ними, можливість швидкого створення документів, простого надання доступу, впорядкування та опису різних типів даних, в тому числі мультимедійних)	наявність спеціальних інформатичних компетентностей з боку викладача не є обов'язковою, оскільки, як зазначалось, для повноцінного використання ЕСПН Moodle достатньо стандартного web-браузера
функціональна еластичність (нескладний початок роботи, можливість розширення наявних компонентів)	використання ЕСПН Moodle можливо почати з необхідних в даний момент компонентів, за необхідності – далі доповнюючи курс іншими елементами (це можливо завдяки модульній структурі ЕСПН Moodle)
доступність інструментів, що забезпечують можливість співпраці між користувачами (запис розмов, спілкування між групами)	можливість співпраці між учасниками курсу, як в синхронному режимі, так і асинхронному
врахування педагогічних вимог (наявність інструментів та засобів для підтримки всіх етапів і компонентів процесу навчання)	підтримка всіх етапів і компонентів процесу навчання: планування (вибір структури курсу, розробка курсу – додавання елементів або ресурсів), навчання (текстові сторінки, web-сторінки, посилання на файли або каталоги, завдання, журнал, глосарій), адміністрування (присутність, оцінки, логи, групи, ролі), оцінювання (тести, завдання, журнал, ігри, активність, логи, форум, словник), повідомлення результатів (форум, чат, повідомлення)

Для створеного в ЕСПН Moodle НМК було обрано формат «Структура», оскільки курс побудовано за модульним принципом. Вступна частина містить загальні відомості про курс: назву та мету, форум новин, програму курсу та шкалу оцінювання, анкету для виявлення початкового рівня сформованості компетентностей з програмування, необхідні матеріали до курсу (літературу та середовища програмування), глосарій до курсу і засоби для організації спілкування (чат – консультація, форум з актуальних питань та систему Skype) (рис. 2.6).

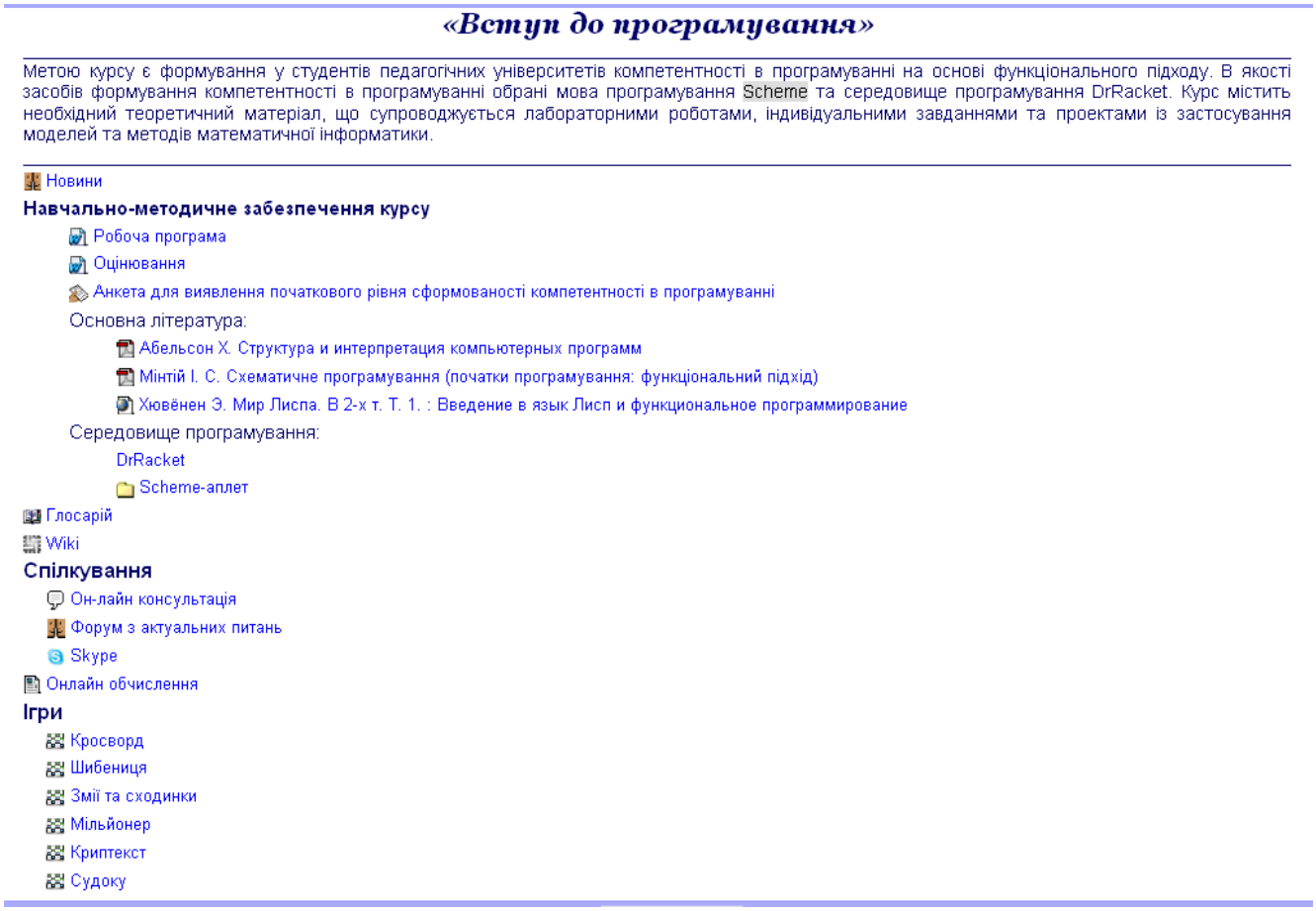


Рис. 2.6. Вступна частина курсу

Основна частина курсу містить чотири тематичні модулі (рис. 2.7).

Формування гносеологічної складової компетентностей з програмування відбувається переважно під час ознайомлення студентів з теоретичним матеріалом, для цього в курсі за допомогою ресурсу «Web-сторінка» було розроблено лекції (рис. 2.8).

Лекції створено на основі матеріалів посібника [191], але їх перевагою, порівняно з посібником є виокремлення ключових термінів, занесених до глосарію,

та наявність відеододатків, які були розроблені для більшої наочності.

Рис. 2.7. Основна частина курсу

1. Керування виконанням програми: умовні вирази

При виконанні програми може знадобитись здійснення певних перевірок та в залежності від їх результату виконання тих чи інших дій або ж виконання певних дій необхідну кількість разів чи доти, доки виконуються певні умови. В таких випадках використовують умовні або циклічні вирази.

2. Умовний вираз if

Розглянемо формулу для знаходження модуля числа:

$$|x| = \begin{cases} x, & x >= 0 \\ -x & x < 0 \end{cases}$$

Для розв'язання подібних задач можна скористатись примітивним умовним виразом if (if є примітивним умовним виразом, оскільки, використовуючи його, можна реалізувати всі інші умовні вирази). Загальна форма:

(if умова вираз-так вираз-інакше)

Якщо значенням умови є «істина», значенням умовного виразу буде значення виразу-так, якщо «хиба» – виразу-інакше.

Рис. 2.8. Фрагмент лекції до теми «Умовні вирази»,
розробленої у вигляді web-сторінки

Глосарій призначений для пояснення основних понять та ключових термінів, що використані в курсі. В розроблений глосарій будь-хто з учасників курсу може додавати нові записи, редагувати, оцінювати та коментувати існуючі записи, що також сприяє формуванню здатності оцінювати роботу інших, вести дискусію (соціально-поведінкова складова компетентностей з програмування).

Тип глосарію – основний, він має вигляд звичайного словника, відображається абетка. На рис. 2.9 показано загальний вид глосарію курсу. Занесені до глосарію терміни виокремлюються кольором у інших ресурсах курсу (рис. 2.10). Натиснувши на виокремленому терміні у навчальному ресурсі (1), отримаємо вікно з його означенням, яке наведене у глосарію (2).

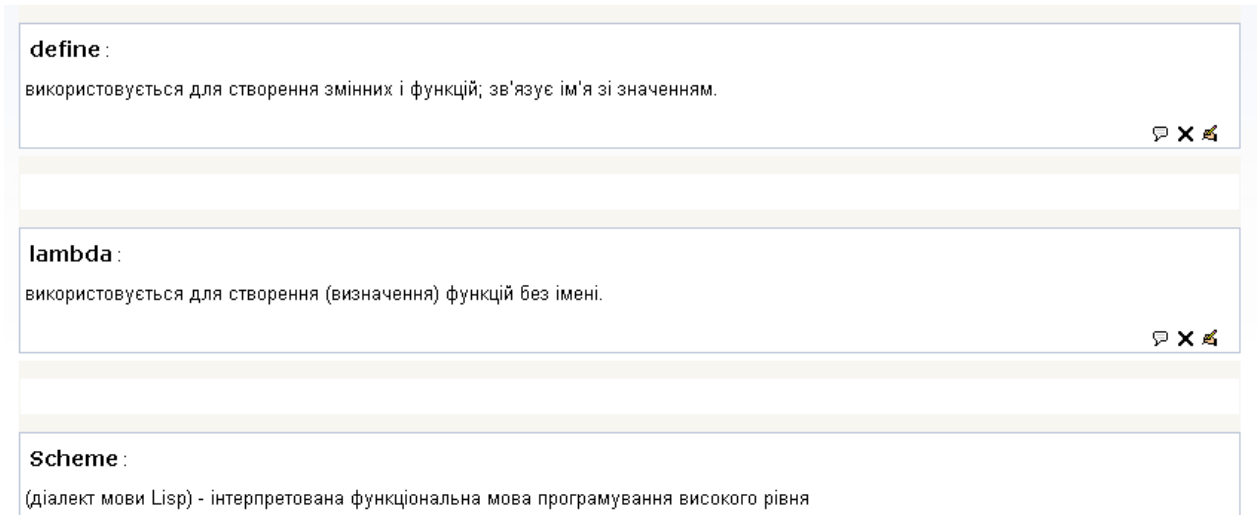


Рис. 2.9. Глосарій курсу

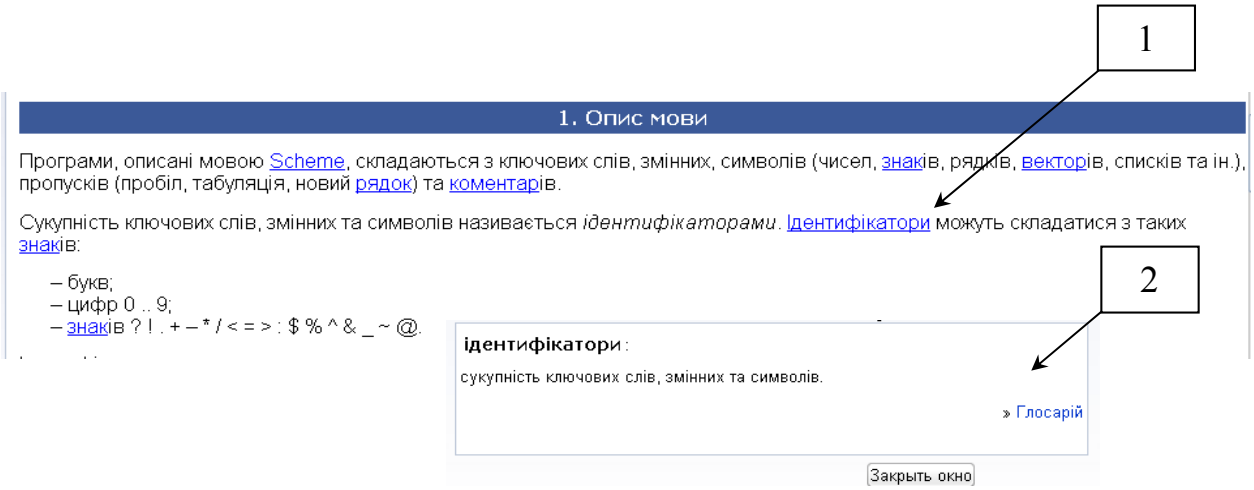


Рис. 2.10. Вигляд занесених до глосарію термінів

Відеододатки подаються студентам у двох форматах: у форматі відеофайлу та у форматі потокового відео (рис. 2.11). Відеоматеріали у форматі відеофайлів призначені для автономного перегляду, у форматі потокового відео – для онлайн перегляду.

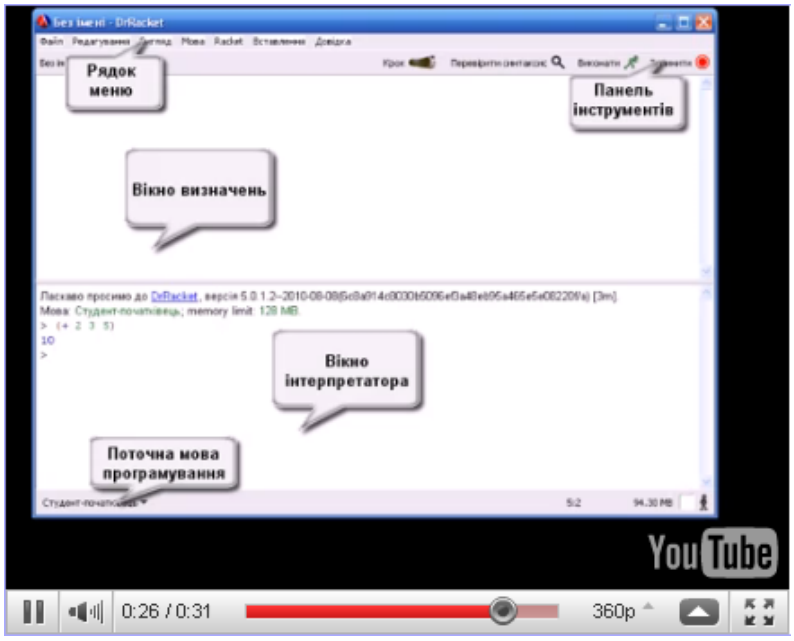


Рис. 2.11. Фрагмент відеододатку у форматі потокового відео «Короткий огляд DrRacket»

Дублювання відеоресурсів зумовлено тим, що для кожного з них можна зазначити як переваги, так і недоліки їх використання (відеофайл: переваги – після завантаження файлу доступ до його перегляду не регламентується доступом до

мережі та серверу дистанційного курсу, недоліки – необхідність добору програмних засобів для його адекватного перегляду; потокове відео: переваги – відсутність необхідності добору програмних засобів для перегляду, і в той же час недоліки – доступ до перегляду потокового відео регламентується доступом до мережі, серверу дистанційного курсу та серверу, на якому розміщено відеоресурси, у даному випадку – http://www.youtube.com/) [310, 154].

Після опрацювання лекційного матеріалу студент має можливість виконати лабораторну та індивідуальну роботи з відповідної теми та надіслати електронний варіант звіту, для даного випадку в НМК до кожної теми додано елемент «Завдання», адже в цьому випадку до журналу оцінок автоматично додається стовбець з оцінкою для цього виду роботи.

При створенні завдань було обрано параметр «обмеження терміну здачі» для кожної з робіт (оскільки за кредитно-модульною системою для будь-якого виду роботи передбачений термін її виконання та здачі), кожен студент може подавати файли неодноразово – залежно від результатів їх перевірки; це дає можливість вчасно коригувати роботу студента, наполегливо добиватися повного розв’язання навчальної задачі. Завдання оцінюються викладачем згідно встановлених критеріїв оцінювання (за вищою оцінкою).

Викладач може перевірити подані студентом файли, прокоментувати їх та, за необхідності, запропонувати їх доопрацювати. Якщо викладач вважає це за потрібне, він може відкрити посилання на файли, подані учасниками курсу, і зробити ці роботи предметом обговорення у форумі.

Під час виконання лабораторних та індивідуальних завдань у студентів формуються вміння розробки, опису, налагодження, тестування програм та аналізу результатів їх роботи (тобто, відбувається формування праксеологічної складової компетентностей з програмування), оскільки при виконанні завдань можуть виникнути труднощі, будь-хто зі студентів може на форумі курсу або в чаті задати питання та отримати оперативну допомогу (як з боку викладача, так і з боку студентів). Найактивніші учасники форуму, які надаватимуть цікаві та доступні відповіді на питання, отримують додаткові бали від викладача. Таким чином, під час ви-

конання лабораторних та індивідуальних робіт, відбувається внесок у формування соціально-поведінкової складової компетентностей з програмування.

Також для формування праксеологічної складової компетентностей з програмування у вступній частині курсу було розміщено web-сторінку з вбудованим в неї аплетом Scheme, за допомогою якого студенти можуть виконувати завдання безпосередньо на сторінці курсу (рис. 2.12).

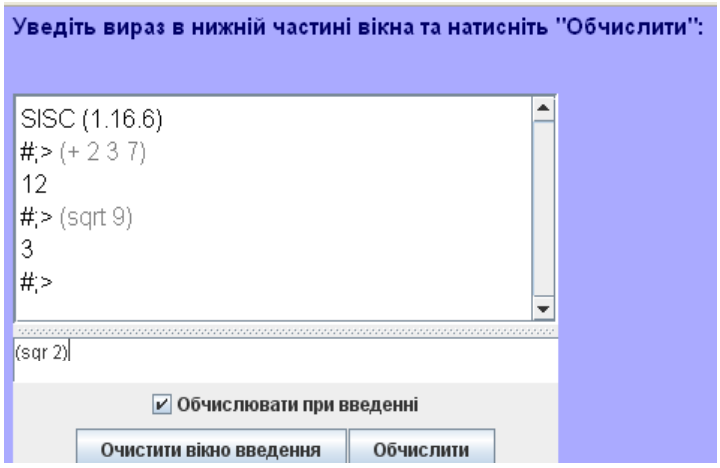


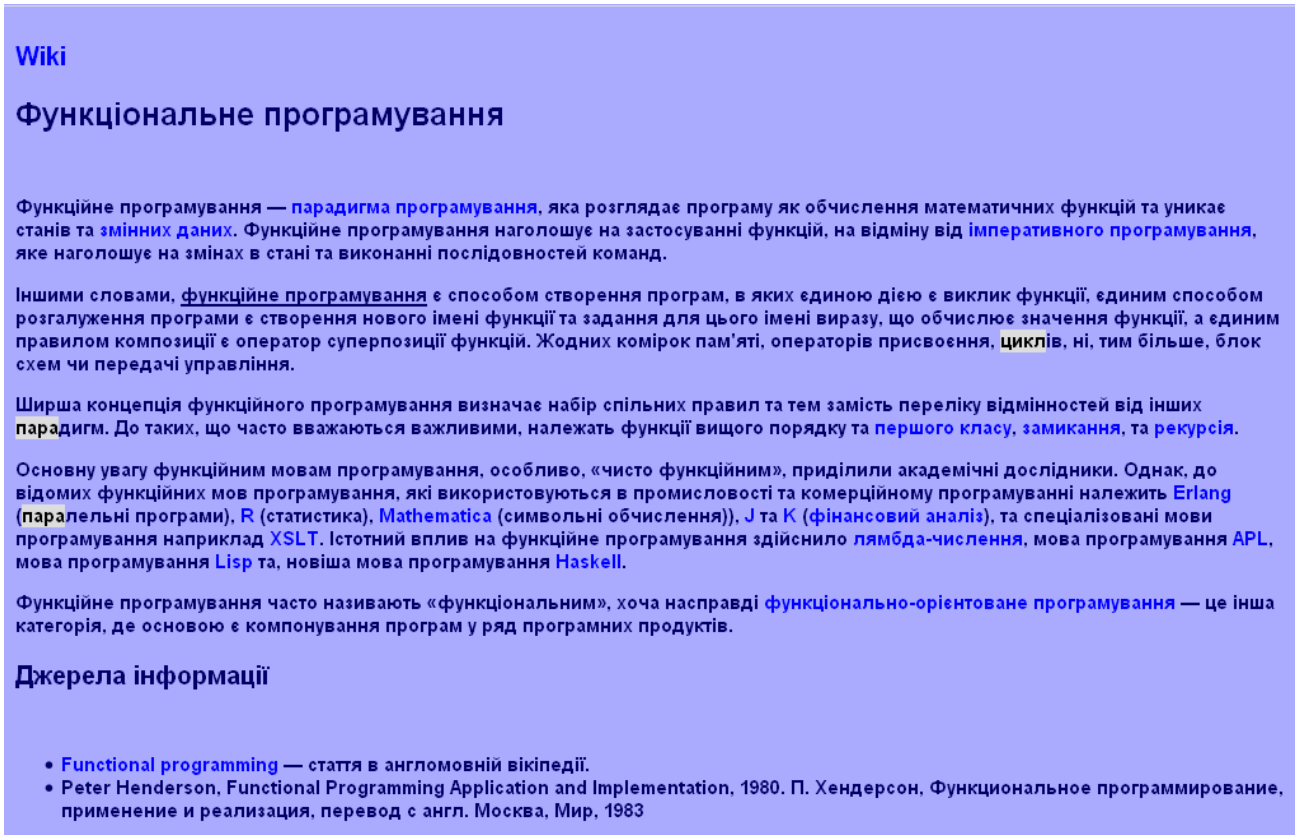
Рис. 2.12. Web-сторінка з вбудованим в неї аплетом Scheme

Для можливості використання аплету до курсу були завантажені файли цієї програми, а код програми був розміщений на сторінці з обчисленнями:

```
<span style="font-weight: bold;">Уведіть вираз в нижній частині вікна та натис-
ніть &quot;Обчислити&quot;:</span><br /><br /><br />
<applet width="400" height="532"
  code="sisc.contrib.applet.SISCApplet"
  archive="sisc-applet-uk.jar,sisc.jar,sisc-heap.jar"
  codebase="http://www.moodle.tryus.ii.npu.edu.ua/file.php/56/Scheme-applet/">
  <param name="CODE" value="sisc.contrib.applet.SISCApplet" />
  <param name="ARCHIVE" value="sisc-applet-uk.jar,sisc.jar,sisc-heap.jar" />
  <param name="type" value="application/x-java-applet;version=1.2" />
  <param name="scriptable" value="false" /></applet>
```

Для формування компетентностей з програмування, в курсі також було використано елемент Wiki, який дозволяє створювати гіпертекстовий документ, що містить навчальний матеріал окремого студента або групи студентів (рис. 2.13). Наявність цього ресурсу дає можливість створення навчального контенту: редагу-

вати елементи курсу, додавати, змінювати вміст та створювати засоби навчання, аналогів яким немає в традиційній формі навчання.



Wiki

Функціональне програмування

Функційне програмування — парадигма програмування, яка розглядає програму як обчислення математичних функцій та уникає станів та змінних даних. Функційне програмування наголошує на застосуванні функцій, на відміну від імперативного програмування, яке наголошує на змінах в стані та виконанні послідовностей команд.

Іншими словами, функційне програмування є способом створення програм, в яких єдиною дією є виклик функції, єдиним способом розгалуження програми є створення нового імені функції та задання для цього імені виразу, що обчислює значення функції, а єдиним правилом композиції є оператор суперпозиції функцій. Жодних комірок пам'яті, операторів присвоєння, циклів, ні, тим більше, блок схем чи передачі управління.

Ширша концепція функційного програмування визначає набір спільних правил та тем замість переліку відмінностей від інших парадигм. До таких, що часто вважаються важливими, належать функції вищого порядку та першого класу, замикання, та рекурсія.

Основну увагу функційним мовам програмування, особливо, «чисто функційним», приділили академічні дослідники. Однак, до відомих функційних мов програмування, які використовуються в промисловості та комерційному програмуванні належить Erlang (паралельні програми), R (статистика), Mathematica (символьні обчислення)), J та K (фінансовий аналіз), та спеціалізовані мови програмування наприклад XSLT. Істотний вплив на функційне програмування здійснило лямбда-числення, мова програмування APL, мова програмування Lisp та, новіша мова програмування Haskell.

Функційне програмування часто називають «функціональним», хоча насправді функціонально-орієнтоване програмування — це інша категорія, де основою є компонування програм у ряд програмних продуктів.

Джерела інформації

- Functional programming — стаття в англомовній вікіпедії.
- Peter Henderson, Functional Programming Application and Implementation, 1980. П. Хендерсон, Функциональное программирование, применение и реализация, перевод с англ. Москва, Мир, 1983

Рис. 2.13. Wiki

При виконанні групових проектів (модуль 3 та 4) організація спільної роботи між студентами відбувається за допомогою елементів «Чат», «Форум» та «Skype».

Також в курсі для оцінювання та контролю знань з кожної теми розроблено тестові завдання за допомогою елементу «Тест». Використання комп’ютерних тестів надає можливість:

– зробити процес оцінювання студентів об’єктивнішим, оскільки виключається можливість особистісного ставлення;

– економії навчального часу: одночасна перевірка знань всієї групи;

– формування мотивації та зацікавленості студентів до підготовки до занять, дисциплінування студентів.

Тести автоматично оцінюються і можуть бути переоцінені, якщо питання змінюються. Після виконання тесту, студенти отримують відгук та оцінку. При

створенні тестів виставлено обмеження в часі між двома спробами складання тесту та випадковий порядок питань.

Для формування ціннісно-мотиваційної та гносеологічної складових компетентностей з програмування у вступній частині курсу додано елемент курсу «Ігри», який розроблений на основі тестових завдань та глосарію.

На рис. 2.14 наведено приклад сторінки гри «Кросворд» (для питань кросворду використовується словник глосарію).

Також у курсі розроблено ігри «Мільйонер», «Шибениця», «Змії та сходи» і «Криптекст».

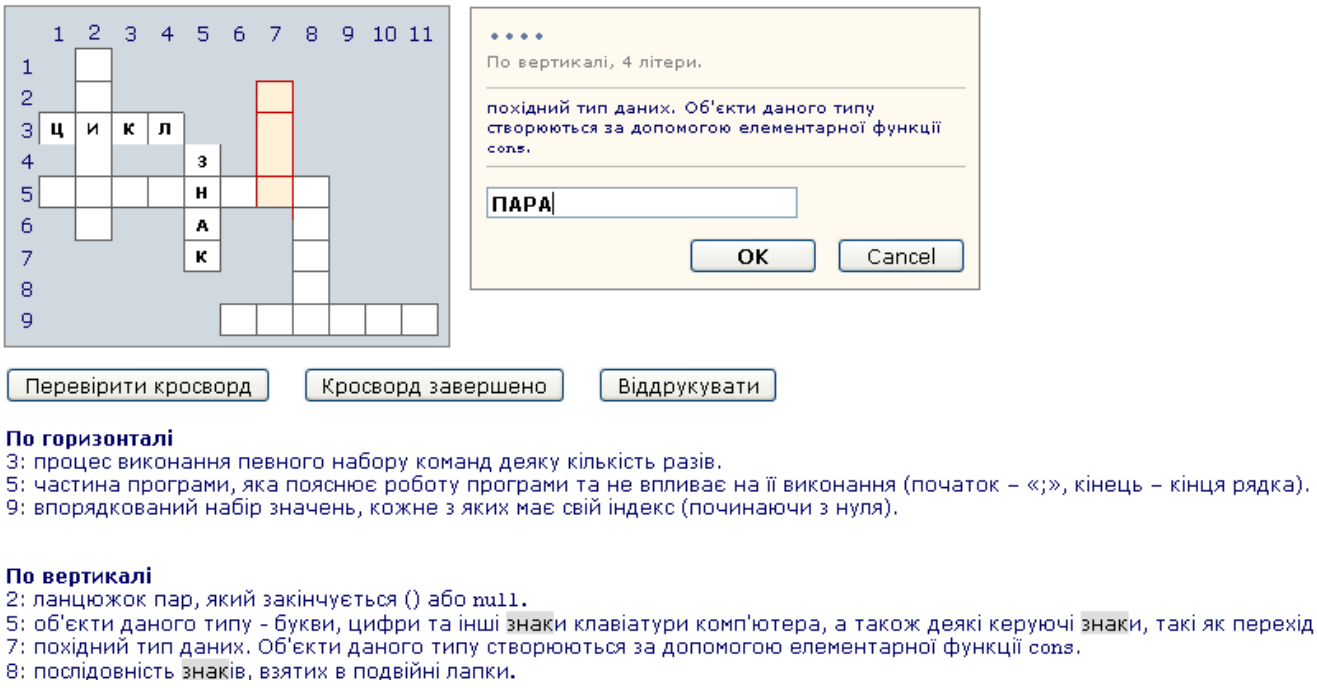


Рис. 2.14. Гра «Кросворд»

ЕСПН Moodle надає можливість переглянути як всіх зареєстрованих учасників курсу – сторінка «Учасники», так і тих учасників, хто знаходиться на сайті курсу в даний час (для цього до курсу додано блок «Online Users (mobile)»). Обравши сторінку зарахованого на курс студента, викладач може довідатись про діяльність студента на курсі. У «Звіти про діяльність» фіксуються всі дії студента на сторінках курсу.

В ЕСПН Moodle є «електронний журнал». Викладач має можливість переглядати оцінки як всіх учасників курсу, так і обраного користувача, відсортувати

учасників курсу за результатами того чи іншого виду діяльності.

Dropbox – безкоштовний сервіс, що надає можливість об'єднати всі файли (фотографії, відео, документи) користувача в будь-якому місці. Будь-який файл, збережений в Dropbox, буде автоматично збережений на всі комп'ютери та мобільні пристрої користувача і навіть на сайті Dropbox. Отож, тепер можливо розпочати роботу на комп'ютері у навчальному закладі і закінчити на домашньому комп'ютері. Необхідність резервного копіювання та синхронізації файлів зникла.

Детальний огляд роботи з web-сервісом Dropbox наведено в Додатку Б.

Враховуючи всі можливості Dropbox для курсу «Вступ до програмування» було розроблено папку в Dropbox з навчальними матеріалами (рис. 2.15). Оскільки у створеному в ЕСПН Moodle НМК «Вступ до програмування» розміщено всі матеріали даного курсу (як основні, так і допоміжні), використання Dropbox зумовлене насамперед можливістю спільного використання документів та синхронізації файлів.

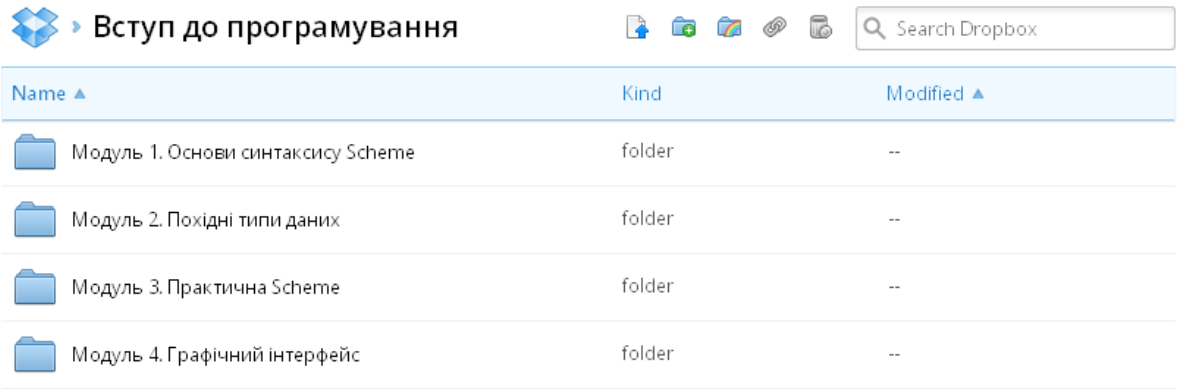


Рис. 2.15. Папка «Вступ до програмування» в Dropbox

У даному курсі необхідна організація спільної роботи над проектними завданнями (модуль 3 та модуль 4), але для того, щоб студенти повноцінно використовували всі можливості Dropbox, доцільним є більш раннє ознайомлення з принципами роботи з даним web-сервісом. Тому в Dropbox було створено папку курсу, що містить лабораторні й індивідуальні роботи 1 та 2 модулів, а також проекти 3 та 4 модулів. Оскільки Dropbox надає можливість працювати з усіма типами файлів для спільного доступу, окрім текстових файлів, було розміщено і робо-

чі файли проектів. Після створення папки курсу в Dropbox до спільного доступу було запрошено усіх учасників даного курсу.

Таким чином, Dropbox є засобом швидкого реагування – при зміні файлу на комп'ютері викладача або будь-кого зі студентів, відбувається автоматичне оновлення цього ж файлу в усіх учасників навчального процесу. Окрім того, при розробці індивідуальних завдань у третьому та четвертому модулях, студенти самі створюють папку із робочими матеріалами та запрошують до співпраці лише тих, хто співпрацює з ними у даному проекті.

Web-сервіс Dropbox є засобом технології Web 2.0, яка реалізує теорію соціального конструктивізму.

Skype. Ще одним засобом організації спільної роботи є система Skype, використання якої надає можливість передавати будь-які види повідомлень (голосові, текстові, графічні тощо). В основі цієї системи лежить технологія VoIP (Voice over Internet Protocol – передача голосу через Інтернет-протокол).

Серед переваг Skype порівняно з аналогічними програмами можна виокремити: 1) широкий спектр надаваних користувачу послуг (організація аудіо та відео конференцій, обмін файлами, передавання текстових повідомлень в онлайн режимі, автовідповідач, запис дзвінків); 2) багатомовний інтерфейс; 3) інтуїтивно зрозумілий інтерфейс; 4) висока якість передавання аудіоданих навіть на лініях з невеликою пропускною здатністю; 5) безкоштовність.

Особливе значення у web-орієнтованому навчанні має ще одна послуга Skype – можливість отримання доступу до робочих столів співрозмовників [156, 71] (рис. 2.16).

Подкастинг (podcasting – від iPod та broadcasting – повсюдне, широкоформатне мовлення) – процес створення і поширення звукових або відео-передач (подкастів) у Internet (зазвичай в форматі MP3, AAC або Ogg / Vorbis для звукових і Flash Video та інших для відео-передач). Як правило, подкасти мають певну тематику і періодичність видання, однак є й винятки [17; 252, 134].

Цільова аудиторія подкастингу – користувачі персональних або портативних комп'ютерів, а також власники портативних програвачів. Для зручного про-

слуховування подкастів створено багато програмних продуктів, таких як iTunes та AmaroK, що відслідковують оновлення подкаст-стрічок та автоматично завантажують новий матеріал.

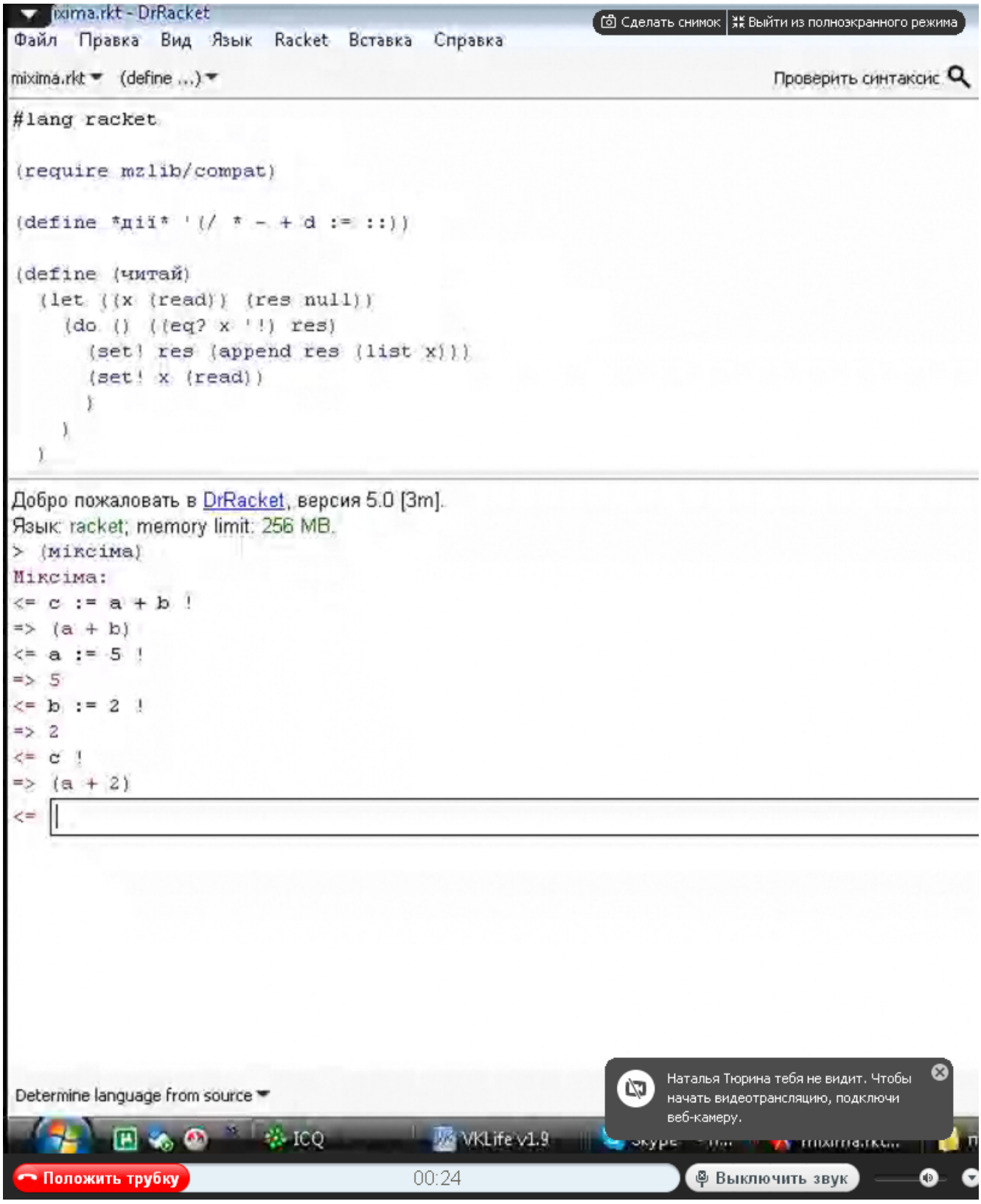


Рис. 2.16. Доступ до робочого столу співрозмовника засобами Skype

Подкаст-термінал – це web-сайт, що підтримує хостинг медіа-файлів та певною мірою автоматизує розміщення записів на сайті та підписку на оновлення. *Подкастом* називається або окремий файл, або регулярно оновлювана серія таких файлів, що публікуються за однією адресою в мережі. Поняттю подкастинга відповідає поняття аудіоблогу: під блогом зазвичай розуміють послідовність записів у вигляді звичайних web-сторінок, а подкаст завжди забезпечує автоматичну перевірку оновлень за допомогою формату RSS.

Таким чином, структура КОЗН для формування у студентів педагогічних університетів компетентностей з програмування на основі ФП включає середовища програмування, ЕСПН та середовища організації спільної роботи (рис. 2.17).

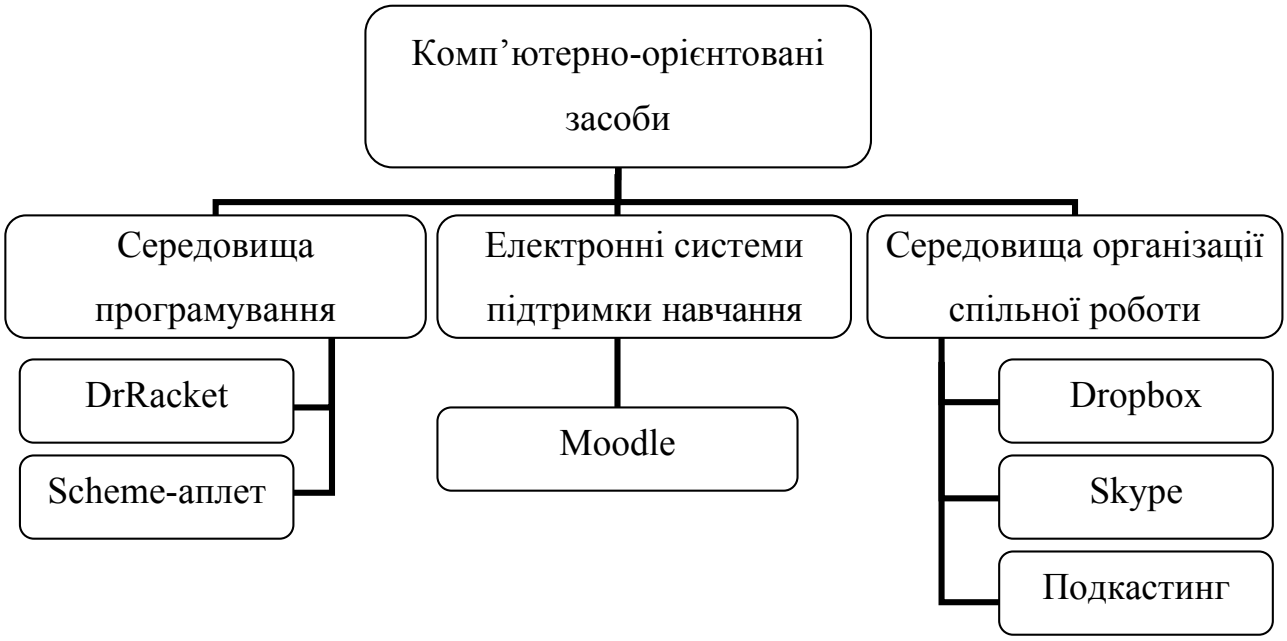


Рис. 2.17. КОЗН для формування
у студентів педагогічних університетів компетентностей з програмування
на основі функціонального підходу

Всі використовувані засоби індиферентні до операційної системи та архітектури комп'ютера – це мобільне, локалізоване програмне забезпечення.

2.2.3 Форми організації навчання для формування у студентів напряму підготовки «Інформатика*» педагогічних університетів компетентностей з програмування. *Форма організації навчання* – це обмежена в просторі та часі взаємозумовлена діяльність викладача й студента [139, 247].

Враховуючи особливості комунікативної взаємодії між викладачем і студентами та між самими студентами, серед загальних форм організації навчання розглядають фронтальні (колективні), парні, індивідуальні та зі змінним складом студентів [252, 200].

Під час *фронтального* навчання всі студенти, присутні на занятті, працюють над одним і тим самим завданням в єдиному темпі. У курсі «Вступ до програмування» на початку вивчення тем першого та другого модулів на лабораторних заняттях здебільшого використовується саме ця форма організації навчання, оскільки рівень компетентностей з програмування у студентів є переважно низьким (відбувається формування гносеологічної та праксеологічної складових).

Оскільки студенти, що мають достатні та високі знання і навички з програмування, мають можливість виконати завдання лабораторних робіт заздалегідь (завдяки використанню КОЗН, а саме ЕСПН Moodle), на лабораторних заняттях викладач може залучити їх до навчання слабших студентів. Реалізація цієї форми знайшла своє відображення в методі *учіння через навчання* [252, 202], що сприяє формуванню ще й соціально-поведінкової складової компетентностей з програмування.

При навчанні програмуванню ефективною є *парна* форма організації навчання, за якої основна взаємодія відбувається між двома студентами, що працюють за одним робочим місцем, котрі можуть обговорювати завдання, здійснювати взаємонавчання або взаємоконтроль. Парне програмування полягає у тому, що один студент працює над написанням коду, а інший сидить поряд, і спостерігає за його роботою, таким чином контролюючи його роботу, і уявляє проект в цілому. За домовленістю, вони міняються місцями [252, 201]. Ця форма навчання є переважною на лабораторних заняттях, як в аудиторній роботі, так і позааудиторній (наприклад, в гуртожитку).

Індивідуальна форма навчання передбачає виконання кожним студентом власного завдання у власному темпі під керівництвом викладача.

Особливістю лабораторних занять в комп’ютерній аудиторії є індивідуальний темп роботи кожного студента, навіть за спочатку фронтальної форми органі-

зації заняття. В умовах комп'ютерного класу керувати індивідуальною діяльністю студентів досить складно: ситуація за кожним комп'ютером практично унікальна. В цьому випадку знову доцільним буде використання методу учіння через навчання.

Ефективній організації індивідуальної роботи сприяє використання всіх можливостей КОЗН:

– елементів ЕСПН Moodle: індивідуальні завдання, тестові завдання, чат;

– системи Skype: чат, доступ до робочого столу, обмін файлами.

Під час *роботи над проектами* (третій та четвертий модуль розробленого курсу) використовується *групова* форма організації навчання: студенти працюють у групах. При об'єднанні студентів у групи варто враховувати бажання студентів, але керівництво викладача є необхідним. Групова форма роботи є ефективною, якщо групи є неоднорідними як за рівнем предметних компетентностей, так і за пізнавальною активністю.

Використання КОЗН сприяє не лише індивідуалізації навчання, а й оптимізації групової роботи (за рахунок засобів для організації спільної роботи).

Навчальний процес у ВНЗ здійснюється у таких формах: навчальні заняття; самостійна робота; практична підготовка; контрольні заходи [276, 235].

Зовнішні форми організації навчання інформатики позначають певний вид заняття: лекція, семінар, практичне заняття, лабораторне заняття, практикум, факультативне заняття, екзамен, предметні гуртки, студентські наукові співтовариства й т.д.

У навчанні інформатики найпоширенішою є лекційно-лабораторна форма [252, 201]. Згідно з робочою програмою, на вивчення курсу «Вступ до програмування» передбачається 18 год. лекцій та 36 год. лабораторних занять. Також в курсі передбачається проведення консультацій, самостійна робота студентів.

Лекція – усне систематичне та послідовне подання матеріалу з певної проблеми, методу, теми, питання й т.д. є провідною формою і методом навчання у ВНЗ [252, 203]. Проведення лекцій з інформатики вимагає від лектора високого рівня методичних, психолого-педагогічних та інформатичних компетентностей.

Незважаючи на критику за пасивність навчання, лекція є необхідною формою організації навчання зі студентами молодших курсів.

Підвищенню ефективності лекції сприяють використання наочних методів, проведення комп'ютерних презентацій, організація зворотнього зв'язку з аудиторію. Завданням лектора є активізація мислительної діяльності та розвиток уваги студентів, організація на занятті атмосфери співтворчості та емоційної взаємодії [125, 107].

Керівництво роботою студентів на лекції полягає у визначенні вимог щодо загальної культури поведінки на занятті (запитання до лектора, їх форма, час); навчання студентів конспектуванню; допомога в здійсненні записів через зміни темпу подання навчальних матеріалів, його повтор, застосування пауз тощо; перегляд конспектів протягом лекції, на семінарах і практичних заняттях; використання прийомів підтримки уваги студентів (запитання, риторичні запитання, ораторські прийоми) [292, 134].

Перевагою лекції є її економічність: викладач подає матеріал за порівняно незначний час.

У зв'язку з широким впровадженням КОЗН (зокрема, ЕСПН Moodle, в якій у вигляді web-сторінок розроблено лекції до курсу «Вступ до програмування») функції лекції з інформаційної (виклад необхідних відомостей) зміщуються в бік інших: стимулюючої (пробудження інтересу до теми), виховної, розвивальної (оцінювання явищ, розвиток психічних функцій – уваги, мислення), орієнтовної (в проблемі, в літературі), пояснювальної, переконувальної (з акцентом на систему доведень) [292, 133].

Якщо студенти ознайомлені з матеріалом лекцій та мають їх у роздрукованому вигляді, можливо не вимагати від студентів детального конспектування. В цьому випадку оптимальною є форма конспекту, в якій з одного боку тезами подаються основні моменти, а з іншого – коментарі. Або ж можливо провести лекцію – прес-конференцію (якщо рівень підготовки студентів високий): на початку лекції впродовж декількох хвилин лектор отримує запитання від студентів в усній або письмовій формі. Викладач, розглянувши і відсортувавши запитання, починає

лекцію, розглядаючи лише ті питання, що викликали труднощі у студентів. Це дає змогу інтенсифікувати процес навчання, залучити студентів до активної діяльності щодо оволодіння змістом курсу, процесу здобуття нових знань, ознайомити з новою для студентів формою організації навчальних занять [246, 145].

За цілями проведення виділяють вступну, інформаційну та оглядову лекції.

Завданням вступної лекції курсу «Вступ до програмування» є знайомство студентів з метою та призначенням курсу, його роллю та місцем в системі учбових дисциплін. На лекції розглядаються і організаційні питання: загальна методика роботи над курсом (обсяг годин, форми організації навчання, самостійна робота); форми контролю, критерії оцінювання, навчально-методичний матеріал (підручники, електронні ресурси, освітні ресурси web). Також дається короткий історичний огляд курсу (етапи розвитку, імена відомих вчених) та перспективи розвитку науки і її вкладу в практику. У вступній лекції важливо показати, яку роль в подальшому відіграватиме теоретичний матеріал з даного курсу, як у навчальній, так і професійній діяльності. Вступна лекція сприяє професійній адаптації студентів молодших курсів.

Інформаційна лекція розкриває зміст теми у відповідності з навчальною програмою.

Оглядова лекція є підсумковою у вивченні теми або ж курсу в цілому і сприяє систематизації знань. В курсі «Вступ до програмування» оглядовою є остання лекція. На ній варто приділити увагу питанням, що викликають особливі складнощі у студентів, оголосити кількість балів, набраних кожним студентом протягом вивчення курсу.

Також лекції можна класифікувати за формами.

Проблемна лекція передбачає виклад інформації за допомогою створення проблемної ситуації, яка спонукає студентів до пошуку і вирішення певних суперечностей, а отже, набуття нових знань.

Лекція-візуалізація подає інформацію у візуальній формі. Демонстраційні матеріали не просто доповнюють словесну інформацію, а й є носіями змістовної інформації. Викладання такої лекції зводиться до зведеного, розгорнутого комен-

тування підготовлених візуальних матеріалів, які повинні:

– забезпечити систематизацію наявних знань;

– забезпечити засвоєння нової інформації;

– забезпечити створення та розв’язання проблемних ситуацій;

– демонструвати різні способи візуалізації.

У лекцію із запланованими помилками викладач спеціально закладає певні помилки (змістові, методичні, поведінкові). Завдання студентів – розпізнати, зафіксувати, а наприкінці лекції – назвати ці помилки. Методичні засоби, що застосовуються на таких лекціях, дають змогу студентам не тільки сприймати і осмислювати інформацію, а й оволодівати способами пошуку нових знань. Вони є доцільними, коли з’ясовуються дискусійні наукові проблеми, повідомляються і узагальнюються нові наукові дослідження [292, 136].

Лабораторна робота (фронтальна) є основною формою роботи в комп’ютерному класі. Діяльність студентів може бути як синхронна, так і асинхронна. Досить часто відбувається швидке «розтікання» фронтальної діяльності навіть при спільному вихідному завданні. Роль викладача під час фронтальної лабораторної роботи – спостереження за роботою студентів (у тому числі через мережу), надання їм оперативної допомоги [157, 58].

В розробленому курсі студенти мають можливість заздалегідь ознайомитися з завданнями лабораторних робіт (в НМК «Вступ до програмування» – елемент «лабораторна робота»).

Додаткові (*консультаційні*) форми організації навчання розраховані на окремих студентів або групу з метою заповнення пробілів у знаннях, вироблення вмінь і навичок, задоволення підвищеного інтересу до навчального предмету [216]. Так, на консультаціях можуть бути роз’яснені окремі питання, організоване повторне пояснення теми і т.п.

Для задоволення пізнавального інтересу та поглибленого вивчення предмета з окремими студентами проводяться заняття, на яких розв’язуються завдання підвищеної складності, обговорюються наукові проблеми, що виходять за рамки програми, даються рекомендації із самостійного опанування проблем, що цікав-

лять студентів.

Розрізняють поточні, тематичні й узагальнюючі (наприклад, при підготовці до екзаменів або заліків) консультації. Консультації найчастіше є груповими (від 5 студентів), що однак не виключає й індивідуальних консультацій. Завдяки використанню КОЗН можливе проведення консультацій «на відстані» – чат, форум або ж система Skype (особливо цінною у даному випадку є послуга доступу до робочих столів співрозмовників).

Комп'терно-орієнтований лабораторно-обчислювальний практикум (за типом «занурення») – форма, за якою передбачається інтенсивна концентрована робота студентів у комп'ютерному класі з відривом від інших занять протягом 1–2 тижнів. У ході занурення може бути опрацьований матеріал з окремого курсу або сукупності тем. Така форма була використана в курсі «Вступ до програмування» зі студентами першого курсу спеціальності «Інформатика» КДПУ (КНУ) у 2009-2010 н.р. Оскільки рівень і знань, і пізнавальної активності студентів був дуже низьким, протягом одного тижня (36 аудиторних год.), під час занурення студентами було засвоєно перший та другий модулі курсу.

Комп'ютерно-орієнтовані семінари та *практичні заняття* є перехідною формою від фронтальної до індивідуальної роботи. В навчанні програмування необхідно виробляти ряд немашинних та домашинних навичок і вмінь (наприклад, розробка та обговорення алгоритму, моделі тощо). Практичне заняття – найбільш адекватна форма роботи для колективного осмислення того, що треба зробити або вже зроблено на комп'ютері, і чому такі результати отримані.

Важливим інтелектуальним умінням є здатність до розгорнутого прогнозу результатів, отриманих за допомогою комп'ютера на основі накопиченого досвіду роботи з ним. Для його формування доцільно застосовувати семінарські заняття.

Студентам корисно знати, що саме зараховується як результат роботи на семінарі. На семінарах з програмування можливі контрольовані результати:

– текст алгоритму, готовий для введення;

– таблиця виконання алгоритму, складена без застосування комп'ютера;

– проект роботи із програмою;

– відповіді на питання інструкції;

– інструкція до власної або чужої програми;

– коментарі до своєї або чужої програми;

– опис очікуваних результатів роботи з програмою [157, 59].

Проектна форма навчання. В основі проектної форми лежить творча діяльність студента. Ознаками проектної форми навчання є:

– наявність організаційного етапу підготовки до проекту – самостійний вибір і розробка варіанту виконання, вибір програмних і технічних засобів, вибір джерел потрібних відомостей;

– вибір із числа учасників проекту лідера (організатора, координатора), розподіл ролей;

– наявність етапу самоекспертизи й самооцінки (рефлексії), захисту результату та оцінювання рівня виконання;

– кожна група може займатися розробкою окремого проекту або брати участь у втіленні колективного проекту [157, 60].

Самостійна робота – це спланована викладачем робота студентів, що виконується за його завданням і під його керівництвом, але без особистої участі викладача.

В організації самостійної роботи студентів можна виокремити декілька стадій:

1. Планування (здійснюється при розробці робочої програми). При визначенні змісту та форм організації самостійної роботи слід врахувати вимоги: відповідність меті і завданням курсу, доступність, конкретність та чіткість постановки завдань, систематичність, контроль, однозначність оцінювання.

Плануючи самостійну роботу студента, викладач зобов’язаний створити відповідні умови для її виконання. Для цього потрібен підвищений рівень мотивації виконання тієї чи іншої роботи, чітке визначення значення кожної в майбутній навчальній або професійній діяльності, адже студенти засвоюють лише те, чого хочуть навчитися [125, 24].

2. Виконання. В разі необхідності можливе проведення консультацій (як ін-

дивідуальних, так і групових). Завдяки КОЗН можливе проведення «заочних» консультацій – використовуючи можливості ЕСПН Moodle: чат, форум; системи Skype: чат, надсилання файлів, аудіо та відео конференції. В навчанні програмування особливо цінною є послуга системи Skype з надання доступу до робочого столу співрозмовника, що інтенсифікує процес надання допомоги студентам з розробки та налагодження програми.

3. Контроль та корекція. Перевірка виконання самостійної роботи здійснює визначальний вплив на її якість. Ефективність самостійної роботи студентів залежить не лише від вимог, що висуваються до виконання самостійної роботи, але й від прийомів її перевірки, які повинні бути різноманітними, як за формою, так і за змістом. Головними вимогами до контролю за самостійною роботою є систематичність, своєчасність, об’єктивність, оперативність, дієвість, економічність. Можливості ЕСПН Moodle відповідають всім зазначеним вимогам: завдяки електронному журналу викладач має можливість отримати відомості про всіх зареєстрованих учасників курсу: вид роботи, кількість часу, витраченого на той чи інший вид роботи, отримані результати.

Варто зауважити, що своєчасність, систематичність та якість виконання самостійної роботи є одним із показників сформованості ціннісно-мотиваційної складової компетентностей. Адже вона свідчить про формування таких якостей як наполегливість, уміння ставити цілі та знаходити шляхи їх досягнення; в разі необхідності – вносити зміни та діяти відповідно до зовнішніх обставин; прагнення досягнути успіху.

В курсі «Вступ до програмування» передбачаються такі види самостійної роботи: підготовка до лекцій (у тому числі виступів на них), лабораторних робіт, заліку, виконання рефератів, виконання індивідуальних завдань та проектів.

Самостійна робота з даного курсу сприяє формуванню не лише компетентностей з програмування, а й навчальній компетентності: засвоєнню прийомів процесу пізнання; розвитку пізнавальних здібностей; умінь та навичок самостійного дослідження проблем; здатності до самостійного здобуття знань, тобто здатності до самонавчання; уміння користуватися джерелами і засобами різноманітних да-

них, повідомлень, постійно підвищувати рівень своєї освіти.

Використання КОЗН надає можливість ефективної реалізації принципу неперервності навчання і відкритості змісту навчання, а також дає змогу реалізовувати самостійну навчальну діяльність студентів.

Перевагами організації самостійної роботи з використанням КОЗН є:

– ефективність контролю;

– забезпечення зворотного зв’язку;

– висока активність взаємодії між студентами і викладачем, між студентами;

– індивідуалізація та диференціація навчання;

– можливість використання колективної форми самостійної роботи;

– створення атмосфери суперництва;

– комфортніша атмосфера самостійної роботи (встановлення гуманістичного стилю педагогічного спілкування між студентами і викладачем);

– забезпечення кожного студента необхідними матеріалами для самостійного навчання (лекційний матеріал, наявність глосарія, корисні посилання, приклади виконання завдань, онлайнові консультації викладача, ресурси мережі Інтернет тощо) [286, 75].

Самостійна навчальна діяльність студентів у розробленому НМК «Вступ до програмування» здійснюється через:

– опрацювання теоретичних відомостей з самоконтролем засвоєння навчального матеріалу;

– дослідницьку і пошукову діяльність (шляхом виконання індивідуальних завдань та проектів);

– участь в іграх (навчальних, розвиваючих тощо);

– самотестування і тестування;

– залучення студентів до створення різноманітних елементів (глосаріїв, wiki, ігор та ін.).

2.3 Методика формування у студентів напряму підготовки «Інформатика*» педагогічних університетів компетентностей з програмування на основі функціонального підходу на прикладі окремих тем

Формування компетентностей з програмування, як зазначалось вище, носить акумулюючий характер – внесок у формування всіх складових компетентностей під час вивчення різних тем є неоднаковим: так, наприклад, під час проходження студентами перших двох модулів курсу «Вступ до програмування» здійснюється більший внесок у формування гносеологічної та праксеологічної складових (через досить значний об'єм нових знань (синтаксис мови програмування, визначення функцій та змінних, типи даних, функції для роботи з ними і т.д.), формування аксіологічної складової досягається шляхом використання методу доцільно дібраних задач. Формування соціально-комунікативної складової компетентностей з програмування відбувається у зв'язку з необхідністю подання виконаних індивідуальних робіт викладачу за допомогою web-сервісу Dropbox, через використання системи Skype та використання методу парного програмування – таким чином, існує необхідність постійного зв'язку зі своїм «напарником».

Під час проходження третього та четвертого модулів («Практична Scheme» та «Графічний інтерфейс») здійснюється більший внесок у формування праксіологічної складової (через необхідність застосування вже набутих знань до розв'язування практично-значущих задач), аксіологічної та соціально-комунікативної (через використання методу проектів).

Розглянемо детальніше ці особливості на прикладі навчання конкретних тем курсу «Вступ до програмування».

2.3.1 Умовні вирази в мові програмування Scheme. На початку цієї теми варто зауважити, що вона є складовою розділу «Керування виконанням програми». Вивчення теми передбачає такі етапи:

- опрацювання студентами теоретичного матеріалу;
- виконання і захист лабораторної роботи;
- виконання і захист індивідуальних завдань;
- проходження тестування.

На першому занятті студенти ознайомлюються з системою оцінювання та термінами виконання і здачі того чи іншого виду роботи. Для цього в НМК створено ресурс «Робоча програма».

Розглянемо всі виокремлені етапи.

- Для викладення теоретичного матеріалу використовуємо лекційний метод, а для підтримки самостійної роботи над цим матеріалом – розроблений автором посібник [191] або лекції в НМК (див. рис. 2.8).

Теоретичний матеріал

При виконанні програми може знадобитись здійснення певних перевірок та, в залежності від їх результату, виконання тих чи інших дій або ж виконання певних дій необхідну кількість разів чи доти, доки виконуються певні умови. В таких випадках використовують вирази, що містять перевірку умови (надалі будемо вживати скорочення «умовні вирази») або вирази, що містять повторення виконання певних дій («циклічні вирази»).

Умовні вирази

Розглянемо формулу для знаходження модуля числа:

$$|x| = \begin{cases} x & якщо \quad x \geq 0 \\ -x & якщо \quad x < 0 \end{cases}.$$

Умовний вираз if

Для розв'язування цієї задачі можна скористатись елементарним умовним виразом `if` (`if` є елементарним умовним виразом, оскільки з його допомогою реалізовуються всі інші умовні вирази). Загальна форма:

```
(if предикат вираз-так вираз-інакше)
```

Якщо `предикат` приймає значення «істина», значенням умовного виразу буде значення `виразу-так`, якщо «хиба» – `виразу-інакше`.

Функція обчислення модуля числа з використанням умовного виразу `if`:

```
> (define (модуль x)
    (if (>= x 0)
        x
        (- x)))
> (модуль -5)
```

```
5
```

Розглянемо формулу для знаходження значень signum-функції (функції знаку) числа:

$$signx = \begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0 \end{cases}.$$

В цьому випадку для розв'язування задачі необхідно скористатись вкладеними умовними виразами:

```
> (define (знак x)
    (if (> x 0)
        1
        (if (= x 0)
            0
            -1)))
> (знак 3)
1
```

Умовний вираз `cond`

Якщо у задачі потрібно перевірити три або більше умови, можна скористатися похідними умовними виразами `cond` або `case` (ці вирази називаються похідними, оскільки їх можна виразити, використовуючи елементарний умовний вираз `if`, тобто вони «походять» від нього). Скорочена форма умовного виразу `cond`:

```
(cond (предикат1 тіло1)
      (предикат2 тіло2)
      . . .
      (предикатn тілоn))
```

Тіло може мати вигляд `вираз1 вираз2 … вирazk`.

Форма складається з виразу `cond`, після якого записані взяті в дужки пари виразів `(предикат тіло)`, які називаються гілками. Значення умовного виразу `cond` обчислюється так: спочатку здійснюється перевірка значення `предикату1`, якщо він приймає значення «істина», результатом всього `cond`-виразу є значення останнього виразу в `тілі1`, інакше – здійснюється перевірка значення `предикату2`. Так продовжується доти, доки один із предикатів не прийме значення «істина». В цьому ви-

падку інтерпретатор повертає значення останнього виразу у відповідному тілі в якості значення всього виразу cond. Якщо жоден з предикатів не прийме значення «істина», значення умовного виразу буде невизначеним.

З використанням cond функція обчислення модуля числа матиме вигляд:

```
> (define (модуль x)
    (cond ((>= x 0) x)
          ((< x 0) (- x))))
> (модуль -5)
5
```

Повна форма умовного виразу cond:

```
(cond (предикат1 тіло1)
      (предикат2 тіло2)
      ...
      (предикатn тілоn))
      (else тіло))
```

else – службове слово в заключній гілці cond. Якщо жоден з предикатів не прийме значення «істина», в якості результату буде повернуто значення останнього виразу саме з цієї гілки.

Використовуючи повну форму умовного виразу cond, функція обчислення знаку числа матиме вигляд:

```
> (define (знак x)
    (cond ((>= x 0) x)
          (else (- x))))
> (знак 0)
0
```

Умовний вираз case

case-вираз, як і cond, є похідним умовним виразом. Він має дві форми:

– скорочену (case ключ гілка1 гілка2 ... гілкаn);

– повну: (case ключ гілка1 гілка2 ... гілкаn (else тіло))

Ключ може бути будь-яким виразом, а гілки повинні мати вигляд (список_значень тіло), де тіло може мати вигляд вираз1 вираз2 … виразk.

Значення case-виразу обчислюється таким чином: значення ключа порівнюється зі списком_значень гілки1, якщо значення ключа не дорівнює жодному із

значень, значення ключа порівнюється зі списком_значень гілки2 і т. д., доки значення не співпадуть, в цьому випадку значенням `case`-виразу буде значення відповідного тіла. Якщо значення ключа не співпало з жодним значенням, результатом буде значення тіла `else`.

Приклад 1. Визначити функцію, яка отримує номер дня тижня та повертає його назву.

```
> (define (день-тижня x)
    (case x
      ((1) "понеділок")
      ((2) "вівторок")
      ((3) "середа")
      ((4) "четвер")
      ((5) "пятниця")
      ((6) "субота")
      ((7) "неділя")
      (else "невідомо")))
> (день-тижня 13)
невідомо
```

Приклад 2. У деякого працівника повні робочі дні – понеділок, вівторок, четвер; середа і п’ятниця – неповні. Субота і неділя – вихідні. Визначити функцію, яка за номером дня тижня, повертає його «робочу характеристику».

```
> (define (робочий-день x)
    (case x
      ((1 2 4) "повний")
      ((3 5) "неповний")
      ((6 7) "вихідний")
      (else "невідомо")))
> (робочий-день 6)
вихідний
```

Умовні вирази `when` *і* `unless`

Похідні умовні вирази `when` і `unless` зручно використовувати тоді, коли є тільки одна гілка в умовному виразі («коли» або «інакше»). Загальна форма умовного виразу `when`:

```
(when предикат тіло)
```

Тіло може мати вигляд `вираз1 вираз2 … виразk`.

Якщо предикат набуває значення «істина», то обчислюються всі вирази. Значенням всього `when`-виразу буде значення останнього виразу, обчисленого в його тілі.

Загальна форма умовного виразу `unless`:

```
(unless предикат тіло)
```

Тіло може мати вигляд `вираз1 вираз2 … виразk`.

Якщо предикат набуває значення «хиба», то обчислюються всі вирази. Значенням всього `unless`-виразу буде значення останнього виразу, обчисленого в його тілі.

Вирази логічної композиції

Крім символів елементарних співвідношень існують вирази логічної композиції, які надають можливість утворювати складені умови:

– **(and вираз1 вираз2 … виразn)**

Інтерпретатор обчислює значення виразів по одному, зліва направо. Якщо якийсь із виразів набуває значення «хиба», то значення всього `and`-виразу – «хиба», інші вирази навіть не обчислюються. Якщо всі вирази набувають значення «істина», то значенням виразу `and` є значення останнього з них.

```
> (define a 3)
> (and (< 2 5) (number? a))
#t
> (define a 3)
> (define b 5)
> (and (= a 3) (< b a) (and (number? a) (number? b)))
#f
```

– **(or вираз1 вираз2 … виразn)**

Інтерпретатор обчислює значення виразів по одному, зліва направо. Якщо якийсь із виразів набуває значення «істина», то його значення повертається як результат виразу `or`, а інші вирази не обчислюються. Якщо всі вирази набувають значення «хиба», то значенням виразу `or` є хиба.

```
> (define a 3)
> (or (< 2 5) (number? a))
```

#t

– **(not вираз)**

Якщо вираз набуває значення «хиба», то значенням виразу `not` є «істина», і «хиба» – в протилежному випадку.

```
> (define a 5)
> (not (< a 10))
#f
```

- Завдання лабораторної роботи:

1. Проаналізуйте, коли найраціональніше використовувати умовний вираз if, cond, case, when, unless. Наведіть приклади задач та їх розв'язки, використовуючи всі можливі для даного випадку умовні вирази.

В наступних завданнях (2-8) обов'язково перевірте правильність вхідних даних та продемонструйте роботу функцій для всіх можливих випадків.

2. Визначте функції:

– (max x y), яка повертає більше з двох чисел;

– (min x y), яка повертає менше з двох чисел;

– $z = \begin{cases} x - y, & якщо \quad x > y \\ y - x + 1 & інакше \end{cases}$.

Протестуйте їх роботу з комплексними числами.

3. Визначте функцію, яка за номером місяця повертає відповідну місяцю назву пори року.

4. Використовуючи функцію `remainder`, визначте, чи є задане ціле число парним.

5. Визначте функцію, яка отримує ціле число копійок n (n>99) та визначає число гривень в ньому.

6. Визначте функцію, яка отримує число n (n≤100) та обчислює:

– кількість цифр в числі n;

– суму цифр числа n;

– останню цифру числа n;

– першу цифру числа n.

7. «Копійка гривню береже»

Визначте функцію (сума копійки), яка отримує кількість копійок (ціле число з діапазону 1..100 000) та визначає ціле число гривень і залишок у копійках. При цьому слова "гривня" та "копійка" треба узгоджувати з числівниками. Наприклад: "1 гривня", але "73 гривні"; "1 копійка", але "25 копійок".

Виводити кількості гривень та копійок потрібно в окремих рядках. Якщо кількість гривень чи копійок дорівнює нулю – відповідний рядок виводити не треба.

8. Визначте функцію (відрахування сума), яка за нарахованою заробітною платнею розраховує суму, яку робітник отримає «на руки» Необхідно врахувати збір до пенсійного фонду (ПФ), збір до фонду соціального страхування з тимчасової втрати працездатності (ФСС_ТВП), збір до фонду соціального страхування на випадок безробіття (ФСС_ВБ), податок з доходів фізичних осіб (ПДФО). Ставки та методику обчислення відрахувань дізнайтесь самостійно з інтернет-джерел.

- Індивідуальні завдання. Кожен студент отримує власне завдання. Приклад одного із варіантів індивідуальних завдань:

Визначте функції всіма можливими способами. Обґрунтуйте, чому для їх визначення не можна скористатись іншими умовними виразами. Перевірте правильність вхідних даних та продемонструйте роботу функцій хоча б для трьох випадків:

1. Визначте функцію, яка отримує кількість товару (кг) та видає суму до сплати. Ціна товару визначається за формулою:

$$Ціна = \begin{cases} 15 \ грн, \ якщо \ кг < 5 \\ 13 \ грн, \ якщо \ 5 \le кг \le 7 \\ 11 \ грн, \ якщо \ кг > 7 \end{cases}$$

2. Визначте функцію y(x):

$$y(x) = \begin{cases} \sqrt{|\cos 9.2x^8 + 4|}, & якщо \ x = 1 \ або \ x = 2 \ або \ x = 3 \\ \sin|x^3 + 5x| & інакше \end{cases}.$$

3. Визначте функцію, яка отримує числовий код товару та надає довідку про ціну і кількість товару на складі.

- Тестування. Для перевірки засвоєння студентами даної теми в НМК

«Вступ до програмування» розроблено тестові завдання (приклади завдань наведено на рис. 2.18, рис. 2.19 та рис. 2.20).

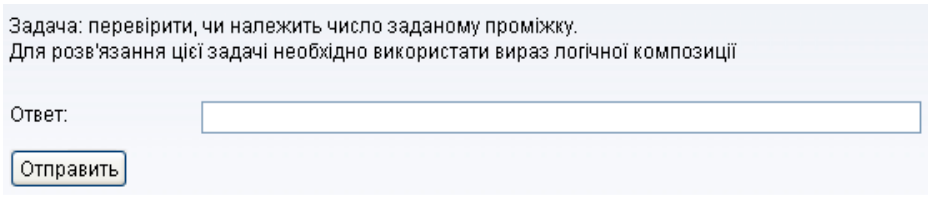


Рис. 2.18. Питання «Коротка відповідь»

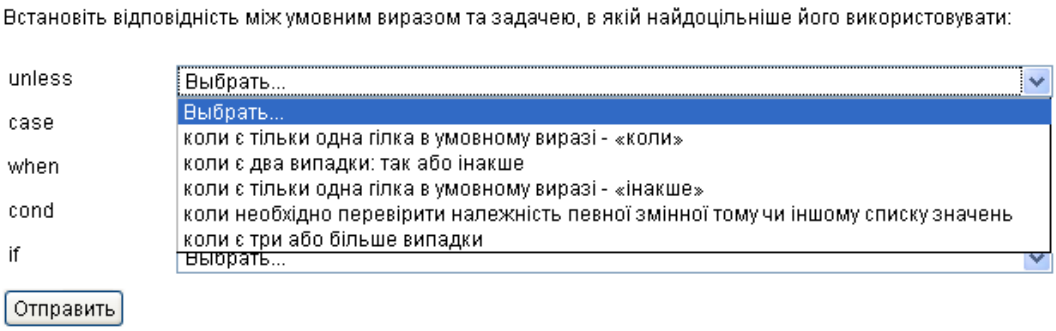
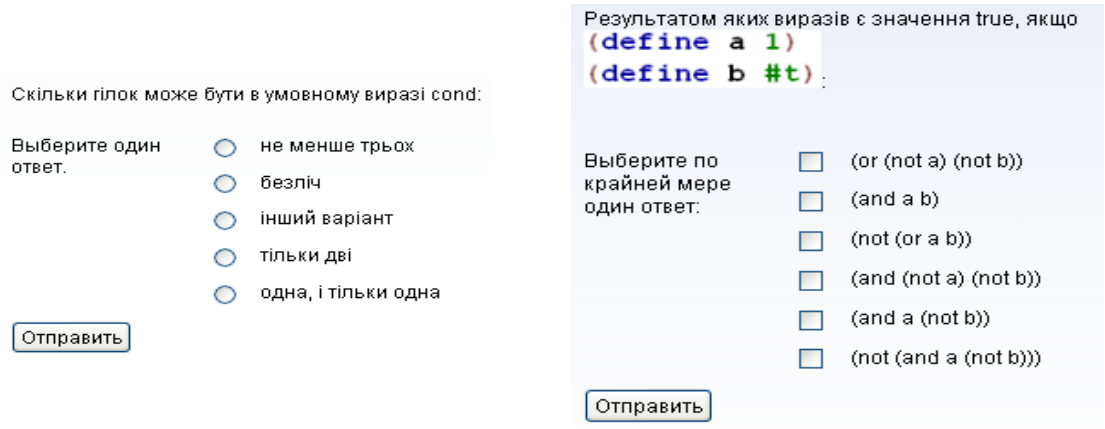


Рис. 2.19. Питання на відповідність



а) б)

Рис. 2.20. Питання «В закритій формі (множинний вибір)»
з однією правильною відповіддю (а) та з кількома правильними відповідями (б)

Таким чином, у процесі вивчення цієї теми в студентів:

1) формуються такі складові компетентностей з програмування:

– уявлення про задачі, для розв’язання яких необхідні умовні вирази;

– уміння розпізнавати задачі, для розв’язування яких необхідне використан-

ня умовних виразів;

– уміння записувати логічні вирази та визначати їх значення істинності;

– знання загальних та скорочених форм умовних виразів і вміння їх використовувати при розв’язуванні задач;

– знання відмінностей різних форм умовних виразів, уміння їх пояснити і навести відповідні приклади;

– уміння подавати похідні умовні вирази за допомогою елементарного умовного виразу;

– знання пріоритетних випадків для використання того чи іншого умовного виразу та вміння наводити відповідні приклади;

– уміння класифікувати задачі за типом умовного виразу, пріоритетного для їх розв’язування;

– знання загальних форм виразів логічної композиції та вміння їх використовувати при розв’язуванні задач;

– уміння наводити приклади для використання виразів логічної композиції;

– уміння демонструвати спрощення програми за рахунок використання виразів логічної композиції;

2) продовжується формування таких складових:

– знання етапів розв’язування задачі та вміння наводити відповідні приклади;

– знання етапів проектування програми та вміння наводити відповідні приклади;

– уміння скласти алгоритм розв’язування задачі;

– уміння здійснити постановку задачі для розв’язання її за допомогою комп’ютера;

– уміння спроектувати, описати, перевірити та проаналізувати результати виконання програми;

– уміння пояснити призначення та функції існуючої програми, знайти помилки в логіці розв’язання задачі;

– уміння порівняти різні способи розв’язування задачі та обрати серед них

раціональний;

– знання простих типів даних, функцій для роботи з ними;

– уміння визначати тип даних, необхідних для розробки конкретної задачі та вміння обґрунтовувати свій вибір;

– уміння спростити програму за рахунок використання композицій функцій;

– уміння коментувати програму;

– уміння користуватись довідковою інформацією;

– уміння використовувати засоби налагодження програми;

– уміння виводити результати програми у заданому форматі;

– здатність отримувати задоволення від процесу розв’язування задачі з допомогою комп’ютера;

– здатність виконувати поставлені викладачем завдання із задоволенням;

– здатність виявляти стійкий інтерес до вивчення програмування;

– уміння відстоювати власну думку;

– прагнення до самовдосконалення.

2.3.2 Створення експертної системи. Ця тема належить до третього модуля курсу – «Практична Scheme». Робота над будь-якою темою з цього модуля складається з кількох етапів:

1) ознайомлення з теоретичним матеріалом;

2) тестування програми та аналіз результатів;

3) модифікація та вдосконалення програми;

4) виконання колективних проектів без створення графічного інтерфейсу програми;

5) вдосконалення та захист колективних проектів з графічним інтерфейсом.

Так, наприклад, у проекті «Експертна система» (Додаток В) на *третьому етапі* студенти отримують такі завдання: доповніть ЕС меню, що містить пункти:

1. Провести експертну оцінку.

2. Вивести відомості про ... .

3. Пояснити останнє виведення.

4. Завершення роботи.

В результаті обрання першого пункту повинна запускатися машина виведення, другого – вводитися запит про предмет і виводитися всі правила і гіпотези, що його містять, третього – в зручній формі виводитись ланцюжок висновків останнього твердження (гіпотези) у вигляді послідовностей «Так як ..., можна зробити висновок, що ...». Вибір четвертого пункту повинен призводити до завершення роботи системи.

На *четвертому етапі* роботи над даною темою студенти отримують завдання: створити одну із ЕС:

1. ЕС, що допомагає вирішити, яку формулу застосовувати при розв'язуванні завдань зі шкільного курсу фізики (розділи кінематика і динаміка). В якості гіпотези задайте формули, правила побудуйте на основі наступного фрагменту бази знань:

```
багатозначний(формула)
дозвзн(розділ)=кінематика,динаміка
дозвзн(рух_тіла)=прямолінійний,обертальний
дозвзн(швидкість)=постійна,рівнозмінна
дозвзн(причини_руху)=не_враховуються,враховуються
дозвзн(сила)=сила_тертя,сила_тяжіння,сила_пружності,декілька_сил
дозвзн(зад_сил)=задані,не_задані
питання(причини_руху)=Чи враховуються причини руху?
питання(рух_тіла)=Який характер руху тіла?
питання(швидкість)=Який характер швидкості?
питання(сила)=Яка сила розглядається в задачі?
питання(зад_сил)=Чи задані сили?

правило1:
якщо причини_руху=не_враховуються
то розділ=кінематика.
правило2:
якщо причини_руху=враховуються
то розділ=динаміка.
правило3:
якщо розділ=кінематика
```

```
і рух_тіла=прямолінійний
і швидкість=постійна
то формула= X=X0+V*t
і формула= V=S/t.
правило4:
якщо розділ=кінематика
і рух_тіла=прямолінійний
і швидкість=рівнозмінна
то формула= a=(V-V0)/t
і формула= V=V0+a*t
і формула= X=X0+V0*t+A*t^2/2
bmp=pryskor.
правило5:
якщо розділ=кінематика
і рух_тіла=обертальний
то формула= w=f/t
і формула= V=w*r
і формула= a=V^2/r.
правило0:
якщо розділ=кінематика
то підрозділ=невизн.
правило6:
якщо розділ=динаміка
і зад_сил=задані
то підрозділ=рух_під_дією_сил.
правило7:
якщо розділ=динаміка
і зад_сил=не_задані
то підрозділ=закони_ньютона.
правило8:
якщо розділ=динаміка
і підрозділ=закони_ньютона
то формула= F=M*a
і формула=  F1=-F2.
правило9:
якщо розділ=динаміка
```

```
і підрозділ=рух_під_дією_сил
і сила=сила_тертя
то формула= F=-k*N.
правило10:
якщо розділ=динаміка
і підрозділ=рух_під_дією_сил
і сила=сила_тяжіння
то формула= F=G*M1*M2/r^2.
правило11:
якщо розділ=динаміка
і підрозділ=рух_під_дією_сил
і сила=сила_пружності
то формула= F=-k*X.
правило12:
якщо розділ=динаміка
і підрозділ=рух_під_дією_сил
і сила=декілька_сил
то формула= m*a=F1+F2+  +FN.
кінець
```

2. ЕС «Класифікатор рослин» на основі наступного фрагменту бази знань:

Якщо клас голонасінні і форма листка лускоподібна, то сімейство – кипарисові.

Якщо клас голонасінні і форма листка голкоподібна і конфігурація хаотична, то сімейство – соснові.

Якщо клас голонасінні і форма листка голкоподібна і конфігурація – 2 рівних ряди і сріблясta смуга, то сімейство – соснові.

Якщо клас голонасінні і форма листка голкоподібна і конфігурація – 2 рівних ряди і сріблястої смуги немає, то сімейство – болотний кипарис.

Якщо тип – дерева і форма листка широка і пласка, то клас – покритонасінні.

Якщо тип – дерева і невірно, що форма листка широка і пласка, то клас – голонасінні.

Якщо стебло зелене, то тип – трав’янисті.

Якщо стебло дерев’янисте і положення, що стелиться, то тип – ліани.

Якщо стебло дерев’янисте і положення прямостояче і один основний стовбур, то тип – дерева.

Якщо стебло дерев’янисте і невірно, що положення прямостояче і один основний

стовбур, то тип – чагарникові.

3. ЕС «Прогнозування повеней» на основі наступного фрагмента бази знань:

```
1 якщо рівень-води = високий і
   дощ = рясний,
   то місто = евакуювати
2 якщо рівень-води = високий і
   дощ = не сильний і
   снігу = багато,
   температура = висока,
   то місто = евакуювати
3 якщо рівень-води = високий і
   дощ = не сильний і
   снігу = багато,
   температура = середня і
   дощ = помірний,
   то місто = посилити увагу
4 якщо рівень-води = високий і
   дощ = ні і
   снігу = багато,
   температура = середня і
   дощ = слабкий,
   то місто = не турбуватись
5 якщо рівень-води = високий і
   дощ = не сильний і
   снігу = мало,
   то місто = не турбуватись
6 якщо рівень-води = невисокий і
   дощ = сильний і
   снігу = багато,
   температура = висока,
   то місто = посилити увагу
7 якщо рівень-води = невисокий і
   дощ = сильний і
   снігу = багато,
   температура = середня,
```

```
   то місто = не турбуватись
8 якщо рівень-води = невисокий і
   дощ = сильний і
   снігу = мало,
   то місто = не турбуватись
9 якщо рівень-води = високий і
   дощ = несильний,
   то місто = не турбуватись
```

4. Створіть ЕС «Класифікатор птахів», яка розрізняє орла, сокола і горобця, використовуючи в якості принципів класифікації ім’я об’єкта, довжину дзьоба, забарвлення пір’я, розмах крил, висоту польоту, аеродинамічні принципи.

5. ЕС за Вашим вибором.

П’ятий етап роботи: після вивчення теми «Графічний інтерфейс» студенти отримують завдання розробити графічний інтерфейс до вже створеної ними ЕС та публічно захищають виконані проекти.

У розробленому в ЕСПН Moodle НМК «Вступ до програмування» до всіх робіт з цього модуля розроблені такі матеріали (рис. 2.7):

– завдання проекту;

– робочий файл проекту;

– відео – приклад роботи проекту;

– звіт з проекту.

Завдання проекту містять теоретичний матеріал та більшість функцій, необхідних для «базової» роботи проекту (в наступних завданнях проект буде модифіковано й ускладнено).

Робочий файл проекту містить або програму, повністю готову до роботи (як, наприклад, у проекті «ЕС» – див. Додаток В), або ж за виключенням деяких необхідних функцій.

У відео – прикладі роботі проекту демонструється запуск роботи та можливі результати (рис. В.1, рис. В.2).

З даної теми передбачається підготовка та виступ студентів на лекціях з доповідями:

– Бази знань. Моделі подання знань;

– ЕС. Технологія розробки баз знань із використанням експертних оболонок;

– Покоління ЕС.

Цей вид роботи сприяє формуванню у студентів здатностей знаходити матеріал за темою доповіді, аналізу знайдених відомостей, формулюванню висновків, досвіду оволодіння прийомами публічного захисту виконаної роботи.

Під час виконання проектів з даного модуля у студентів окрім систематизації, поглиблення та вдосконалення теоретичних знань і вмінь, одержаних в попередніх модулях, формуються такі здатності, що характеризують сформованість компетентностей з програмування:

– *гносеологічна складова*:

– знання характеристик ЕС;

– знання структури ЕС та етапів розробки ЕС;

– знання класифікацій ЕС;

– знання відомих ЕС;

– уявлення про моделі подання знань;

– уявлення про сфери застосування та перспективи розвитку ЕС.

– *праксеологічну складова*:

– розуміння, налагодження та модифікація програм;

– бачення задачі одночасно на різних рівнях деталізації;

– узагальнення типових задач;

– використання знань і навичок в нових ситуаціях;

– розуміння зв’язку теорії програмування із практикою програмування;

– розуміння галузі застосування здобутих знань;

– встановлення причинно-наслідкових зв’язків;

– створення документації до програми.

– *аксіологічна складова*:

– емоційно-ціннісне ставлення до процесу розробки, опису, налагодження, тестування програм та аналізу результатів їх роботи;

– знаходження нових, нестандартних рішень задачі;

– внутрішня мотивація до опанування програмуванням;

– прагнення до самовдосконалення, потреба у саморозвитку гносеологічних та праксеологічних складових;

– самостійне прийняття рішень, критичне ставлення до чужих впливів,

– здатність за власним почином організовувати діяльність, ставити мету, в разі необхідності вносити в поведінку зміни;

– педантичність, дисциплінованість, акуратність, уважність, внутрішня керованість;

– цілеспрямованість, наполегливість, працездатність і ретельність у праці;

– внутрішня потреба у створенні програмних продуктів.

– *соціально-поведінкова складова:*

– здатність працювати як індивідуально, так і колективно (як лідер або член колективу);

– розподіл функцій і взаємостосунків у колективі;

– точне і лаконічне формулювання та висловлювання власних думок і ідей;

– аргументоване відстоювання власної точки зору;

– погодження своїх міркувань з іншими учасниками колективу;

– аналіз власних помилок;

– розуміння та готовність до конструктивного обговорення думок та ідей інших;

– розуміння відповідальності за виконану роботу;

– використання засобів для організації спільної роботи над проектом.

2.3.3 Графічний інтерфейс. Цей модуль навчальної програми передбачає декілька навчальних цілей:

– формування вмінь роботи з підключення модулів до програми;

– формування вмінь обирати засоби для розв’язання задачі створення графічного інтерфейсу та обґрунтовувати свій вибір;

– вдосконалення вмінь використовувати можливості обраних засобів (довідка, налагодження програми, налаштування необхідних параметрів та ін.);

– ознайомлення з основними поняттями та принципами об'єктно-орієнтованого програмування;

– удосконалення проектів з попереднього модуля шляхом розробки їх графічного інтерфейсу.

Основним методом навчання при роботі з даним модулем є *метод демонстраційних прикладів* – важливо, щоб студенти мали можливість не лише одразу переконатись в дієвості програмного коду, але й проекспериментувати з ним, змінюючи ті чи інші параметри, та *метод проектів* – студенти завершують виконання проектів з попереднього модуля.

Для створення графічного інтерфейсу програми необхідно визначити мову racket/gui, оскільки вона поєднує в собі всі прив'язки необхідних модулів. Тому будь-яку програму з графічним інтерфейсом необхідно починати з `#lang racket/gui`.

Основні конструкції для програм з графічним інтерфейсом – це вікна: рамки, діалоги, меню, кнопки, флажки, текстові поля, радіо-кнопки та ін. Цей інструментарій надається через вбудовані класи. За домовленістю, імена класів закінчуються %. Наприклад, `frame%`:

```
 ; створимо Рамку як об'єкт класу frame%
(define Рамка (new frame%
      [label "Приклад"]))
 ; відобразимо Рамку
 ; пошлемо об'єкту Рамка повідомлення show з параметром #t
(send Рамка show #t)
```

Результат – рис. 2.21.

Рис. 2.21. Вікно-рамка мінімальних розмірів

Будь-який клас має визначені поля та повідомлення (функції). Вбудовані класи забезпечують різні способи обробки повідомлень. Наприклад, при створенні об'єкту класу `button%`, можна визначити функцію оберненого виклику для ви-

падку, коли користувач натискає кнопку.

Наступний приклад створює рамку, повідомлення з заголовком «Поки що подій немає» та кнопку. При натисненні кнопки, заголовок повідомлення змінюється на «Натиснули кнопку» (рис. 2.22).

```
; створимо Рамку як об'єкт класу frame%
 (define Рамка (new frame%
     [label "Приклад"]))

; створимо повідомлення на рамці
 (define Повідомлення (new message%
     [parent Рамка]
     [label "Поки що подій немає..."]))

 ; створимо кнопку на рамці
 (new button%
     [parent Рамка]
     [label "Натисни мене"]
 ; функція оберненого виклику для натиснення кнопки
     [callback (lambda (button event)
                 (send Повідомлення set-label "Натиснули кнопку"))])

; відобразимо рамку
 (send Рамка show #t)
```

а) до натиснення кнопки б) після натиснення кнопки

Рис. 2.22. Опрацювання подій

В системі розсилок графічного інтерфейсу повідомлення опрацьовуються послідовно, тобто, наступна функція оберненого виклику або повідомлення чекають, доки обробник подій повернеться за наступним повідомленням. Для ілюст-

рації послідовності обробки повідомлень додамо у попередній код ще одну кнопку (рис. 2.23):

```
; створимо на рамці ще одну кнопку, натиснення якої приводить
; до бездіяльності – "засипання" об'єкту на 5 секунд
 (new button%
     [parent Рамка]
     [label "Пауза"]
     [callback (lambda (button event)
       (sleep 5))])
```

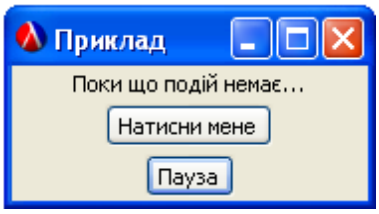


Рис. 2.23. Послідовність опрацювання подій

При натисненні кнопки «Пауза», вся рамка перестає реагувати на будь-які події протягом 5 секунд. Але після закінчення 5 секунд, всі неопрацьовані події будуть послідовно виконані.

На додачу до обробки повідомлень та функцій оберненого виклику, в графічному інтерфейсі можна керувати розміщенням елементів, як шляхом порядку їх створення, так і шляхом задання їх батьківських об'єктів. Наприклад, всі об'єкти на вертикальній панелі розміщуються зверху вниз, а на горизонтальній – зліва направо (якщо не задано їх інакше розміщення шляхом указування координат верхнього лівого кута).

Типи вікон та ієрархія класів наведено у Додатку Г. В посібнику [191] наведено детальний опис та приклади використання деяких вікон (див. Додаток Д – клас «кнопка-прапорець» – `check-box%`).

Базовим завданням даного модуля є створення графічного інтерфейсу та реалізація відповідних функцій програми «Калькулятор Плюс» (рис. 2.24), а варіативними завданнями є удосконалення проектів попередніх модулів шляхом розробки їх графічних інтерфейсів.

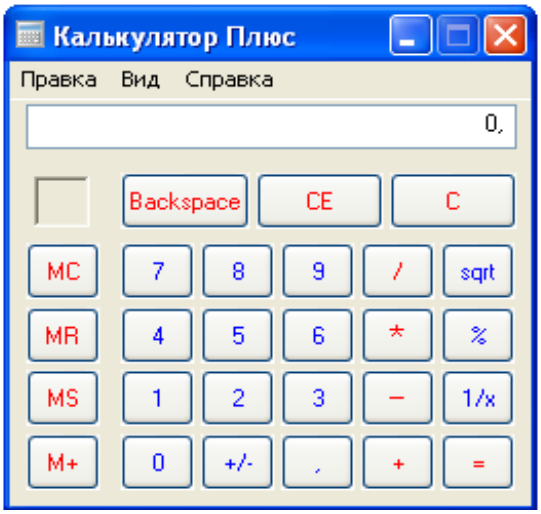


Рис. 2.24. Програма «Калькулятор Плюс»

2.4 Організація, проведення та результати педагогічного експерименту

Розробка й апробація теоретичних положень дисертаційного дослідження проходило у три етапи:

1) констатувальний (2005–2006 рр.);

2) пошуковий (2006–2009 рр.);

3) формувальний (2009–2012 рр.).

Завданням першого етапу дисертаційного дослідження було вивчення існуючого стану процесу формування компетентностей з програмування майбутніх учителів інформатики та виділення вихідних положень дослідження. Для реалізації поставленого завдання було виконано аналіз навчальних програм та підручників з шкільного курсу інформатики, на підставі якого було визначено фундаментальні інформатичні компетентності, спільні для старших класів середньої школи та молодших курсів ВНЗ. Головну увагу на констатувальному етапі дослідження було приділено математичним основам програмування (зокрема, математичній логіці та її застосуванням). Основні результати, отримані в процесі експериментальної роботи, відображені у статтях [169; 190] та доповідалися на VI міжнародній науково-практичній конференції «Теорія та методика навчання фундаментальних дисциплін у вищій школі» (Кривий Ріг, 2006).

Результати констатувального експерименту виявили наступне:

1. Незважаючи на широку поширеність функціонального підходу до навчання початків програмування у зарубіжних країнах, у країнах СНД переважають імперативний та об'єктний підходи до формування компетентностей з програмування. При цьому навчання початків програмування у середній школі із застосуванням діяльнісних середовищ Logo, Scratch, Squeak, Alice та інших відбувається без урахування особливостей реалізації функціонального підходу в мовах програмування.

2. Мови та середовища функціонального програмування ряд вітчизняних дослідників пропонують використовувати виключно для ілюстрації основних положень теорії програмування та інтелектуальних систем, що відповідає історичним застосуванням мов ML та Lisp. При цьому в якості аргументів того, що мови функціонального програмування не можуть бути використані для формування компетентностей з програмування, наводяться переважно технічні обмеження середовищ функціонального програмування, характерні для програмних засобів 20-30-річної давнини (інтерпретована природа мов, обмеження розміру стеку для рекурсивних програм, повільне виконання, неможливість роботи із зовнішніми бібліотеками тощо).

Виявлені невідповідності:

– між станом проблеми у спеціальній і методичній літературі та сучасним рівнем розвитку засобів обчислювальної техніки і програмних систем;

– між педагогічним потенціалом діяльнісних середовищ функціонального програмування та невикористанням функціонального підходу в процесі їх застосування;

– між сучасною тенденцією до формування компетентностей з програмування на мультипарадигменній основі та наявним станом навчанням початків програмування виключно імперативними засобами зумовили вибір мети дослідження [193], виділення його об'єкту та предмету і формулювання завдань.

Другий етап дисертаційного дослідження характеризувався пошуком програмних засобів реалізації функціонального підходу, придатних для застосування в процесі навчання початків програмування студентів перших курсів педагогічних

ВНЗ. З метою обґрунтованого вибору засобів навчання та розробки методичної системи формування компетентностей з програмування у 2006-2007 н.р. було проведене анкетування (Додаток Е) студентів І курсу спеціальностей «Математика та інформатика», «Фізика та інформатика», «Інформатика» КДПУ (КНУ), в результаті якого було виявлено наступне:

– систематично навчалися програмуванню (оцінки 3–5) не більше 50% вступників;

– навчання програмування, за одиничними виключеннями, відбувалося у середовищах імперативного програмування;

– в процесі навчання програмування методи активного навчання (зокрема, метод проектів) і засоби організації спільної роботи не використовувались;

– навчання програмування розглядалося без відповідних математичних основ;

– переважно низький рівень пізнавальної активності в процесі навчання програмування був обумовлений високим порогом входження у предмет (значним обсягом знань та вмінь, необхідних для написання особистісно значущої програми).

Виявлені проблеми надали можливість сформулювати вимоги до засобів навчання програмування на основі функціонального підходу:

– стандартність та локалізованість інтерфейсу користувача;

– наявність можливості задання імен ідентифікаторів рідною мовою;

– мале синтаксичне ядро мови програмування;

– можливість створення коротких програм, що мають розвинену функціональність;

– можливість модифікації програмного коду в процесі його виконання;

– підтримка багатьох підходів до програмування в межах одного середовища.

Огляд існуючих засобів навчання програмування показав, що жоден з них не відповідає сформульованим вимогам, проте найбільш близьким до їх реалізації було середовище програмування DrScheme. Головними недоліками його були не-

локалізованість інтерфейсу користувача. Додатковим обмеженням на його застосування була мала кількість джерел українською та російською мовами, присвячених функціональній мові Scheme.

Для усунення виявлених недоліків було виконано локалізацію середовищ PLT Scheme та DrScheme українською та російськими мовами [179, 23]. За результатами дослідної експлуатації авторської локалізації DrScheme 4.1 у 5 комп'ютерних класах КДПУ (КНУ) було прийняте рішення про її включення до версій PLT Scheme та DrScheme (починаючи з червня 2010 р. – Racket та DrRacket), що розповсюджуються через офіційний сайт проекту (адреса – http://www.racket-lang.org/) (рис. 2.25).

Створена вітчизняна версія середовища DrRacket задовольняла всім поставленим вимогам до засобу навчання програмування на основі функціонального підходу та може розглядатися як середовище моделювання, а реалізація мови Scheme у ньому – як комп'ютерна інтерпретація математичних моделей функціонального підходу (λ-числення та комбінаторної логіки), що надало можливість використати DrRacket надалі як основне навчальне середовище, мобільне відносно апаратної та програмної складової обчислювальної системи.

Водночас безпосереднє застосування даного середовища на мобільних пристроях є недоцільним через орієнтацію інтерфейсу користувача на екрани з великою роздільної здатністю. Це зумовило необхідність розроблення простого мобільного середовища, орієнтованого на комп'ютерні пристрої з середньою та низькою роздільною здатністю екрану. На основі інтерпретатора мови Scheme, реалізованого на Java (автор – Скотт Міллер), було створено 2 аплети, що надають інтерфейс українською та російськими мовами [179, 25].

Виконана розробка та локалізація програмного забезпечення створили умови для організації навчання на основі інтеграції технологій аудиторного, дистанційного та мобільного навчання. Запровадження методів активного навчання вимагало залучення програмних засобів організації спільної роботи у мобільному навчальному середовищі – web-сервісу Dropbox та Skype [170].

Рис. 2.25. Інформація про DrRacket 5.0

Реалізація процесу формування у студентів педагогічних університетів компетентностей з програмування на основі функціонального підходу вимагала особливої уваги до змісту навчання основ програмування. Експериментальна робота за методикою, розробленою на факультеті електромеханіки та комп'ютерних наук Массачусетського технологічного інституту (США) та узагальненою в матеріалах навчального курсу «Структура та інтерпретація комп'ютерних програм», показала наступні її особливості:

1) підручник з курсу та методичні матеріали для викладача передбачають велику кількість лекцій-семінарів (близько 80 год.), в ході яких основні теоретичні поняття моделюються засобами мови Scheme;

2) лабораторні роботи з курсу являють собою, починаючи з першої ж робо-

ти, навчальні проекти, що вимагає попереднього знайомства студентів з даною навчальною технологією;

3) курс спрямований на формування компетентностей з програмування майбутніх інженерів-електриків, фахівців з інформатики.

Навчальна дисципліна «Шкільний курс інформатики» та курс за вибором «Вступ до програмування» передбачають 27 та 18 год. лекцій відповідно, що не дозволяє організувати лекції-семінари. Крім того, недостатня впровадженість проектних технологій у навчальний процес середньої школи не дозволяє їх застосувати на початку роботи за курсом. І, нарешті, система компетентностей майбутніх інженерів та майбутніх викладачів суттєво відрізняється.

Таким чином, виникла необхідність не в модифікації існуючої методичної системи навчання програмування студентів І курсу через зміну її технологічної складової, а в розробленні нової методичної системи, змістову частину якої було узагальнено в навчальному посібнику «Схематичне програмування (початки програмування: функціональний підхід)» [191].

Упровадження розробленої методичної системи та експериментальна перевірка її ефективності були виконані на третьому етапі дослідження. Протягом 2009-2010, 2010-2011 та 2011-2012 н.р. за розробленою методикою навчалися 340 студентів фізико-математичного, природничого, технолого-педагогічного факультетів КДПУ (КНУ), 53 студенти факультету природничої і фізико-математичної освіти Глухівського національного педагогічного університету імені Олександра Довженка, 21 студент Криворізького металургійного факультету НМетАУ (КНУ), 98 студентів Криворізького технічного університету (КНУ) та 14 студентів Криворізького відокремленого підрозділу ЗІЕІТ. Враховуючи спрямованість методичної системи на формування компетентностей з програмування у студентів педагогічних університетів, результати її впровадження в інших ВНЗ враховувались, проте статистично не опрацьовувались.

Контрольні та експериментальні групи формувалися наступним чином:

– до контрольних груп (КГ) відносилися студенти першого курсу 2009 та 2010 років вступу (групи МІ-09-1, І-09-2, ФІ-10-1), навчання основ програмування

яких здійснювалось за традиційної методикою (на основі імперативного підходу);

– до експериментальних груп (ЕГ) відносилися студенти першого курсу 2009 та 2010 років вступу (групи МІ-09-2, І-09-1, ФІ-10-2), навчання основ програмування яких здійснювалось за розробленою методикою (на основі функціонального підходу) (табл. 2.3).

Таблиця 2.3

Розподіл студентів по групах на формувальному етапі експерименту

Шифр групи	**Рік вступу**	**Кількість студентів**	**КГ чи ЕГ**
МІ-09-1	2009	25	КГ
МІ-09-2	2009	25	ЕГ
І-09-1	2009	15	ЕГ
І-09-2	2009	15	КГ
ФІ-10-1	2010	13	КГ
ФІ-10-2	2010	12	ЕГ
Всього КГ: 53 студенти			
Всього ЕГ: 52 студенти			

З метою забезпечення рівних умов проведення експерименту: 1) навчання в ЕГ та КГ здійснювалось по можливості одним і тим самим викладачем; 2) утворені групи було досліджено на статистичну рівнозначність шляхом аналізу оцінок студентів КГ та ЕГ зі шкільної інформатики (результати аналізу подано в табл. 2.4).

Таблиця 2.4

Порівняльний розподіл студентів

за оцінками зі шкільної інформатики

Оцінка	**Кількість студентів**		**% студентів**	
	КГ	ЕГ	КГ	ЕГ
2	0	0	0	0
3	0	1	0	1,92
4	19	28	35,85	53,85
5	34	23	64,15	44,23
Всього	**53**	**52**	**100**	**100**

Гістограми порівняльного розподілу студентів (у відсотках, оскільки маємо різну кількість студентів для КГ та ЕГ) за оцінками зі шкільної інформатики подано на рис. 2.26.

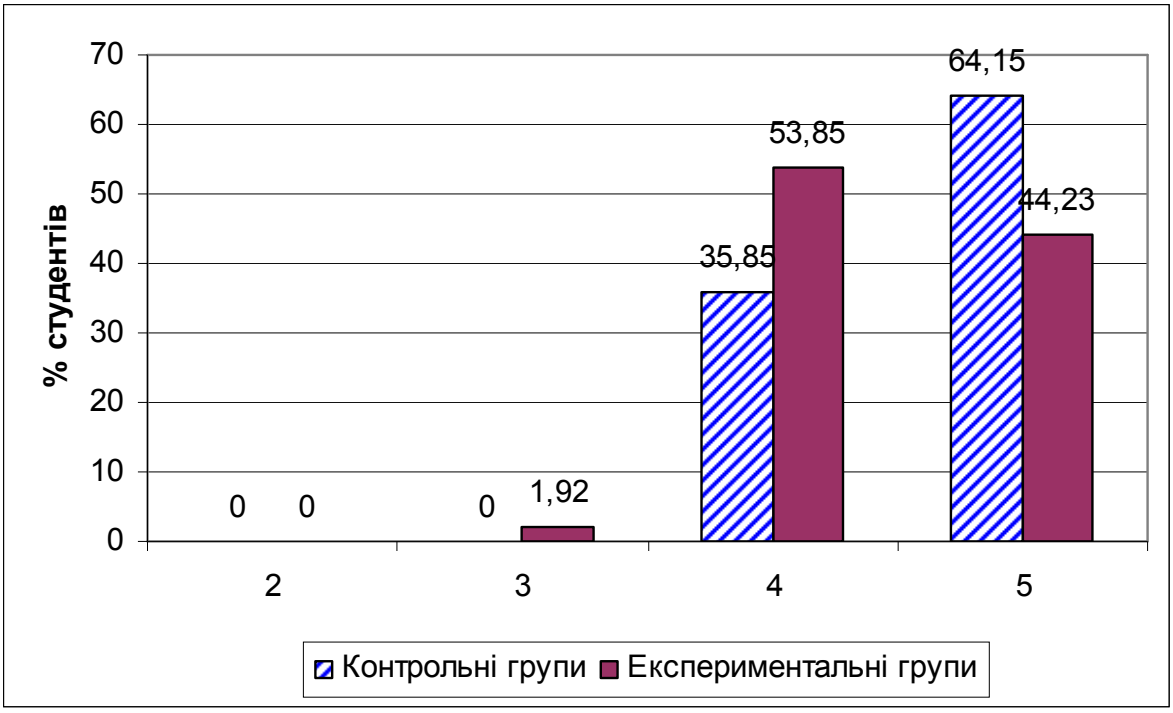


Рис. 2.26. Порівняльний розподіл студентів в КГ та ЕГ
за оцінками зі шкільної інформатики

Опрацювання результатів аналізу оцінок студентів зі шкільної інформатки здійснювалось методами математичної статистики [253]. Для оцінювання відхилення розподілу в ЕГ від розподілу в КГ використаємо λ-критерій Колмогорова-Смирнова [253, 34].

Цей критерій є непараметричним і застосовується за таких умов:

– вибірки випадкові і незалежні;

– категорії впорядковані за зростанням або спаданням.

Оскільки обидві ці умови для отриманих вибірок виконуються, можна застосувати цей критерій. Позначимо: $F(x)$ – невідома функція розподілу ймовірностей оцінок зі шкільної інформатики студентів в КГ; $G(x)$ – невідома функція розподілу ймовірностей оцінок зі шкільної інформатики студентів в ЕГ.

Нульова гіпотеза H_0 : $F(x)=G(x)$. Альтернативна гіпотеза H_1 : $F(x)\neq G(x)$

Коли гіпотеза $H_0 : F(x)=G(x)$ справджується, відхилення $d = \sup_x |G(x) - F(x)|$ мале, а коли гіпотеза H_0 не справджується, це відхилення велике.

Результати обробки експериментальних даних наведені в табл. 2.5, з якої знаходимо, що d=0,199.

Таблиця 2.5

Обчислення λ-критерію Колмогорова

(для оцінок зі шкільної інформатики)

Оцінки	Абсолютна частота		Накопичена частота		Відносна накопичена частота		d
	КГ	ЕГ	КГ	ЕГ	КГ	ЕГ	
2	0	0	0	0	0,000	0,000	0,000
3	0	1	0	1	0,000	0,019	0,019
4	19	28	19	29	0,358	0,558	0,199
5	34	23	53	52	1,000	1,000	0,000
d_{max}							0,199

За формулою $\lambda = d_{max}\sqrt{\frac{n_1 \cdot n_2}{n_1 + n_2}}$ обчислимо $\lambda_{емп}$ = 1,02. Побудуємо вісь значущості (рис. 2.27).

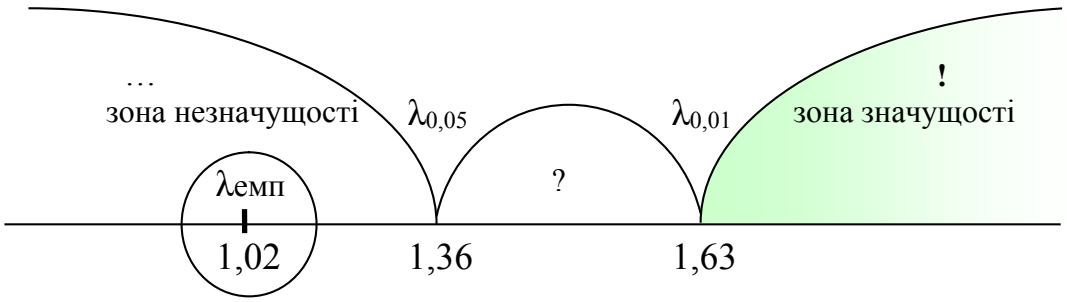


Рис. 2.27. Вісь значущості для λ-критерію Колмогорова-Смирнова (для оцінок зі шкільної інформатики)

Таким чином, гіпотеза H_0 справджується, тобто КГ та ЕГ не мають статистично значущих відмінностей у розподілі оцінок студентів зі шкільної інформатики.

Для перевірки ефективності розробленої методичної системи було виконано

порівняння розподілів студентів КГ та ЕГ за рівнями сформованості компетентностей з програмування. Діагностика рівнів сформованості компетентностей з програмування виконувалась за показниками з табл. 2.6.

Таблиця 2.6

Оцінювання рівнів сформованості компетентностей з програмування та вага складових у загальній сформованості компетентностей

i	**Складова** (S_i, $i = \overline{1,\ 4}$)	**Рівні сформованості складової**			**вага складової** (p_i, $i = \overline{1,\ 4}$)
		низький	**достатній**	**високий**	
		0	**1**	**2**	
1.	гносеологічна	має низький рівень знань про основні синтаксичні форми, типи даних та етапи розробки програм	має певні знання про основні синтаксичні форми, типи даних та етапи розробки програм	має високий рівень знань про основні синтаксичні форми, типи даних та етапи розробки програм	0,4
2.	праксеологічна	не вміє пояснити призначення та функції існуючої програми, описати етапи розробки програм, пояснити створення похідних типів даних, спроектувати, описати, перевірити та проаналізувати результати виконання програми	частково вміє пояснити призначення та функції існуючої програми, описати етапи розробки програм, пояснити створення похідних типів даних, спроектувати, описати, перевірити та проаналізувати результати виконання програми	вміє пояснити призначення та функції існуючої програми, описати етапи розробки програм, пояснити створення похідних типів даних, спроектувати, описати, перевірити та проаналізувати результати виконання програми	0,3
3.	аксіологічна	процес проектування, опису, перевірки та аналізу результатів не становить для студента жодної цін-	низький рівень зацікавленості процесами проектування, опису, перевірки та аналізу результатів;	висока зацікавленість та внутрішня мотивація до опанування програмуванням, готовність	0,2

i	Складова (S_i, $i = \overline{1,\ 4}$)	Рівні сформованості складової			вага складової (p_i, $i = \overline{1,\ 4}$)
		низький	достатній	високий	
		0	1	2	
		ності; внутрішня мотивація до опанування програмуванням відсутня	присутня певна внутрішня мотивація до опанування програмуванням	застосовувати набуті знання та вміння у практичній діяльності	
4.	соціально-поведінкова	відсутня здатність до співпраці у процесі програмування; невміння використовувати засоби спільної роботи в процесі програмування	інколи має здатність до співпраці у процесі програмування; частково вміє використовувати засоби спільної в процесі програмування	постійна готовність до співпраці в процесі програмування; раціональне та творче використання засобів спільної роботи в процесі програмування	0,1

Кожна складова компетентностей з програмування оцінювалась за трьохбальною шкалою (від 0 до 2), що відповідала недостатньому, достатньому та високому рівню сформованості відповідної складової. Показники сформованості складової діагностувалися методами педагогічного спостереження, в процесі контролю знань, захисту лабораторних робіт. Вага кожної складової у загальній сформованості компетентностей з програмування визначалась методом експертних оцінок (за результатами опитування викладачів інформатики, що брали участь у VII міжнародній науково-практичній конференції «Теорія та методика навчання фундаментальних дисциплін у вищій школі» (Кривий Ріг, 2009). Таким чином, числове значення рівня сформованості компетентностей з програмування визначалось за формулою:

$$B = \frac{1}{\min\limits_{i=1,4} p_i} \sum_{i=1}^{4} S_i p_i \ .$$

Множник $\frac{1}{\min\limits_{i=1,4} p_i}$ перед сумою (в нашому випадку – 10) обирається для того, щоб отримане в результаті числове значення рівня сформованості компетентнос-

тей було цілим числом. Таким чином, у відповідності до експертних оцінок ваги всіх складових компетентностей з програмування, значення *B* може бути в межах від 0 до 20. Отримані числові значення були розподілені за рівнями сформованості компетентностей з програмування в такий спосіб (табл. 2.7):

Таблиця 2.7

Розподіл рівнів сформованості компетентностей з програмування

Рівні	**Бали**
низький	0–5
достатній	6–10
середній	11–15
високий	16–20

Результати формувального етапу педагогічного експерименту в КГ та ЕГ наведено у таблиці 2.8. Гістограми порівняльного розподілу студентів (у відсотках, оскільки маємо різну кількість студентів для КГ та ЕГ) за рівнями сформованості компетентностей з програмування подано на рис. 2.28.

Таблиця 2.8

Порівняльний розподіл студентів
за рівнями сформованості компетентностей з програмування

рівень сформованості компетентностей з програмування	**% студентів**	
	КГ	**ЕГ**
низький	15,09	9,62
достатній	45,28	17,31
середній	28,30	51,92
високий	11,33	21,15

Опрацювання результатів експерименту та оцінка ефективності розробленої методичної системи здійснювалась методами математичної статистики [253]. Оскільки задача полягає у виявленні відмінностей в розподілі певної ознаки (сформованості компетентностей з програмування) при порівнянні двох емпіричних розподілів згідно [253, 34] можна скористатись χ^2-критерієм Пірсона, λ-критерієм Колмогорова-Смирнова або ж φ^*-критерієм (кутовим перетворенням Фішера).

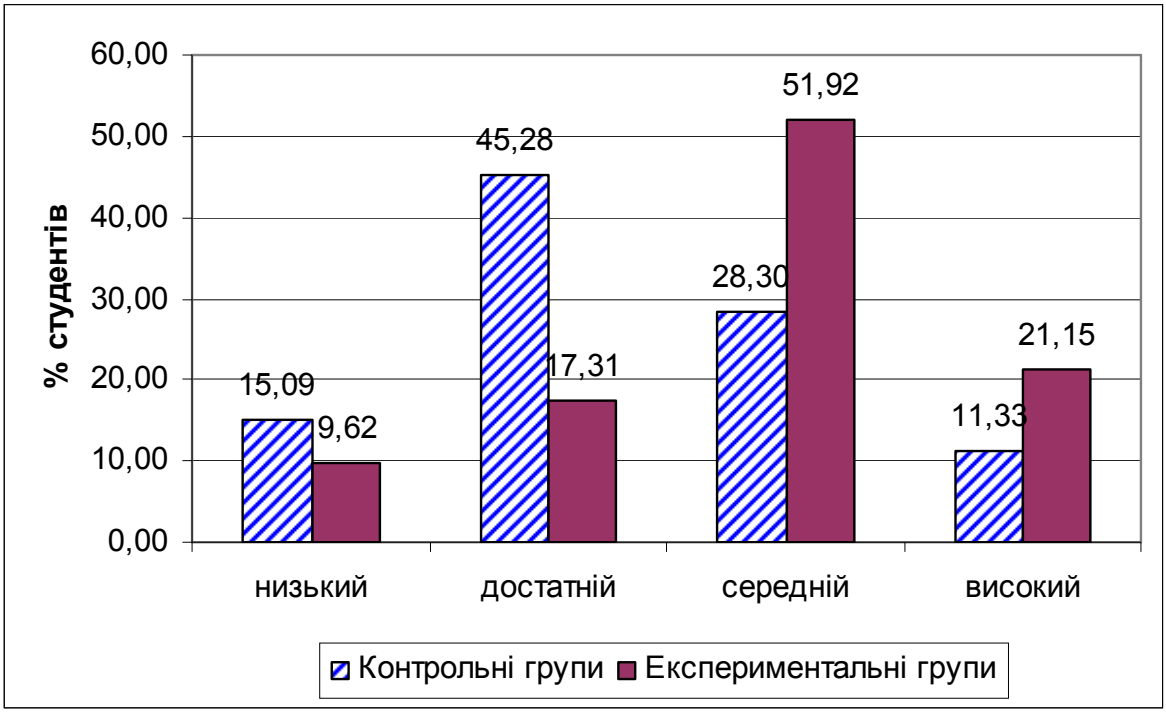


Рис. 2.28. Порівняльний розподіл студентів після формувального етапу педагогічного експерименту в КГ та ЕГ
за рівнями сформованості компетентностей з програмування

χ^2-критерій Пірсона

Обмеження критерію: об'єм вибірки n ≥ 30; частота для кожної комірки таблиці не повинна бути менше 5 (саме тому цей критерій не було використано при порівнянні розподілів оцінок зі шкільної інформатики студентів КГ та ЕГ); обрані розряди мають вичерпувати весь розподіл, тобто охоплювати весь діапазон варіативності ознак; якщо одне спостереження віднесене до одного розряду, то воно вже не може бути віднесене ні до якого іншого розряду. Всі ці умови для отриманих вибірок виконуються.

Для критерію оцінка рівнів значущості визначається числом ступенів свободи, яке позначається грецькою літерою v і розраховується за формулою $v = c – k–1$, де c – кількість категорій у вибірці, k – кількість накладених незалежних умов.

У нашому дослідженні вибірки випадкові і незалежні. Шкалою вимірювань є шкала з c=4 категоріями (низький, достатній, середній та високий рівні), накла-

дено дві незалежні умови. Отже, кількість ступенів свободи $v = c - 2 - 1 = 1$.

Нульова гіпотеза H_0: ймовірність попадання студентів КГ та ЕГ вибірки (n_1=53, n_2=52) в кожну з i (i= низький, достатній, середній та високий рівні) категорій однакова, тобто H_0: $p_{1i}=p_{2i}$ (i= низький, достатній, середній та високий рівні).

Альтернативна гіпотеза H_1: $p_{1i}\neq p_{2i}$ хоча б для однієї із C категорій.

p_{1i} – ймовірність оцінювання рівня сформованості компетентностей з програмування учасників КГ на i рівні (i= низький, достатній, середній та високий);

p_{2i} – ймовірність оцінювання рівня сформованості компетентностей з програмування учасників ЕГ на i рівні;

Q_{1i} – кількість учасників КГ, в яких компетентність сформована на i рівні;

Q_{2i} – кількість учасників ЕГ, в яких компетентність сформована на i рівні.

Значення χ^2 обчислюється за формулою:

$$\chi^2 = \frac{1}{n_1 n_2}\sum_{i=0}^{C-1}\frac{\left(n_1 Q_{2i} - n_2 Q_{1i}\right)^2}{Q_{1i} + Q_{2i}}.$$

Якщо v=1, необхідно внести поправку на неперервність (перед піднесенням до квадрату різниці частот зменшити модуль різниці на 0,5) [253, 134].

Результати обчислення статистики вказаних вибірок наведені в табл. 2.9. З таблиці IX [253, 328] значень χ^2 для рівня значущості α=0,05 і кількості степенів свободи v=1 визначаємо критичне значення статистики $\chi^2_{крит}$=3,84, а для α=0,01 $\chi^2_{крит}$=6,64. Побудуємо вісь значущості для отриманих даних (рис. 2.29).

Таблиця 2.9

Обчислення χ^2 для контрольної та експериментальної груп після формувального етапу експерименту

i	Q_{1i}	Q_{2i}	S_{12i}
0	8	5	1742,33
1	24	9	17990,01
2	15	27	10075,01
3	6	11	4304,13
χ^2			**12,38**

З рис. 2.29 маємо, що $\chi^2_{емп} > \chi^2_{0,01}$, тобто $\chi^2_{емп}$ не потрапляє у критичну зону.

Це є основою для відхилення нульової гіпотези H_0. Прийняття альтернативної гіпотези H_1 дає підстави стверджувати, що ці вибірки мають статистично значущі відмінності, тобто *експериментальна методична система більш ефективна, ніж традиційна*.

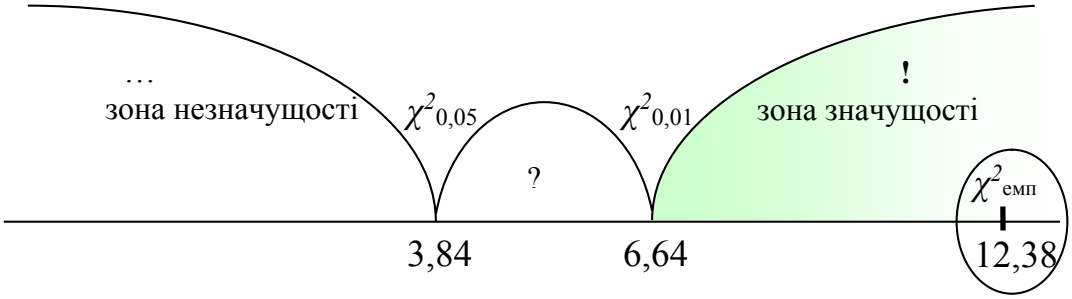


Рис. 2.29. Вісь значущості для χ^2-критерію Пірсона
(після формувального етапу педагогічного експерименту)

λ-критерій Колмогорова-Смирнова

Для більшої переконливості виконаємо перевірку отриманих під час формувального етапу педагогічного експерименту вибірок за λ-критерієм Колмогорова-Смирнова.

Умови про випадковість та незалежність вибірок і впорядкованість категорій для даного розподілу виконуються. Позначимо: $F(x)$ – невідома функція розподілу ймовірностей рівня сформованості компетентностей з програмування студентів в КГ; $G(x)$ – невідома функція розподілу ймовірностей рівня сформованості компетентностей з програмування студентів в ЕГ.

Нульова гіпотеза $H_0 : F(x)=G(x)$.

Альтернативна гіпотеза $H_1 : F(x)\neq G(x)$

Коли гіпотеза $H_0 : F(x)=G(x)$ справджується, відхилення

$$d = \sup_x |G(x) - F(x)|$$

мале, а коли гіпотеза H_0 не справджується, це відхилення велике.

Результати обробки експериментальних даних наведені в табл. 2.10, з якої видно, що $d_{max}= 0{,}335$.

Таблиця 2.10

Обчислення λ-критерію Колмогорова
після формувального етапу експерименту

Рівні	Абсолютна частота		Накопичена частота		Відносна накопичена частота		D
	КГ	ЕГ	КГ	ЕГ	КГ	ЕГ	
низький	8	5	8	5	0,151	0,096	0,055
достатній	24	9	32	14	0,604	0,269	0,335
середній	15	27	47	41	0,887	0,788	0,098
високий	6	11	53	52	1,000	1,000	0,000
d_{max}							0,335

За формулою $\lambda = d_{max}\sqrt{\frac{n_1 \cdot n_2}{n_1 + n_2}}$ обчислимо $\lambda_{емп}$ = 1,71. Побудуємо вісь значущості (рис. 2.30).

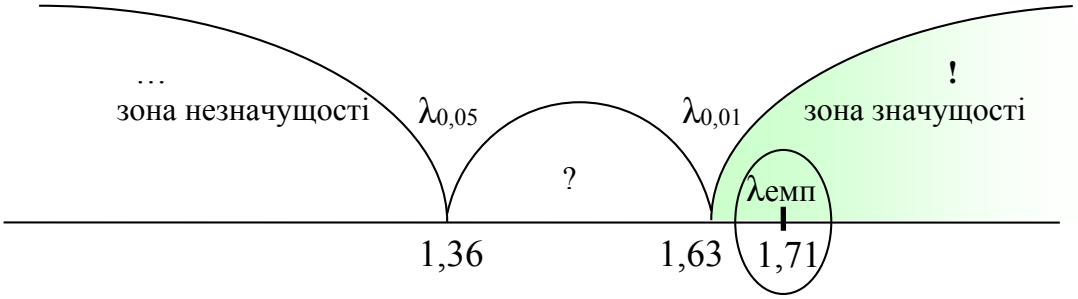


Рис. 2.30. Вісь значущості для λ-критерію Колмогорова-Смирнова
(після формувального етапу педагогічного експерименту)

Оскільки $\lambda_{емп} > \lambda_{0,01}$, нульова гіпотеза $H_0 : F(x)=G(x)$ відхиляється і приймається альтернативна гіпотеза $H_1 : F(x)\neq G(x)$. Це означає, що існує відмінність розподілу рівня сформованості компетентностей з програмування у студентів, які навчалися за традиційною методичною системою і експериментальною. Таким чином, студенти, що навчалися в ЕГ, мали більш високий рівень сформованості компетентностей з програмування.

Ураховуючи, що в ЕГ формування компетентностей з програмування здійс-

нювалось за розробленою методичною системою, можна припустити, що саме це і дало можливість досягти кращих результатів. Отже, можна говорити про експериментальне підтвердження висунутої гіпотези.

Кутове перетворення Фішера

Для розрахунку кутового перетворення Фішера враховується наступне: в КГ низький та достатній рівні сформованості компетентностей з програмування спостерігались у 60,4 % студентів, середній та достатній – у 39,6 %; в ЕГ ефект 26,9 % і 73,1 % відповідно (табл. 2.11).

Таблиця 2.11

Розподіл студентів в КГ та ЕГ за спостережуваним ефектом

	Компетентності сформовані на низькому або достатньому рівнях	**Компетентності сформовані на середньому або високому рівнях**
КГ	60,4	39,6
ЕГ	26,9	73,1

Експериментальні дані повністю задовольняють обмеження, що накладаються кутовим перетворенням Фішера: а) жодна з часток, що порівнюються, не дорівнює нулю (тому цей критерій не було використано при порівнянні розподілів оцінок зі шкільної інформатики студентів КГ та ЕГ); б) кількість спостережень у обох вибірках більше 5, що дозволяє будь-які співставлення.

Сформулюємо гіпотези:

H_0: Частка студентів, у яких компетентності сформувались на середньому або високому рівнях, у ЕГ не більше, ніж у КГ.

H_1: Частка студентів, у яких компетентності сформувались на середньому або високому рівнях, у ЕГ більше, ніж у КГ.

З таблиці XII [253, 330] визначимо значення кутів для кожної з груп: $\varphi_1(73{,}1\%)=2{,}051$; $\varphi_2(39{,}6\%)=1{,}361$.

Розрахуємо емпіричне значення $\varphi^*_{емп}$ за формулою

$$\varphi^*_{\text{емп}}=(\varphi_1-\varphi_2)\sqrt{\frac{n_1 n_2}{n_1+n_2}},$$

де n_1=53 – кількість спостережень у КГ, n_2=52 – кількість спостережень у ЕГ. Отримаємо $\varphi^*_{емп}$=3,54.

Побудуємо вісь значущості для отриманих даних (рис. 2.31).

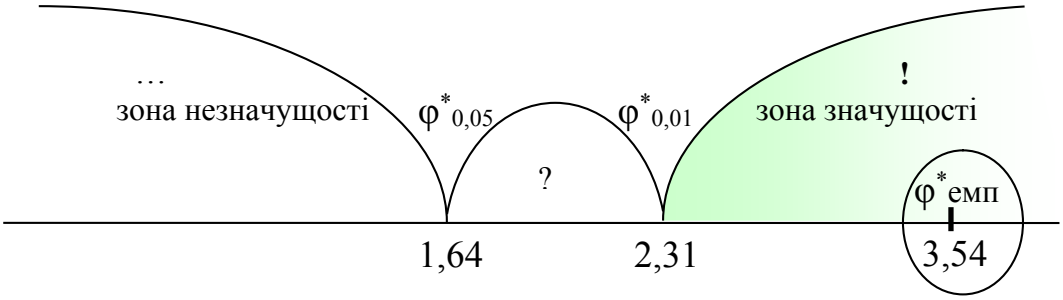


Рис. 2.31. Вісь значущості для кутового перетворення Фішера
(після формувального етапу педагогічного експерименту)

З рис. 2.31 маємо, що $\varphi^*_{емп} > \varphi^*_{0,01}$, що дає нам підставу для відхилення H_0 і прийняття альтернативної гіпотези H_1. Таким чином, КГ та ЕГ мають статистично значущі відмінності у рівнях сформованості компетентностей з програмування, що є результатом використання запропонованої методичної системи.

Підводячи підсумок, приходимо до висновку, що педагогічний експеримент підтвердив гіпотезу нашого дослідження. Аналіз його результатів свідчить про підвищення рівня сформованості компетентностей з програмування при використанні розробленої методичної системи, а, отже, і про її ефективність.

Висновки до розділу 2

1. Реалізацією функціонального підходу до програмування є функціональні мови програмування, які на початку навчання програмування виступають в якості об’єкта вивчення, а далі – в якості засобу навчання. З метою скорочення терміну початкового опанування мови програмування доцільним є вибір синтаксично компактної, розширюваної мови програмування, такої як Scheme. Додатковими її перевагами є: стисліcть, модульність, використання функцій як значень, невеликий розмір ядра мови, мобільність програм, можливість використання в діалоговому режимі, зручність для розв’язування математичних задач, підтримка багатьох парадигм (функціональної, об’єктно-орієнтованої та імперативної).

2. Метою розробленої методичної системи є формування у студентів педагогічних університетів компетентностей з програмування на основі функціонального підходу. Для досягнення цієї мети було створено навчально-методичний комплекс з курсу «Вступ до програмування», що включає в себе авторський посібник, відеоуроки, середовища програмування DrRacket та Scheme-аплет, а також інструкції щодо роботи з курсом у цілому та його окремими модулями.

3. Зміст навчання, відображений у посібнику [191] та методичних рекомендаціях до його використання, передбачає широке застосування моделей та методів математичної інформатики. Представлені у посібнику [191] проекти ілюструють внутрішньопредметні та міжпредметні зв'язки з різних інформатичних дисциплін та створюють умови для опанування різних підходів в межах єдиного середовища.

4. Засоби формування компетентностей з програмування у студентів педагогічних університетів на основі функціонального підходу діляться на 3 основні групи, що відповідають різним складовим компетентностей з програмування. До першої групи входять доопрацьовані та локалізовані автором середовища програмування, спрямовані на розвиток гносеологічної та праксеологічної складової: DrRacket та Scheme-аплет.

Гносеологічна та праксеологічна складові реалізуються через засоби електронної системи підтримки навчання Moodle. Досягнення цілі розвитку аксіологічної та соціально-комунікативної складових реалізується, насамперед, через залучення студентів до спільної навчальної діяльності, вихід якої за межі аудиторії можливий при застосуванні третьої групи засобів, представлених web-сервісом Dropbox та системою Skype.

5. Перспективним засобом формування компетентностей з програмування у студентів педагогічних університетів на основі функціонального підходу є засоби Web 2.0, особливо – навчальні подкаст. Впровадження цих засобів вимагає застосування насамперед методів мобільного навчання та відповідних форм організації навчання.

6. Результати педагогічного експерименту підтверджують ефективність розробленої методичної системи формування у студентів напряму підготовки «Ін-

форматика*» педагогічних університетів компетентностей з програмування на основі функціонального підходу.

Основні результати другого розділу опубліковано в роботах [168; 170; 171; 172; 179; 180; 181; 182; 183; 187; 188; 191; 192; 195].

ВИСНОВКИ

В ході дослідження отримані такі основні *результати*:

1) узагальнений вітчизняний та зарубіжний досвід застосування компетентнісного підходу до підготовки майбутніх учителів інформатики;

2) розроблена структура компетентностей з програмування та показники сформованості їх складових;

3) розроблена методична система формування у студентів педагогічних університетів компетентностей з програмування на основі функціонального підходу;

4) розроблено програмно-методичне забезпечення курсу «Вступ до програмування» для студентів напряму підготовки «Інформатика*» педагогічних університетів з використанням мов функціонального програмування;

5) розроблені окремі компоненти методичної системи навчання об'єктно-орієнтованого програмування та систем штучного інтелекту на основі функціонального підходу;

6) експериментально перевірена ефективність методичної системи формування у студентів напряму підготовки «Інформатика*» педагогічних університетів компетентностей з програмування на основі функціонального підходу.

Результати проведеного дослідження дають підстави зробити такі *висновки*:

1. Сучасна освітня парадигма передбачає розробку методичних систем навчання всіх дисциплін на основі компетентнісного підходу, впровадження якого у процес навчання надає можливість його гуманізувати, підвищити професійну мобільність та створити умови для включення особистості в систему неперервної освіти. Для досягнення цієї мети мають бути сформовані такі складові компетентностей, як гносеологічна, праксеологічна, аксіологічна та соціально-комунікативна.

2. В структурі спеціальних професійних компетентностей учителя інформатики одними з найбільш значущих є компетентності з програмування, формування яких можливе на основі різних підходів. Формування у студентів педагогічних університетів компетентностей з програмування на основі функціонального підходу надає можливості фундаменталізації процесу навчання на основі широкого

застосування моделей та методів математичної інформатики.

3. Реалізацією функціонального підходу до програмування є функціональні мови програмування, які на початку навчання програмування виступають в якості об'єкта вивчення, а далі – в якості засобу навчання. З метою скорочення терміну початкового опанування мови програмування доцільним є вибір синтаксично компактної, розширюваної мови програмування Scheme.

4. Упровадження компетентнісного підходу вимагає зміни всіх компонентів методичної системи навчання програмування: цілей (результатом навчання стають компетентності з програмування), змісту (через добір матеріалу, який сприятиме формуванню не лише гносеологічного та праксеологічного компонентів компетентностей, але й аксіологічного) та технології навчання (особливістю компетентнісного підходу є особлива увага до методів, засобів та форм активного навчання).

5. Метою розробленої методичної системи є формування у студентів педагогічних університетів компетентностей з програмування на основі функціонального підходу. Для досягнення цієї мети було створено комп'ютерно-орієнтований навчально-методичний комплекс з курсу «Вступ до програмування» для студентів напряму підготовки «Інформатика*» педагогічних університетів, що включає в себе авторський посібник, відеоуроки, середовища програмування DrRacket та Scheme-аплет, а також інструкції щодо роботи з курсом в цілому та його окремими модулями. Представлені у посібнику [191] проекти ілюструють внутрішньопредметні та міжпредметні зв'язки різних інформатичних дисциплін та створюють умови для опанування різних підходів до програмування в межах єдиного середовища.

6. Упровадження функціонального підходу у процес навчання програмування впливає на методичну систему навчання на всіх її рівнях:

– на рівні цілей навчання – з'являється мета навчання програмування як комп'ютерної інтерпретації λ- і комбінаторної алгебр та фундаментальної основи теоретичної інформатики;

– на рівні змісту навчання – створюються умови для пропедевтики навчання

об’єктно-орієнтованого, подіє-орієнтованого, візуального і мережного програмування та інтелектуальних систем;

– на рівні методів навчання – надає можливість ширше застосовувати методи активного навчання (моделювання, метод проектів);

– на рівні засобів навчання – виникає можливість застосування мобільних програмних середовищ (DrRacket та Scheme-аплет), та засобів організації спільної роботи (Dropbox та Skype);

– на рівні форм організації навчання – впровадження таких прогресивних форм навчання, як проектна та поява нових форм комбінованого навчання.

Результати дослідження можуть бути використані для організації навчання програмування на основі функціонального підходу бакалаврів програмної, комп’ютерної та системної інженерії.

Отримані результати надають можливість вказати *напрями подальших досліджень*:

1) розробка методичних основ навчання об’єктно-орієнтованого, подіє-орієнтованого, логічного програмування у середовищах функціонального програмування;

2) розробка комп’ютерно-орієнтованої системи педагогічної діагностики майбутніх учителів інформатики;

3) дослідження можливостей технологій соціального конструктивізму в процесі формування компетентностей з програмування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Building a Foundation for Tomorrow : Tech Prep Information Technology Skill Standards-Based Curriculum. – Washington : U.S. Department of Education, 1999. – 124 p.
2. Cardone F. Lambda-calculus and combinators in the 20th century / Felice Cardone, J. Roger Hindley // Handbook of the History of Logic, Volume 5 : Logic from Russell to Church ; Dov M. Gabbay and John Woods (eds.). – Amsterdam : Elsevier Co., 2009. – P. 723–817.
3. Competency Standards Modules : ICT competency standards for teacher. – Paris : United Nations Educational, Scientific and Cultural Organization, 2008. – 13 p.
4. Computer Science Curricula 2013 : Strawman Draft (February 2012) / [Mehran Sahami, Steve Roach, Andrea Danyluk et all]. – 2012. – 170 p.
5. Computer Science Curriculum 2008 : An Interim Revision of CS 2001 (December 2008) / [Lillian Cassel, Alan Clements, Gordon Davies et all]. – 2008. – 108 p.
6. Computing Curricula 2001 : Computer science – Final Report (December 15, 2001) / [Carl Chang, Peter J. Denning, James H. Cross II et all]. – 2001. – 240 p.
7. Definition and Selection of Competencies : Theoretical and Conceptual Foundations (DeSeCo) [Electronic resource]. – Mode of access : http://www.deseco.admin.ch/bfs/deseco/en/index/04.parsys.21201. downloadList.54559.DownloadFile.tmp/1999.backgroundnote.pdf
8. EECS Instructional Support Group Home Page [Electronic resource]. – Mode of access : http://inst.eecs.berkeley.edu/
9. Etheridge B. Computer Competencies for All Educators in North Carolina Public Schools. Revised / Bob Etheridge. – North Carolina, 1992. – 24 p.
10. Gomez, S. Scroll to ‘E’ for Education / Gomez, S. // The Times Higher Education Supplement. – 2007. – Vol. 1780. – P. 13.
11. Hutmaker W. Key Competencies for Europe. Report of the Symposium (Berne, Switzerland, March 27-30, 1996). A Secondary Education for Europe Project. –

Berne, 1996. – 72 p.

12. Key competences for lifelong learning in Europe : Frequently asked questions [Electronic resource]. – Mode of access : http://europa.eu/rapid/ pressReleasesAction.do?reference=MEMO/05/416&format=HTML&aged=0&language=EN&guiLanguage=en
13. McCarthy J. Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I / John McCarthy // Communications of the ACM. – 1960. – Volume 3, Issue 4. – P. 184–195.
14. MIT/GNU Scheme – GNU Project – Free Software Foundation (FSF) [Electronic resource]. – Mode of access : http://www.gnu.org/software/mit-scheme/
15. Moodle [Electronic resource]. – Mode of access : http://www.moodle.org.
16. Ohio Information Technology Competency Profile. – Columbus : Ohio State Dept. of Education, 1999. – 275 p.
17. Quality education and competencies for life : Workshop 3 [Electronic resource] / National Institute of Technology (TI), Norway. – 2004. – Mode of access : http://www.ibe.unesco.org/International/ICE47/English/ Organisation/Workshops/Workshop3CompENG.pdf
18. Racket [Electronic resource]. – Mode of access : http://racket-lang.org/
19. Recommendation of the European Parliament and of the Council, of 18 December 2006, on key competences for lifelong learning [Official Journal L 394 of 30.12.2006] [Electronic resource]. – Mode of access : http://europa.eu/legislation_summaries/education_training_youth/lifelong_learning/c11090_en.htm
20. Schools Using Scheme [Electronic resource]. – Mode of access : http://www.schemers.com/schools.html
21. Shaw M. We Can Teach Software Better / Mary Shaw // Computing Research News. – September 1992. – 4(4) – P. 2–4, 12.
22. Sussman J. Instructor's Manual to accompany Structure and Interpretation of Computer Programs / Julie Sussman. – 2nd Edition. – Cambridge : The MIT Press, 1998. – 211 p.

23. Sussman G. J. Scheme : An Interpreter for Extended Lambda Calculus / Gerald Jay Sussman, Guy L. Steele, Jr. // MIT Artificial Intelligence Memo 349. – December 1975. – Cambridge : MIT, 1975.
24. Technology Competencies in Teacher Education. An Evaluation To Guide Implementation of Beginning Teacher Technology Competencies : Research in Teacher Education. – Mankato : Minnesota State Univ., College of Education, 2000. – 82 p.
25. The definition and selection of key competencies. Executive Summary [Electronic resource]. – Mode of access : http://www.oecd.org/dataoecd/47/61/35070367.pdf
26. Times Higher Education-QS World University Rankings 2009 [Electronic resource]. – Mode of access : http://www.timeshighereducation.co.uk/Rankings2009-Top200.html
27. Welcome To Schemers Inc. [Electronic resource]. – Mode of access : http://schemers.com/
28. Zgaga P. Joint Degrees – Problems and Developments [Electronic resource]. – Mode of access : http://www.see-educoop.net/education_in/ pdf/joint-degrees-problems-and-develop-oth-enl-t02.pdf
29. Абельсон Х. Структура и интерпретация компьютерных программ : пер. с англ. / Харольд Абельсон, Джеральд Джей Сассман, Джули Сассман. – М. : Добросвет, 2006. – 608 с.
30. Адольф В. А. Профессиональная компетентность современного учителя / В. А. Адольф ; ред. В. Ф. Бехтерев. – Красноярск : КГПУ, 1998. – 310 с.
31. Акуленко І. А. Аксіологічний компонент методичних компетентностей майбутніх учителів математики [Електронний ресурс] / І. А. Акуленко, Н. А. Тарасенкова // Вісник Черкаського університету. Серія педагогічні науки. – Випуск 139. – Режим доступу : http://www.nbuv.gov.ua/portal/Soc_Gum/Vchu/N139/N139p003-010.pdf
32. Алексеев М. В. Ключевые компетенции в педагогической литературе / М. В. Алексеев // Педагогические технологии. – 2006. – № 3. – С. 3–18.

33. Алексюк А. М. Педагогіка вищої освіти України / А. М. Алексюк – К. : Либідь, 1998. – 557 с.
34. Алексюк А. М. Педагогіка вищої школи. Курс лекцій : модульне навчання : навч. посібник / А. М. Алексюк. – К. : ІСДО, 1993. – 220 с.
35. Алексюк А. М. Удосконалення навчального процесу в середній школі / Алексюк А. М., Кашин С. О. – К. : Вища школа, 1986. – 56 с.
36. Ананьев Б. Г. Человек как предмет познания психологии / Б. Г. Ананьев. – СПб. : Питер, 2001. – 288 с.
37. Андреев А. Л. Компетентностная парадигма в образовании: опыт философско-методологического анализа / А. Л. Андреев // Педагогика. – 2005. – № 4. – С. 19–27.
38. Андрущенко В. П. Модернізація педагогічної освіти відповідно до викликів XXI століття / Віктор Андрущенко, Володимир Бондар // Вища школа. – 2009. – № 4. – С. 17–23.
39. Апатова Н. В. Влияние информационных технологий на содержание и методы обучения в средней школе : дисс. ... доктора пед. наук : 13.00.02 – теория и методика обучения (информатика) / Апатова Наталья Владимировна ; Симферопольский гос. ун-т. – М., 1994. – 354 с.
40. Ахо А. Построение и анализ вычислительных алгоритмов : пер. с англ. / А. Ахо, Дж. Хопкрофт, Дж. Ульман. – М. : Мир, 1979. – 536 с.
41. Ахо А. Структуры данных и алгоритмы : пер. с англ. / Альфред В. Ахо, Джон Э. Хопкрофт, Джеффри Д. Ульман. – М. : Вильямс, 2000. – 384 с.
42. Ачкан В. В. Проблема реалізації компетентнісного підходу при вивченні курсу алгебри та початків аналізу [Електронний ресурс]. – Режим доступу : http://www.bdpu.org/scientific_published/pedagogics_1_2007/10.doc
43. Бабин І. І. Формування системи забезпечення якості вищої освіти [Електронний ресурс] / Іван Бабин // Кредитно-модульна система організації навчального процесу, 1-2 липня 2004 р. – Тернопіль, 2004. – Режим доступу : http://www.tspu.edu.ua/php1/include/resurs/kms/6/
44. Байгужин П. А. Информатизация общего среднего образования : научно-

методическое пособие / Байгужин П. А., Боровская Е. В., Матрос Д. Ш. ; под ред. Д. Ш. Матроса. – М. : Педагогическое общество России, 2004. – 384 с.

45. Байденко В. И. Компетентностный подход к проектированию государственных образовательных стандартов высшего профессионального образования (методологические и методические вопросы) : метод пособие / В. И. Байденко. – Изд. 2-е. – М. : Исслед. центр проблем качества подготовки специалистов, 2005. – 114 с.
46. Бек К. Экстремальное программирование. Библиотека программиста / Бек К. – СПб. : Питер, 2002. – 224 с.
47. Бен-Ари М. Языки программирования. Практический сравнительный анализ / М. Бен-Ари. – М. : Мир, 2000. – 366 с.
48. Беспалько В. П. Элементы теории управления процессом обучения. Часть I. (Описание целей и способы их достижения в обучении) (Материалы лекций, прочитанных в Политехническом музее на факультете программированного обучения) / В. П. Беспалько. – М. : Знание. – 1970. – 79 с.
49. Биков В. Ю. Моніторинг рівня навчальних досягнень з використанням Інтернет-технологій : [монографія] / Биков В. Ю. , Богачков Ю. М., Жук Ю. О. ; АПН України ; Інститут інформаційних технологій і засобів навчання. – К. : Педагогічна думка, 2008. – 127 с.
50. Бібік Н. М. Компетентнісний підхід до презентації освітніх результатів / Бібік Н. М. // Школа першого ступеня: теорія і практика : збірник наукових праць Переяслав-Хмельницького державного педагогічного університету ім. Григорія Сковороди. – Переяслав-Хмельницький, 2004. – Вип. 10. – С. 18–26.
51. Білодід І. К. Словник української мови. Т. 4. / Голова ред. кол. І. К. Білодід : Наукова думка, 1973. – 840 с.
52. Білоусова Л. І. Інформатика 10-11 : навчальний посібник / Білоусова Л. І., Муравка А. С., Олефіренко Н. В. – Харків : Фоліо, 2007. – 558 с.
53. Болотов В. А. Компетентностная модель: от идеи к образовательной про-

грамме / В. А. Болотов, В. В. Сериков // Педагогика. – 2003. – № 10. – С. 8–14.

54. Бондар С. П. Термінологічний аналіз понять «компетенція» і «компетентність» у педагогіці: сутність та структура / Світлана Бондар // Освіта і управління. – 2007. – Т. 10. – № 2. – С. 93–99.

55. Бондаревская Е. В. Парадигмальный подход к разработке содержания ключевых педагогических компетенций / Е. В. Бондаревская, С. В. Кульневич // Педагогика. – 2004. – № 10. – С. 23–31.

56. Бочкин А. И. Методика преподавания информатики / А. И. Бочкин. – Минск : Вышейшая школа, 1998. – 431 с.

57. Вегнер Е. Г. Формирование методологической компетентности будущего учителя географии средствами модульного обучения : автореф. дисс. на соискание ученой степени канд. пед. наук : 13.00.08 – теория и методика профессионального образования, 13.00.02 – теория и методика обучения и воспитания (география) / Вегнер Елена Григорьевна ; Ин-т содержания и методов обучения РАО. – М., 2007. – 23 с.

58. Великий тлумачний словник сучасної української мови / [Уклад. і голов. ред. В. Т. Бусел]. – К., Ірпінь : Перун, 2004. – 1440 с.

59. Верлань А. Ф. Информатика и ЭВМ / А. Ф. Верлань, В. П. Широчин. – К. : Техніка, 1987. – 342 с.

60. Вирт Н. Алгоритмы + структуры данных = программы : пер. с англ. / Н. Вирт. – М. : Мир, 1985. – 406 с.

61. Вікова та педагогічна психологія : навч. посіб. / О. В. Скрипченко, Л. В. Долинська, З. В. Огороднійчук та ін. – К. : Просвіта, 2001. – 416 с.

62. Вінник Н. Д. Проблематика концепцій ключових кваліфікацій і компетенцій у професійній освіті / Наталія Вінник // Соціальна психологія. – 2008. – № 1. – 149–157.

63. Волкова В. В. Формування професійної спрямованості студентів-менеджерів на початковому етапі навчання (на матеріалі англійської мови) : дис. …канд. пед. наук : 13.00.01 – загальна педагогіка та історія педагогіки /

Волкова Валерія Володимирівна ; Луганський державний педагогічний університет ім. Тараса Шевченка. – Луганськ, 2000. – 205 с.

64. Воронин Ю. А. Компьютеризированные технологии в процессе подготовки учителя / Воронин Ю. А. // Педагогика. – 2003. – №8. – С. 53–59.
65. Воронцов Г. Д. Информологический подход к развитию профессиональной компетентности педагога в процессе постдипломного образования : автореф. дисс. на соискание научной степени канд. пед. наук : 13.00.08 – теория и методика профессионального образования (педагогические науки) / Воронцов Георгий Дмитриевич ; ГОУ ДПО «Санкт-Петербургская академия постдипломного педагогического образования». – СПб., 2006. – 20 с.
66. Габрусєв В. Ю. Комп'ютерно-орієнтовані засоби управління навчальними ресурсами. MOODLE (модульна, об'єктно-орієнтована, динамічна навчальна система) / В. Ю. Габрусєв // Науковий часопис НПУ ім. М. П. Драгоманова. Серія №2. Комп'ютерно-орієнтовані системи навчання : зб. наукових праць / Редрада. – К. : НПУ ім. М. П. Драгоманова, 2006. – №6 (11). – С. 24–28.
67. Газейкина А. И. Стили мышления и обучение программированию студентов педагогического вуза : Материалы конференции «Информационные технологии в образовании – 2006» [Электронный ресурс] / Газейкина Анна Ивановна / Режим доступа : http://ito.edu.ru/2006/Moscow/I/1/I-1-6371.html
68. Галузевий стандарт вищої освіти України. Освітньо-кваліфікаційна характеристика : [освітньо-кваліфікаційний рівень] бакалавр. Галузь знань 0403 «Системні науки та кібернетика». Напрям підготовки 040302 «Інформатика». Кваліфікація 3121 «Фахівець з інформаційних технологій». 3340 «Викладач-стажист» / Міністерство освіти і науки України. – К., 2010. – 32 с.
69. Галузевий стандарт вищої освіти України. Освітньо-професійна програма підготовки : [освітньо-кваліфікаційний рівень] бакалавр. Галузь знань 0403 «Системні науки та кібернетика». Напрям підготовки 040302 «Інформатика». Кваліфікація 3121 «Фахівець з інформаційних технологій». 3340 «Ви-

кладач-стажист» / Міністерство освіти і науки України. – К., 2010. – 94 с.

70. Гласс Д. Статистические методы в педагогике и психологии : пер. с англ. / Гласс Д., Стэнли Д. – М. : Прогресс, 1971. – 495 с.
71. Головань М. С. Інформатична компетентність як об'єкт педагогічного дослідження / М. С. Головань // Проблеми інженерно-педагогічної освіти : збірник наукових праць. – Харків : УІПА, 2007. – № 16. – С. 314–324.
72. Головань М. С. Інформатична компетентність: сутність, структура та становлення / М. С. Головань // Інформатика та інформаційні технології в навчальних закладах. – 2007. – № 4. – С. 62–69.
73. Головань М. С. Компетентнісний підхід у навчанні інформатики і комп'ютерної техніки студентів економічного ВНЗ / М. С. Головань // Проблеми інженерно-педагогічної освіти : зб. наук. праць / Українська інженерно-педагогічна академія. – Харків : УІПА, 2007. – Вип. 18–19. – С. 19–32.
74. Головань М. С. Компетенція і компетентність : досвід теорії, теорія досвіду / Микола Головань // Вища освіта України. – 2008. – № 3. – С. 23–31.
75. Головань М. С. Модель процесу розвитку інформатичної компетентності студентів економічного вузу / М. С. Головань // Збірник наукових праць Кам'янець-Подільського національного університету. Серія педагогічна / Кам'янець-Подільський національний ун-т. – Кам'янець-Подільський, 2008. – Вип. 14. – С. 17–20.
76. Головань М. С. Розвиток інформатичної компетентності студентів як педагогічної системи / М. С. Головань // Педагогічні науки : зб. наук. праць / Сумський державний педагогічний ун-т ім. А. С. Макаренка. – Суми, 2008. – С. 88–96.
77. Головлева С. В. Методика обучения функциональному программированию будущих учителей информатики (на базе языка LOGO) : диссертация ... кандидата педагогических наук : 13.00.02 – теория и методика обучения информатике / Головлева Светлана Викторовна ; Российский государственный педагогический университет им. А. И. Герцена. – Санкт-Петербург, 2000. – 205 с.

78. Гончаренко С. У. Український педагогічний словник / Семен Гончаренко. – К. : Либідь, 1997. – 376 с.

79. Гончарова О. М. Теоретико-методичні основи особистісно-орієнтованої системи формування інформатичних компетентностей студентів економічних спеціальностей : автореф. дис. на здобуття наук. ступеня доктора пед. наук : 13.00.02 – теорія і методика навчання (інформатика) / Гончарова Оксана Миколаївна ; Національний педагогічний ун-т ім. М. П. Драгоманова. – К., 2007. – 40 с.

80. Горошко Ю. В. Система інформаційного моделювання у підготовці майбутніх учителів математики та інформатики : дис. … доктора пед. наук : 13.00.02 – теорія та методика навчання (інформатика) / Горошко Юрій Васильович ; Чернігівський національний педагогічний університет ім. Т. Г. Шевченка. – Чернігів, – 2013. – 470 с.

81. Грабарь М. И. Применение математической статистики в педагогических исследованиях : Непараметрические методы / М. И. Грабарь, К. А. Краснянская. – М. : Просвещение, 1977. – 136 с.

82. Грабовський П. П. Інформаційна компетентність учителя середньої школи / П. П. Грабовський // Вісник Житомирського державного університету ім. Івана Франка. – Житомир, 2008. – № 37. – С. 118–123.

83. Гриньова В. М. Професійна компетентність викладача / В. М. Гриньова // Вісник Дніпропетровського університету економіки та права ім. Альфреда Нобеля. Серія «Педагогіка і психологія». – Дніпропетровськ, 2011. – № 1 (1). – С. 21–26.

84. Гришко Л. В. Методична система навчання основ програмування майбутніх інженерів-програмістів : дис. … канд. пед. наук : 13.00.02 – теорія і методика навчання (інформатика) / Гришко Людмила Веніамінівна ; Національний педагогічний ун-т ім. М. П. Драгоманова. – 2009. – 276 с.

85. Гришко Л. В. Функції курсу основ програмування для майбутніх програмістів та принципи навчання / Л. В. Гришко // Теорія та методика навчання математики, фізики, інформатики : збірник наукових праць. – Випуск Х : в 3-х

томах. – Кривий Ріг : Видавничий відділ НМетАУ, 2012. – Т. 3 : Теорія та методика навчання інформатики. – С. 40–48.

86. Гуржій А. М. Інформатика та інформаційні технології : підручник / А. М. Гуржій, Н. І. Поворознюк, В. В. Самсонов. – Х. : СМІТ, 2007. – 352 с.
87. Дахин А. Н. Компетенция и компетентность: сколько их у российского школьника / Дахин А. Н. // Народное образование. – 2004. – №4. – С. 136–144.
88. Дейкстра Э. Дисциплина программирования : пер. с англ. / Э. Дейкстра. – М. : Мир, 1978. – 274 с.
89. Дергач М. А. Дидактичні умови застосування гіпертекстових програм у процесі вивчення гуманітарних дисциплін (на матеріалі історії музики) : дис. ... канд. пед. наук : 13.00.01 – загальна педагогіка та історія педагогіки / Дергач Маргарита Альфритівна ; Київський університет ім. Тараса Шевченка. – К., 1998. – 186 с.
90. Державна національна програма «Освіта. Україна XXI століття». – К. : Райдуга, 1994. – 61 с.
91. Дружилов С. А. Обучение и стадии профессиональной компетентности / С. А. Дружилов // Непрерывное образование как условие развития творческой личности : сб. по материалам Фестиваля педагогического творчества, 28–29 авг. 2000 г. – Новокузнецк : ИПК, 2001. – С. 32–33.
92. Дружилов С. А. Профессиональная компетентность и профессионализм педагога: психологический подход / С. А. Дружилов // Сибирь. Философия. Образование : научно-публицистический альманах. – Новокузнецк : СО РАО, 2005. – Выпуск 8. – С. 26–44.
93. Економічна енциклопедія : у трьох томах. Т. 1 / Редкол. : С. В. Мочерний (відп. ред.) та ін. – К. : Академія, 2000. – 864 с.
94. Ершов А. П. Компьютеризация школы и математическое образование / Ершов А. П. // Информатика и образование. – 1992. – № 5–6. – С. 3–20.
95. Жалдак М. И. О некоторых методических аспектах обучения информатике в школе и педагогическом университете / Жалдак М. И. // Методология и

технология образования в XXI веке: математика, информатика, физика : материалы международной научно-практической конференции 17–18 ноября 2005 г. – Минск : Министерство образования республики Беларусь ; Учреждение образования «Белорусский государственный педагогический университет имени Максима Танка», 2006. – С. 260–268.

96. Жалдак М. И. Система подготовки учителя к использованию информационной технологии в учебном процессе : дисс. ... в форме науч. доклада доктора пед. наук : 13.00.02 – теория и методика обучения информатике / Жалдак М. И. ; АПН СССР ; НИИ содержания и методов обучения. – М., 1989. – 48 с.
97. Жалдак М. І. Інформатика : [навч. посібник ; за ред. М. І. Шкіля] / М. І. Жалдак, Ю. С. Рамський. – К. : Вища шк., 1991. – 319 с.
98. Жалдак М. І. Комп'ютерно-орієнтовані засоби навчання математики, фізики, інформатики / Жалдак М. І, Лапінський В. В., Шут М. І. // Інформатика. – 2006. – №3–4. – С. 3–96.
99. Жалдак М. І. Методика вивчення основ інформатики та обчислювальної техніки в педагогічному вузі : [учбовий посібник] / Жалдак М. І. ; КДПІ ім. О. М. Горького. – К., 1986. – 74 с.
100. Жалдак М. І. Модель системи соціально-професійних компетентностей вчителя інформатики / Жалдак М. І., Рамський Ю. С., Рафальська М. В. // Науковий часопис НПУ ім. М. П. Драгоманова. Серія №2. Комп'ютерно-орієнтовані системи навчання : зб. наукових праць / Редрада. – К. : НПУ ім. М. П. Драгоманова, 2006. – №7 (14). – С. 3–10.
101. Жалдак М. І. Обчислювальна математика : [спец. курс факультативних занять у 9-х і 10-х кл.] / Жалдак М. І., Ковбасенко Б. С., Рамський Ю. С. – К. : Рад. школа, 1973. – 184 с.
102. Жалдак М. І. Основи інформатики та обчислювальної техніки : програма для середніх закладів освіти / М. І. Жалдак, Н. В. Морзе, Г. Г. Науменко ; Головне управління загальної середньої освіти. – К. : Перун, 1996. – 24 с.
103. Жалдак М. І. Основи теорії і методів оптимізації : [навч. посіб. для студ.

мат. спец. вищ. навч. закл.] / Жалдак М. І., Триус Ю. В. – Черкаси : Брама-Україна, 2005. – 607 с.

104. Жалдак М. І. Про проблеми навчання інформатики в середніх та вищих навчальних закладах / Жалдак М. І. // Актуальні проблеми психології : Психологічна теорія і технологія навчання ; [за ред. С. Д. Максименка, М. Л. Смульсон]. – К. : Міленіум, 2005. – Т. 8, вип. 1. – С. 39–53.
105. Жалдак М. І. Програма для загальноосвітніх навчальних закладів «Інформатика 10-11 класи» / Жалдак М. І., Морзе Н. В., Науменко Г. Г., Мостіпан О. І. – Кам'янець-Подільській : Абетка-НОВА, 2002. – 80 с.
106. Жалдак М. І. Чисельні методи математики : [посібник для самоосвіти вчителів] / Жалдак М. І., Рамський Ю. С. – К. : Рад. школа, 1984. – 206 с.
107. Жужжалов В. Е. Интеграция парадигм программирования в курсе информатики / Жужжалов В. Е. // Информатика и образование. – 2004. – № 10. – С. 32–35.
108. Жукова В. М. Формування інформатичної компетентності майбутнього вчителя математики в процесі професійної підготовки : автореф. дис. на здобуття наук. ступеня канд. пед. наук : 13.00.04 – теорія і методика професійної освіти / Жукова Вікторія Миколаївна ; Луганський національний ун-т ім. Тараса Шевченка. – Луганськ, 2009. – 20 с.
109. Завьялов А. Н. Формирование информационной компетентности у будущих специалистов в области новых информационных технологий [Электронный ресурс] / Завьялов Андрей Николаевич // Информационные технологии в образовании (ИТО–2003). – М. : Государственный научно-исследовательский институт информационных технологий и телекоммуникаций, 2003. – Режим доступа : http://ito.edu.ru/2003/I/1/I-1-1473.html
110. Загальна психологія : підручник / О. В. Скрипченко, Л. В. Долинська, З. В. Огороднійчук та ін. – К. : Либідь, 2005. – 464 с.
111. Закон України «Про вищу освіту» : за станом на 17.01.2002 р. № 2984-III // Відомості Верховної Ради України від 17.05.2002. – 2002, № 20, стаття 134.
112. Закон України «Про національну програму інформатизації» : за станом на 4

лют. 1998 р. № 74/98-ВР // Відомості Верховної Ради України від 17.07.1998. – 1998, № 27, стаття 181.

113. Закон України «Про освіту» : за станом на 23 трав. 1991 р. № 1060–XII // Голос України від 26.06.1991.
114. Зеер Э. Ф. Компетентностный подход к модернизации профессионального образования / Э. Зеер, Э. Сыманюк // Высшее образование в России. – 2005. – № 4. – С. 23–30.
115. Зимняя И. А. Ключевые компетентности как результативно-целевая основа компетентностного подхода в образовании. Авторская версия / И. А. Зимняя. – М. : Исследовательский центр проблем качества подготовки специалистов. – 2004. – 40 с.
116. Зимняя И. А. Общая культура и социально-профессиональная компетентность человека / И. А. Зимняя // Высшее образование сегодня. – 2005. – № 11. – С. 14–20.
117. Зінченко В. О. Формування професійної спрямованості студентів економічних спеціальностей на початковому етапі навчання : дис. ... канд. пед. наук : 13.00.04 – теорія і методика професійної освіти / Зінченко Вікторія Олегівна ; Луганський національний педагогічний ун-т ім. Тараса Шевченка. – Луганськ, 2007. – 216 с.
118. Информатика : энциклопедический словарь для начинающих / Сост. Д. А. Поспелов. – М. : Педагогика-Пресс, 1994. −352 с.
119. Ільченко Р. В. Вплив інформаційних технологій на навчальний процес як психолого-педагогічна проблема [Електронний ресурс] / Р. В. Ільченко, Я. М. Рудик // Науковий вісник Національного університету біоресурсів і природокористування України. Серія : Педагогіка, психологія, філософія : зб. наукових праць / К. : Національний університет біоресурсів і природокористування України, 2011. – №159. Частина 1. – Режим доступу : http://www.nbuv.gov.ua/portal/soc_gum/nvnau_ppf/2011_159_1/11irv.pdf
120. Карпенко М. А. Формування інформатичних компетентностей студентів машинобудівних спеціальностей вищих навчальних закладів / Карпен-

ко М. А. // Науковий часопис НПУ ім. М. П. Драгоманова. Серія №2. Комп'ютерно-орієнтовані системи навчання : зб. наукових праць / Редрада. – К. : НПУ ім. М.П. Драгоманова, 2006. – №7 (14). – С. 170–175.

121. Карпова Л. Г. Формування професійної компетентності вчителя загальноосвітньої школи : дис. ... канд. пед. наук : 13.00.04 – теорія і методика професійної освіти / Карпова Лариса Георгіївна ; Харківський державний педагогічний ун-т ім. Г. С. Сковороди. – Харків, 2003. – 209 с.
122. Керниган Б. В. Практика программирования : пер. с англ. / Брайан Керниган, Роб Пайк. – М. : Вильямс, 2004. – 288 с.
123. Кинелев В. Г. Образование и цивилизация / Кинелев В. Г. // Информатика и образование. – 1996. – № 5. – С. 21–28.
124. Кириллов А. Г. Формирование профессиональных компетенций будущего учителя информатики в процессе обучения программированию : дисс. … канд. пед. наук : 13.00.02 – теория и методика обучения и воспитания (информатика, уровень профессионального образования) / Кириллов Андрей Григорьевич ; ГОУ ВПО «Шадринский государственный педагогический институт. – Шадринск, 2005. – 151 с.
125. Кобильник Т. П. Методична система навчання математичної інформатики у педагогічному університеті : дис. ... канд. пед. наук : 13.00.02 – теорія та методика навчання (інформатика) / Кобильник Тарас Петрович ; Національний педагогічний ун-т ім. М. П. Драгоманова. – К., 2009. – 256 с.
126. Ковальська К. Р. Основи компетентнісного підходу в підготовці вчителя інформатики [Електронний ресурс] / Ковальська Катерина Ростиславівна // Інформаційні технології і засоби навчання. – 2008. – №3. – Режим доступу : http://www.nbuv.gov.ua/e-journals/ITZN/em7/content/08kkrtts.htm
127. Козырева О. А. Профессиональная педагогическая компетентность учителя : феноменология понятия / О. А. Козырева // Вестник ТГПУ. – 2009. – Выпуск 2 (80). – С. 17–23.
128. Колос К. Р. Система Moodle як засіб розвитку предметних компетентностей учителів інформатики в умовах дистанційної післядипломної освіти : дис. ...

канд. пед. наук : 13.00.10 – інформаційно-комунікаційні технології в освіті / Колос Катерина Ростиславівна ; Житомирський державний університет ім. Івана Франка. – Житомир, 2011. – 238 с.

129. 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.
130. Компьютерная технология обучения : словарь-справочник / Под ред. Гриценко В. И., Довгялло А. М., Савельева А. Я. – К. : Наукова думка, 1992. – 640 с.
131. Косова И. С. Использование языка LISP при обучении функциональному программированию будущих учителей математики и информатики : автореф. дисс. на соискание ученой степени канд. пед. наук : 13.00.02 – теория и методика обучения и воспитания (информатика, уровни общего и профессионального образования) / Косова Ирина Святославовна ; Российский государственный педагогический ун-т им. А. И. Герцена. – СПб., 2001. – 18 с.
132. Костиков А. Н. Методика обучения компьютерной графике будущих учителей информатики на основе компетентностного подхода : автореф. дисс. на соискание ученой степени канд. пед. наук : 13.00.02 – теория и методика обучения и воспитания (информатика, уровень профессионального образования) / Костиков А. Н. ; Российский государственный педагогический ун-т им. А. И. Герцена. – СПб., 2003. – 16 с.
133. Краевский В. В. Предметное и общепредметное в образовательных стандартах / Краевский В. В., Хуторской А. В. // Педагогика. – 2003. – № 2. – С. 3–10.
134. Красюк Ю. М. Мотиваційні аспекти використання НІТН у процесі навчання інформатики у вищих закладах освіти / Ю. М. Красюк // Науковий часопис НПУ ім. М. П. Драгоманова. Серія № 2. Комп’ютерно-орієнтовані системи навчання : зб. наукових праць / Редрада. – К. : НПУ ім. М. П. Драгоманова, 2002. – № 5. – С. 181–187.
135. Кузнецов А. А. Развитие методической системы обучения информатике в

средней школе : дисс. ... в форме науч. доклада доктора пед. наук : 13.00.02 – теория и методика обучения информатике / Кузнецов А. А. ; НИИСиМО АПН СССР. – М., 1988. – 47 с.

136. Кузнецов Э. И. Общеобразовательные и профессионально-прикладные аспекты изучения информатики и вычислительной техники в педагогическом институте : автореф. дисс. на соискание ученой степени доктора пед. наук : 13.00.02 – теория и методика обучения (информатика) / Кузнецов Эдуард Иванович ; Мос. пед. ун-т. – М., 1990. – 38 с.
137. Кузьмина Н. В. Профессионализм деятельности преподавателя и мастера производственного обучения профтехучилища / Н. В. Кузьмина. – М. : Высшая школа, 1989. – 166 с.
138. Кузьмінський А. І. Наукові засади методичної підготовки майбутнього вчителя математики : [монографія] / А. І. Кузьмінський, Н. А. Тарасенкова, І. А. Акуленко. – Черкаси : Вид. від. ЧНУ ім. Б. Хмельницького, 2009. – 320 с.
139. Кузьмінський А. І. Педагогіка вищої школи : навч. посіб. для студ. вищ. навч. закладів / А. І. Кузьмінський. – К. : Знання-Прес, 2005. – 485 с. – (Вища освіта XXI століття).
140. Кулагина И. Ю. Возрастная психология: Полный жизненный цикл развития человека : учебное пособие для студентов высших учебных заведений / И. Ю. Кулагина, В. Н. Колюцкий. – М. : Сфера, 2001. – 464 с.
141. Кучай О. В. Обґрунтування основних понять категорії «Професійна компетенція вчителя інформатики» / Кучай О. В. // Вісник Черкаського університету. Серія педагогічні науки. – Випуск 142. – Черкаси : Вид. від. ЧНУ ім. Б. Хмельницького, 2008. – С. 78–82.
142. Лаптев В. В. Методическая теория обучения информатике. Аспекты фундаментальной подготовки / В. В. Лаптев, Н. И. Рыжова, М. В Швецкий. – СПб. : Изд-во С.-Петерб. ун-та, 2003. – 352 с.
143. Лапчик М. П. Структура и методическая система подготовки кадров информатизации школы в педагогических вузах : дисс. на соискание ученой

степени доктора пед. наук в форме научн. докл. : 13.00.02 – теория и методика обучения (информатика) / Лапчик Михаил Павлович. – М., 1999. – 82 с.

144. Лапчик М. П. Теория и методика обучения информатике : учебник / М. П. Лапчик, И. Г. Семакин, Е. К. Хеннер ; под общей ред. М. П. Лапчика. – М. : Академия, 2008. – 592 с.
145. Ларионова О. Г. Интеграция личностно-центрированного и компетентностного подходов в контекстном обучении (на материале подготовки учителя математики) : автореф. дисс. на соискание ученой степени доктора пед. наук : 13.00.01 – общая педагогика, история педагогики и образования / Ларионова Ольга Гавриловна ; Ин-т содержания и методов обучения РАО. – М., 2007. – 54 с.
146. Лебедев О. Е. Компетентностный подход в образовании / Лебедев О. Е. // Школьные технологии. – 2004. – № 5. – С. 3–12.
147. Лебедева О. В. Развитие методической компетентности учителя как средство повышения эффективности учебного процесса в общеобразовательной школе : автореф. дисс. на соискание ученой степени канд. пед. наук : 13.00.01 – общая педагогика, история педагогики и образования / Лебедева Ольга Васильевна ; ГОУ ВПО «Нижегородский гос. ун-т им. Н. И. Лобачевского». – Нижний Новгород, 2007. – 24 с.
148. Лебедева Т. Н. Формирование информационной компетентности учащихся посредством изучения рекурсивных алгоритмов и функций / Лебедева Т. Н. // Образовательные технологии. – 2005. – № 1 (14). – С. 18–21.
149. Лебеденко Ю. М. Компетентнісний підхід в системі вищої освіти / Ю. М. Лебеденко // Гуманізм та освіта : зб. матеріалів VIII міжнар. наук.-практ. конф., м. Вінниця, 19-21 верес. 2006 р. / Вінниц. нац. техн. ун-т, Мінво освіти і науки України, АПН України, Вінниц. нац. техн. ун-т, Ун-т ЄВЛЄ (Швеція) ; [редкол. : Мокін Б. І., Буяльська Т. Б., Азарова Л. Є. та ін.]. – [Вінниця] : УНІВЕРСУМ-Вінниця, 2006. – С. 93–95.
150. Левченко И. В. Развитие системы методической подготовки учителей ин-

форматики в условиях фундаментализации образования : автореф. дисс. на соискание ученой степени доктора пед. наук : 13.00.02 – теория и методика обучения и воспитания (информатика) / Левченко Ирина Витальевна ; Тульский государственный педагогический университет им. Л. Н. Толстого. – М., 2009. – 46 с.

151. Леонтьев А. Н. Деятельность. Сознание. Личность : учеб. пособие для студентов вузов по направлению и специальностям «Психология», «Клин. психология» / А. Н. Леонтьев. – М. : Смысл ; Academia, 2004. – 345 с.
152. Леонтьев А. Н. Проблемы развития психики / А. Н. Леонтьев – М. : Изд-во Моск. ун-та, 1981. – 584 с.
153. Лернер И. Я. Дидактические основы методов обучения / Лернер И. Я. – М. : Педагогика, 1981. – 185 с.
154. Литвинова С. Г. Формування інформаційно-комунікаційної компетентності (ІКК) вчителів-предметників [Електронний ресурс] / Литвинова Світлана Григорівна // Інформаційні технології і засоби навчання. – 2008. – №3. – Режим доступу : http://www.ime.edu-ua.net/em5/content/08lsgtso.htm.
155. Макконнелл С. Совершенный код. : пер. с англ. / Стив Макконнелл. – М. : Русская редакция ; СПб. : Питер, 2005. – 867 с.
156. Маклаков Г. Ю. Особенности использования технологии IP-телефонии для совершенствования процесса дистанционного обучения / Г. Ю. Маклаков, Г. Г. Маклакова // Матеріали V Міжнародної науково-технічної конференції «Комп'ютерні технології в будівництві» : Київ–Севастополь, 18-21 вересня 2007 р. – Кривий Ріг, 2007. – С. 71–72.
157. Малев В. В. Общая методика преподавания информатики : учебное пособие / В. В. Малев. – Воронеж : ВГПУ, 2005. – 271 с.
158. Малярчук С. М. Формування основ інформаційної культури учнів 6-7 класів з допомогою системи ЛОГО : автореф. дис. на здобуття наук. ступеня канд. пед. наук : 13.00.02 – теорія та методика навчання інформатики / Малярчук Сергій Миколайович ; Український держ. педагогічний ун-т ім. М. П. Драгоманова. – К., 1997. – 24 с.

159. Маркова А. К. Психологический анализ профессиональной компетентности учителя / А. К. Маркова // Советская педагогика. – 1990. – № 8. – С. 82–88.

160. Машбиц Е. И. Введение в язык Лого : учебник / Е. И. Машбиц ; ред. А. А. Стогний. – К. : Выща шк., 1989. – 207 с.

161. Машбиц Е. И. Основы компьютерной грамотности / Е. И. Машбиц, Л. П. Бабенко, Л. В. Верник и др. – К. : Выща шк., 1988. – 215 с.

162. Машбиц Е. И. Психологические основы управления учебной деятельностью / Машбиц Е. И. – К. : Выща школа, 1987. – 224 с.

163. Машбиц Е. И. Психолого-педагогические проблемы компьютеризации обучения / Е. И. Машбиц. – М. : Педагогика, 1988. – 192 с.

164. Меняйленко О. С. Розробка критерію сформованості професійної компетентності фахівців у автоматизованих навчаючих системах [Електронний ресурс] / О. С. Меняйленко, Г. В. Монастирна // Сучасні інформаційні технології та інноваційні методики навчання у підготовці фахівців: методологія, теорія, досвід, проблеми : збірник наукових праць. – 2010. – Випуск 23. – Режим доступу : http://www.nbuv.gov.ua/portal/soc_gum/Sitimn/2010_23/

165. Митина Л. М. Психология профессионального развития учителя / Л. М. Митина. – М. : Академия, 2004. – 320 с.

166. Михалін Г. О. Професійна підготовка вчителя математики у процесі навчання математичного аналізу / Михалін Г. О. – К. : ДІНІТ, 2003. – 320 с.

167. Михалін Г. О. Формування основ професійної культури вчителя математики у процесі навчання математичного аналізу : дис. ... доктора пед. наук : 13.00.04 – теорія і методика професійної освіти / Михалін Геннадій Олександрович ; Національний педагогічний ун-т ім. М. П. Драгоманова. – К., 2004. – 413 с.

168. Мінтій І. С. Dropbox у навчальному процесі: спільне використання та синхронізація файлів / В. С. Мазур, І. С. Мінтій // Матеріали X Міжнародної науково-технічної конференції «Новітні комп'ютерні технології» : Севастополь, 11–14 вересня 2012 р. – К. : Мінрегіон України, 2012. – С. 128–130.

169. Мінтій І. С. Вивчення логічних основ ЕОМ в шкільному курсі інформатики / Л. О. Лісіна, І. С. Мінтій // Теорія та методика навчання математики, фізики, інформатики : збірник наукових праць. – Випуск VI : в 3-х томах. – Кривий Ріг : Видавничий відділ НМетАУ, 2006. – Т. 3 : Теорія та методика навчання інформатики. – С. 320–326.

170. Мінтій І. С. Використання Документів Google як умова оптимізації спільної роботи / І. С. Мінтій // Теорія та методика електронного навчання : збірник наукових праць. Випуск І. – Кривий Ріг : Видавничий відділ НМетАУ, 2010. – С. 150–154.

171. Мінтій І. С. Експериментальне дослідження ефективності формування у студентів педагогічних університетів компетентності з програмування на основі функціонального підходу / Мінтій І. С. // Науковий часопис НПУ ім. М. П. Драгоманова. Серія №2. Комп'ютерно-орієнтовані системи навчання : зб. наукових праць / Редрада. – К. : НПУ ім. М.П. Драгоманова, 2012. – №12 (19). – С. 153–158.

172. Мінтій І. С. Засоби формування у студентів педагогічних університетів компетентності з програмування на основі функціонального підходу / І. С. Мінтій // Вісник Черкаського університету. Серія педагогічні науки. – Випуск 191. Частина І. – Черкаси : Вид. від. ЧНУ ім. Б. Хмельницького, 2010. – С. 86–92.

173. Мінтій І. С. Інформатичні компетентності: аналіз зарубіжного досвіду / І. С. Мінтій // Науковий часопис НПУ ім. М. П. Драгоманова. Серія №2. Комп'ютерно-орієнтовані системи навчання : зб. наукових праць / Редрада. – К. : НПУ ім. М.П. Драгоманова, 2006. – №7 (14). – С. 215–218.

174. Мінтій І. С. Компетентнісний підхід: надбання та шляхи подальшої розробки / І. С. Мінтій, С. О. Семеріков // Молодий науковець XXI століття : матеріали Міжнародної науково-практичної конференції (Кривий Ріг, 17–18 листопада 2008 р.). – Кривий Ріг : Видавничий центр КТУ, 2008. – С. 18–20.

175. Мінтій І. С. Математичне моделювання та прикладні задачі в шкільному курсі математики / Ірина Мінтій, Володимир Петров // Математика в школі.

– 2007. – № 1. – С. 3–8.

176. Мінтій І. С. Математичні основи функціонального підходу / І. С. Мінтій // Розвиток інтелектуальних умінь і творчих здібностей учнів та студентів у процесі навчання математики : матеріали Всеукр. наук.-метод. конф. (3–4 грудня 2009 р., м. Суми). – Суми : Вид-во СумДПУ ім. А. С. Макаренка, 2009. – С. 219–220.

177. Мінтій І. С. Математичні основи функціонального програмування / І. С. Мінтій // Педагогічні науки : теорія, історія, інноваційні технології : науковий журнал. – Суми : Вид-во СумДПУ ім. А. С. Макаренка, 2009. – № 2. – С. 337–345.

178. Мінтій І. С. Методичні засади навчання програмування майбутніх вчителів інформатичних дисциплін на основі функціонального підходу / І. С. Мінтій / Тези доповідей VII Всеукраїнської науково-практичної конференції «Інформаційні технології в освіті, науці і техніці» (ІТОНТ-2010) : Черкаси, 4-6 травня 2010 р. – У 2-х томах. – Черкаси : ЧДТУ, 2010. – Т. 2. – С. 69.

179. Мінтій І. С. Мобільне програмне забезпечення навчання інформатичних дисциплін у вищій школі / Семеріков С. О., Мінтій І. С., Словак К. І. та ін. // Науковий часопис НПУ ім. М. П. Драгоманова. Серія № 2. Комп'ютерно-орієнтовані системи навчання : зб. наукових праць / Редрада. – К. : НПУ ім. М. П. Драгоманова, 2010. – № 8 (15). – С. 18–28.

180. Мінтій І. С. Навчально-методичне забезпечення курсу «Вступ до програмування» / І. С. Мінтій / Матеріали VIII Міжнародної науково-технічної конференції «Новітні комп'ютерні технології» : Київ–Севастополь, 14–17 вересня 2010 р. – К. : Міністерство регіонального розвитку та будівництва України, 2010. – С. 113–114.

181. Мінтій І. С. Принципи проектування та розвитку методичної системи фундаментальної інформатичної підготовки / С. О. Семеріков, О. І. Теплицький, І. С. Мінтій // Збірник наукових праць. – Харків : ХНАДУ, 2010. – С. 32–34.

182. Мінтій І. С. Програмно-методичний комплекс для підтримки курсу «Вступ до програмування» / Ірина Мінтій // Наукові записки. – Випуск 98. – Серія :

педагогічні науки. – Кіровоград : РВВ КДПУ ім. В. Винниченка, 2011. – С. 224–226.

183. Мінтій І. С. Програмно-методичний комплекс для підтримки курсу «Вступ до програмування» / Мінтій І. С // Матеріали Міжнародної VII (XVII) науково-практичної конференції «Засоби і технології сучасного навчального середовища», м. Кіровоград, 20-21 травня 2011 року. – Кіровоград : КОД, 2011. –С. 124–126.
184. Мінтій І. С. Професійні компетентності вчителя інформатики / І. С. Мінтій // Вісник Черкаського університету. Серія педагогічні науки. – Випуск 162. – Черкаси : Вид. від. ЧНУ ім. Б. Хмельницького, 2009. – С. 99–110.
185. Мінтій І. С. Професійні компетенції фахівців у галузі інформаційних технологій / В. М. Соловйов, І. С. Мінтій // Вісник Черкаського університету. Серія педагогічні науки. – Випуск 155. – Черкаси : Вид. від. ЧНУ ім. Б. Хмельницького, 2009. – С. 102–111.
186. Мінтій І. С. Професійні компетенції фахівців у галузі інформаційних технологій / В. М. Соловйов, І. С. Мінтій // Матеріали Міжнародної науково-методичної конференції «Проблеми математичної освіти» (ПМО – 2009), м. Черкаси, 7-9 квітня 2009 р. – Черкаси : Вид. від. ЧНУ ім. Б. Хмельницького, 2009. – С. 194–196.
187. Мінтій І. С. Рівні сформованості компетентності у програмуванні / І. С. Мінтій // Теорія та методика навчання математики, фізики, інформатики : збірник наукових праць. – Випуск Х : в 3-х томах. – Кривий Ріг : Видавничий відділ НМетАУ, 2012. – Т. 3 : Теорія та методика навчання інформатики. – С. 82–86.
188. Мінтій І. С. Розробка фільтру Sage для СДН Moodle / С. О. Семеріков, С. В. Шокалюк, І. С. Мінтій та ін. / Матеріали IX Міжнародної науково-технічної конференції «Новітні комп’ютерні технології» (NOKOTE’2011) : Київ–Севастополь, 13–16 вересня 2011 р. – К. : Мінрегіон України, 2011. – С. 189–194.
189. Мінтій І. С. Спеціальні професійні компетентності вчителя інформатики /

І. С. Мінтій // Інноваційні інформаційно-комунікаційні технології навчання математики, фізики, інформатики у середніх та вищих навчальних закладах : зб. наук. праць за матеріалами Всеукр. наук.-метод. конф. молодих науковців, 17-18 лют. 2011 р. – Кривий Ріг : Криворізький держ. пед. ун-т, 2011. – С. 351–354.

190. Мінтій І. С. Спрощення логічних виразів за допомогою карт Карно в шкільному курсі інформатики / Л. О. Лісіна, І. С. Мінтій // Теорія та методика навчання математики, фізики, інформатики : збірник наукових праць. – Випуск VI : в 3-х томах. – Кривий Ріг : Видавничий відділ НМетАУ, 2006. – Т. 3 : Теорія та методика навчання інформатики. – С. 327–331.
191. Мінтій І. С. Схематичне програмування (початки програмування: функціональний підхід) / І. С. Мінтій ; за ред. академіка НАПН України М. І. Жалдака. – К. : НПУ ім. М. П. Драгоманова, 2010. – 152 с.
192. Мінтій І. С. Формування компетентності у програмуванні під час вивчення теми «Умовні вирази» / І. С. Мінтій // Теорія та методика навчання математики, фізики, інформатики : збірник наукових праць. Випуск IX. – Кривий Ріг : Видавничий відділ НМетАУ, 2011. – С. 496–501.
193. Мінтій І. С. Функціональне програмування в фундаментальній підготовці майбутнього вчителя / С. О. Семеріков, І. О. Теплицький, І. С. Мінтій // Матеріали VI Міжнародної науково-технічної конференції «КОМТЕХБУД 2008» : Київ-Севастополь, 9–12 вересня 2008 р. – К. : Міністерство регіонального розвитку та будівництва України, 2008 р. –С. 54–55.
194. Мінтій І. С. Функціональний підхід у формуванні мислительних операцій / І. С. Мінтій // Матеріали X Міжнародної науково-технічної конференції «Новітні комп’ютерні технології» : Севастополь, 11–14 вересня 2012 р. – К. : Мінрегіон України, 2012. – С. 160–162.
195. Мінтій І. С. Функціональний підхід як основа фундаментальності знань з програмування / І. С. Мінтій // Матеріали VII Міжнародної науково-технічної конференції «Новітні комп’ютерні технології» : Київ–Севастополь, 15–18 вересня 2009 р. – К. : Міністерство регіонального розвитку та будівницт-

ва України, 2009. – С. 42–43.

196. Морзе Н. В. Компетентнісні завдання як засіб формування інформатичної компетентності в умовах неперервної освіти / Морзе Н. В., Кузьмінська О. Г., Вембер В. П., Барна О. В. // Інформаційні технології в освіті : збірник наукових праць. – Херсон : ХДУ. – 2010. – Вип. 6. – С. 23–31.
197. Морзе Н. В. Методика навчання інформатики. Ч. 1. Загальна методика навчання інформатики / Н. В. Морзе. – К. : Навчальна книга, 2003. – 254 с.
198. Морзе Н. В. Методика навчання інформатики. Ч. 2. Методика навчання інформаційних технологій / Н. В. Морзе. – К. : Навчальна книга, 2003. – 288 с.
199. Морзе Н. В. Методика навчання інформатики. Ч. 3. Методика навчання основним послугам глобальної мережі Інтернет / Н. В. Морзе. – К. : Навчальна книга, 2003. – 196 с.
200. Морзе Н. В. Методика навчання інформатики. Ч. 4. Методика навчання основам алгоритмізації і програмування / Н. В. Морзе. – К. : Навчальна книга, 2003. – 250 с.
201. Морзе Н. В. Основи методичної підготовки вчителя інформатики : [монографія] / Морзе Н. В. – К. : Курс, 2003. – 372 с.
202. Морзе Н. В. Система методичної підготовки майбутніх вчителів інформатики в педагогічних університетах : дис. ... доктора пед. наук : 13.00.02 – теорія та методика навчання інформатики / Морзе Наталія Вікторівна ; Національний педагогічний ун-т ім. М. П. Драгоманова. – К., 2003. – 605 с.
203. Морзе Н. В. Формування інформаційної компетентності вчителя сучасної школи [Електронний ресурс]. – 2007. – Режим доступу : http://www.ua.teach-it.net/media/files/iinformacijna_kompetentnist_ suchasnogo_vchitelya_prezentaciya
204. Нигиян С. А. Функциональные и логические языки программирования : Формализация, анализ, интерпретация : автореферат дис. ... доктора физико-математических наук : 05.13.11 – математическое обеспечение вычислительных машин, комплексов и компьютерных сетей / Нигиян Семен Александрович ; МГУ им. М. В. Ломоносова. – М., 1997. – 34 с.

205. Никифорова Е. И. Формирование технологической компетентности учителя в системе повышения квалификации : автореф. дисс. на соискание ученой степени канд. пед. наук : 13.00.08 – теория и методика профессионального образования / Никифорова Елена Ивановна ; Забайкальский государственный гуманитарно-педагогический ун-т им. Н. Г. Чернышевского. – Чита, 2007. – 23 с.

206. Ничкало Н. Г. Педагогіка вищої школи: крок у майбутнє : [монографія] / Н. Г. Ничкало. – К., 2001. – 456 с.

207. Новые педагогические и информационные технологии в системе образования / Е. С. Полат, М. Ю. Бухаркина, М. В. Моисеева, А. Е. Петров. – М. : Академия, 2005. – 272 с.

208. Одинцов И. О. Профессиональное программирование. Системный подход. – 2-е изд. перераб. и доп. / Игорь Одинцов. – СПб. : БХВ-Петербург, 2004. – 624 с.

209. Ожегов С. И. Словарь русского языка / Ожегов С. И. – М. : Советская энциклопедия, 1973. – 846 с.

210. Онопрієнко О. В. Концептуальні засади компетентнісного підходу в сучасній освіті / Олександр Онопрієнко // Шлях освіти. – 2007. – № 4. – С. 32–37.

211. Основи нових інформаційних технологій навчання : [посібник для вчителів] / Авт. кол. ; за ред. Ю. М. Машбиця / Ін-т психології ім. Г. С. Костюка АПН України. – К. : ІЗМН, 1997. – 264 с.

212. Основні засади розвитку вищої освіти України в контексті Болонського процесу (документи і матеріали 2003-2004рр.) / За редакцією В. Г. Кременя. Авторський коллектив : М. Ф. Степко, Я. Я. Болюбаш, В. Д. Шинкарук та ін. – Тернопіль : ТДПУ ім. В. Гнатюка, 2004. – 147 с.

213. Основы информатики и вычислительной техники : проб. учеб. пособие для сред. учеб. заведений / [А. П. Ершов, В. М. Монахов, С. А. Бешенков и др.] ; под ред. А. П. Ершова, В. М. Монахова. – М. : Просвещение, 1988. – Ч. 1. – 94 с. – Ч. 2. – 140 с.

214. Основы педагогики и психологии высшей школы / Под ред. Петровско-

го А. В. – М. : Изд-во МГУ, 1986. – 303 с.

215. Палагина Н. Н. Психология развития и возрастная психология : учебное пособие для вузов / Н. Н. Палагина. – М. : Московский психолого-социальный институт, 2005. – 288 с.
216. Педагогика : учеб. пособие для студентов пед. учеб. заведений / В. А. Сластенин, И. Ф. Исаев, А. И. Мищенко и др. – 4 изд. – М. : Шк. пресса, 2002. – 512 с.
217. Педагогічна психологія : [навч. посібник] / Л. М. Проколієнко та ін. ; за ред. Л. М. Проколієнко, Д. Ф. Ніколенка. – К. : Вища школа, 1991. – 183 с.
218. Пейперт С. Переворот в сознании: Дети, компьютеры, плодотворные идеи : пер. с англ. / Пейперт С. – М. : Педагогика, 1989. – 234 с.
219. Перегинец Н. В. Развитие алгоритмического стиля мышления младших школьников средствами языка Лого / Н. В. Перегинец, И. А. Теплицкий // Теорія та методика навчання математики, фізики, інформатики : збірник наукових праць. Випуск 3 : В 3-х томах. – Кривий Ріг : Видавничий відділ НМетАУ, 2003. – Т. 3 : Теорія та методика навчання інформатики. – С. 264–270.
220. Петровская Л. А. Компетентность в общении. Социально-психологический тренинг / Л. А. Петровская. – М. : Изд-во МТУ, 1989. – 216 с.
221. Пєтухова Л. Є. Дидактико-процесуальне забезпечення формування інформатичних компетентностей майбутніх учителів [Електронний ресурс] / Пєтухова Любов Євгенівна // Інформаційні технології і засоби навчання. – 2008. – №3(7). – Режим доступу : http://www.nbuv.gov.ua/e-journals/ITZN/em7/content/08plytfo.htm
222. Пєтухова Л. Є. Інформатична компетентність майбутнього фахівця як педагогічна проблема / Пєтухова Л. Є. // Комп'ютер у школі та сім'ї. – 2008. – №1. – С. 3-5.
223. Пиаже Ж. Избранные психологические труды / Пиаже Ж. – М. : Международная педаг. академия, 1994. – 680 с.
224. Пидкасистый П. И. Самостоятельная познавательная деятельность школь-

ников в обучении. Теоретико-экспериментальное исследование / Пидкасистый П. И. – М. : Педагогика, 1980. – 240 с.

225. Підгорна Т. В. Вивчення технологій навчання у співробітництві в курсі методики навчання інформатики // Науковий часопис НПУ ім. М. П. Драгоманова. Серія №2. Комп'ютерно-орієнтовані системи навчання : зб. наукових праць / Редрада. – К. : НПУ ім. М. П. Драгоманова, 2009. – №7 (14). – С. 107–110.
226. Полищук А. П. Методы вычислений в классах языка С++ : учебное пособие / Полищук А. П., Семериков С. А. – Кривой Рог : Издательский отдел КГПИ, 1999. – 350 с.
227. Поліщук О. П. Методичні та організаційні проблеми навчання комп'ютерного програмування у вищих навчальних закладах / О. П. Поліщук, С. О. Семеріков // Теорія та методика навчання математики, фізики, інформатики : збірник наукових праць. Випуск VI : В 3-х томах. – Кривий Ріг : Видавничий відділ НМетАУ, 2006. – Т. 3 : Теорія та методика навчання інформатики. – С. 8–11.
228. Пометун О. І. Компетентнісний підхід – найважливіший орієнтир розвитку сучасної освіти / Олена Пометун // Рідна школа. – 2005. – № 1. – С. 65–69.
229. Прата С. Язык программирования С. Лекции и упражнения : пер. с англ. / Стивен Прата. – К. : Диасофт, 2000. – 432 с.
230. Про невідкладні заходи щодо забезпечення функціонування та розвитку освіти в Україні / Указ Президента України від 4 липня 2005 року № 1013/2005 // Офіційний вісник України від 22.07.2005. – 2005. – №27. – С. 30, стаття 1542, код акту 32924/2005.
231. Программа курса «Основы информатики и вычислительной техники» (X–XI классы) // Математика в школе. – 1986. – №3. – С. 49–53.
232. Проектування гіпертекстових навчальних систем : [посібник для вчителів] / М. І. Жалдак, Ю. І. Машбиць, О. О. Гокунь та ін. – К. : НДІ психології АПН України, 200. – 100 с.
233. Пронина С. Е. Лого в школьном курсе информатики / Пронина С. Е. // Ин-

форматика и образование. – 1995. – № 5. – С. 49-51.

234. Професійна педагогічна освіта: компетентнісний підхід : монографія / за ред. О. А. Дубасенюк. – Житомир : Вид-во ЖДУ ім. Івана Франка, 2011. – 412 с.

235. Професійні компетенції та компетентності вчителя : матеріали регіонального науково-практичного семінару. – Тернопіль : Вид-во ТНПУ ім. В. Гнатюка, 2006. – 188 с.

236. Пышкало А. М. Методическая система обучения геометрии в начальной школе : авторский доклад по монографии «Методика обучения геометрии в начальных классах», предст. на соиск. уч. степ. докт. пед. наук / Пышкало Анатолий Михайлович ; Акад. пед. наук СССР. – М., 1975. – 60 с.

237. Равен Дж. Педагогическое тестирование: Проблемы, заблуждения, перспективы : пер. с англ., изд. 2-е, испр. / Дж. Равен. – М. : Когито-Центр, 2001. – 142 с.

238. Равен Дж. Компетентность в современном обществе: выявление, развитие и реализация / Дж. Равен. – М. : Когито-Центр, 2002. – 396 с.

239. Радионова Н. Ф. Компетентностный подход в педагогическом образовании [Электронный ресурс] / Н. Ф. Радионова, А. П. Тряпицына // Электронный научный журнал «Вестник Омского государственного педагогического университета». – Выпуск 2006. – 6 с. – Режим доступа : http://www.omsk.edu/article/vestnik-omgpu-75.pdf.

240. Раков С. А. Математична освіта : компетентнісний підхід з використанням ІКТ : [монографія] / Раков С. А. – Х. : Факт, 2005. – 360 с.

241. Раков С. А. Формування математичних компетентностей учителя математики на основі дослідницького підходу у навчанні з використанням інформаційних технологій : дис. ... доктора пед. наук : 13.00.02 – теорія та методика навчання інформатики / Раков Сергій Анатолійович ; Харківський національний педагогічний ун-т ім. Г. С. Сковороди. – Харків, 2005. – 516 с.

242. Рамський Ю. С. Вивчення інформаційно-пошукових систем мережі Інтернет : [навч.-метод. посіб.] / Рамський Ю. С., Рєзіна О. В. – К. : НПУ

ім. М. П. Драгоманова, 2004. – 60 с.

243. Рамський Ю. С. Логічні основи інформатики : [навч. посіб. для студ. фіз.-мат. спец. вищих пед. навч. закл.] / Рамський Ю. С. – К. : НПУ ім. М. П. Драгоманова, 2003. – 286 с.
244. Рамський Ю. С. Методична підготовка вчителя інформатики та розвиток його фахових компетентностей / Рамський Ю. С., Балик Н. Р. // Науковий часопис НПУ ім. М. П. Драгоманова. Серія №2. Комп'ютерно-орієнтовані системи навчання : зб. наукових праць / Редрада. – К. : НПУ ім. М. П. Драгоманова, 2009. – №7 (14). – С. 32–35.
245. Рамський Ю. С. Основи програмування (мовою Паскаль) : Короткий курс лекцій. Лаборатор. практикум : [навч. посіб. для студ.] / Рамський Ю. С., Цибко Г. Ю. – К. : НПУ ім. М. П. Драгоманова, 2004. – 141 с.
246. Рафальська М. В. Формування інформатичних компетентностей майбутніх вчителів інформатики у процесі навчання методів обчислень : дис. … канд. пед. наук : 13.00.02 – теорія і методика навчання (інформатика) / Рафальська Марина Володимирівна ; Національний педагогічний ун-т ім. М. П. Драгоманова. – К., 2010. – 225 с.
247. Рубинштейн С. Л. О мышлении и путях его исследования / Рубинштейн С. Л. – М. : Изд-во АН СССР, 1958. – 147 с.
248. Рубинштейн С. Л. Основы общей психологии / Рубинштейн С. Л. – СПб. : Питер, 2002. – 720 с.
249. Себеста Р. У. Основные концепции языков программирования. – 5-е издание : пер. с англ. / Роберт У. Себеста – М. : Вильямс, 2001. – 672 с.
250. Сейдаметова З. С. Мови програмування в навчанні майбутніх програмістів / Сейдаметова З. С., Манжос Л. О. // Науковий часопис НПУ ім. М. П. Драгоманова. Серія № 2. Комп'ютерно-орієнтовані системи навчання : зб. наукових праць / Редрада. – К. : НПУ ім. М. П. Драгоманова, 2010. – № 8 (15). – С. 35–41.
251. Семеріков С. О. Активізація пізнавальної діяльності студентів при вивченні чисельних методів у об'єктно-орієнтованій технології програмування : дис.

... канд. пед. наук : 13.00.02 – теорія та методика навчання інформатики / Семеріков Сергій Олексійович ; Криворізький державний педагогічний ун-т. – Кривий Ріг, 2000. – 256 с.

252. Семеріков С. О. Теоретико-методичні основи фундаменталізації навчання інформатичних дисциплін у вищих навчальних закладах : дис. ... доктора пед. наук : 13.00.02 – теорія та методика навчання (інформатика) / Семеріков Сергій Олексійович ; Національний педагогічний ун-т ім. М. П. Драгоманова. – К., 2009. – 369 с.
253. Сидоренко Е. В. Методы математической обработки в психологии / Е. В. Сидоренко. – СПб. : Речь, 2003. – 350 с.
254. Сікора Я. Б. Критерії та рівні сформованості професійної компетентності майбутнього вчителя інформатики / Я. Б. Сікора // Вісник Житомирського державного університету ім. Івана Франка. Педагогічні науки. – Випуск 42. – Житомир : Вид-во ЖДУ ім. Івана Франка, 2008. – С. 154–159.
255. Сікора Я. Б. Формування професійної компетентності майбутнього вчителя інформатики засобами моделювання : автореф. дис. на здобуття наук. ступеня канд. пед. наук : 13.00.04 – теорія і методика професійної освіти / Сікора Ярослава Богданівна ; Житомирський державний ун-т ім. Івана Франка. – Житомир, 2010. – 21 с.
256. Скаткин М. Н. Активизация познавательной деятельности учащихся в обучении / Скаткин М. Н. – М. : АПН РСФСР, 1965. – 48 с.
257. Скаткин М. Н. О методах обучения / Скаткин М. Н., Лернер И. Я. // Советская педагогика. – №3. – 1965. – С. 3 – 10.
258. Сластенин В. А. Педагогика : учеб. пособие / В. А. Сластенин. – М. : Школа-Пресс, 1998. – 512 с.
259. Слинкин Д. А. Использование метода проектов при обучении программированию в курсе информатики : дисс. … канд. пед. наук : 13.00.02 – теория и методика обучения и воспитания (информатика, уровень общего образования) / Слинкин Дмитрий Анатольевич ; Шадринский государственный педагогический институт. – Екатеринбург, 2001. – 157 с.

260. Словник іншомовних слів / [За ред. О. С. Мельничука]. – К. : Гол. ред. укр. рад. енциклопедії, 1985. – 966 с.

261. Словник української мови. В 11 т. Т. 4. / [Голова ред. кол. І. К. Білодід] – К. : Наукова думка, 1973. – 840 с.

262. Смирнова-Трибульска Є. М. Теоретико-методичні основи формування інформатичних компетентностей вчителів природничих дисциплін у галузі дистанційного навчання : автореф. дис. на здобуття наук. ступеня доктора пед. наук : 13.00.02 – теорія та методика навчання (інформатика) / Смирнова-Трибульска Євгенія Миколаївна ; Національний педагогічний ун-т ім. М. П. Драгоманова. – К., 2008. – 44 с.

263. Смирнова-Трибульска Є. М. Теоретичні і практичні аспекти формування у майбутніх вчителів компетентностей в галузі дистанційного навчання / Смирнова-Трибульска Є. М. // Збірник наукових статей «Проблеми сучасної педагогічної освіти». Серія : «Педагогіка і психологія» (Кримський державний гуманітарний інститут). – Вип. №15. – Ялта : РВВ КГУ, 2007. – С. 3–16.

264. Смирнова-Трибульская Е. Н. Основы формирования информатических компетентностей учителей в области дистанционного обучения [монография] / Смирнова-Трибульская Е. Н. ; научный редактор : академик АПН Украины, д.пед.наук, проф. М. И. Жалдак. – Херсон : Айлант, 2007. – 704 с.

265. Смолянинова О. Г. Развитие методической системы формирования информационной и коммуникативной компетентности будущего учителя на основе мультимедиа-технологий : автореф. дисс. на соискание ученой степени доктора пед. наук : 13.00.02 – теория и методика обучения и воспитания (информатика, уровень профессионального образования) / Смолянинова Ольга Георгиевна ; Российский государственный педагогический ун-т им. А. И. Герцена. – СПб., 2002. – 44 с.

266. Современный словарь по педагогике / Сост. Е. С. Рапацевич. – Минск : Современное слово, 2001. – 928 с.

267. Соловова Е. Н. Компоненты профессиональной компетенции учителя [Электронный ресурс] / Соловова Елена Николаевна. – [2008]. – Режим до-

ступа : http://distant.ioso.ru/seminary/13-03-08/compro.ppt

268. Соціологія : словник термінів і понять / [за заг. ред. Біленького Є. А., Козловця М. А.]. – К. : Кондор, 2006. – 372 с.
269. Співаковський О. В. Теорія і практика використання інформаційних технологій у процесі підготовки студентів математичних спеціальностей : моногр. / Співаковський О. В. – Херсон : Айлант, 2003. – 228 с.
270. Співаковський О. В. Алгоритмізація та програмування. Енциклопедичне видання : [навч.-метод. посіб.] / Співаковський О. В. – К. : Комп'ютер, 2007. – 128 с.
271. Спірін О. М. Аналіз стану підготовки вчителя інформатики в умовах упровадження кредитно-модульної системи навчання [Електронний ресурс] / Спірін Олег Михайлович // Інформаційні технології і засоби навчання. – 2008. – №2(6). – Режим доступу : http://www.nbuv.gov.ua/e-journals/ITZN/em6/content/08somsmc.htm
272. Спірін О. М. Інформаційно-комунікаційні та інформатичні компетентності як компоненти системи професійно-спеціалізованих компетентностей вчителя інформатики [Електронний ресурс] / Спірін Олег Михайлович // Інформаційні технології і засоби навчання. – 2009. – №5(13). – Режим доступу : http://www.ime.edu-ua.net/em13/emg.html
273. Спірін О. М. Теоретичні та методичні засади професійної підготовки майбутніх учителів інформатики за кредитно-модульною системою : [монографія] / О. М. Спірін ; за наук. ред. акад. М. І. Жалдака. – Житомир : Вид-во ЖДУ ім. Івана Франка, 2007. – 300 с.
274. Старша школа зарубіжжя: організація та зміст освіти / За ред. О. І. Локшиної. – К. : СПД Богданова А. М., 2006. – 232 с.
275. Стратегия модернизации содержания общего образования: материалы для разработки документов по обновлению общего образования / Под ред. А. А. Пинского. – М. : Мир книги, 2001. – 102 с.
276. Стрельніков В. Ю. Професійна компетентність вчителя / В. Ю. Стрельніков // Актуальні проблеми безперервного підвищення кваліфікації педагогічних

кадрів України в умовах становлення національної школи : [збірник статей / за ред. Крисюка С. В.]. – К., 1992. – С. 44–45.

277. Струтинська О. В. Методика навчання інформаційних систем і технологій майбутніх учителів економіки : дис. ... канд. пед. наук : 13.00.02 – теорія та методика навчання (інформатика) / Струтинська Оксана Віталіївна ; Національний педагогічний ун-т ім. М. П. Драгоманова. – К., 2010. – 225 с.
278. Сухомлин В. А. Разработка системы компетенций для образовательного стандарта нового поколения по направлению «Информационные технологии» [Электронный ресурс] / В. А. Сухомлин. – [2006]. – 17 с. – Режим доступа : http://inno.cs.msu.su/implementation/it-university/01/document.doc
279. Талызина Н. Ф. Педагогическая психология : учеб. пособие для студ. сред. пед. учеб. заведений / Н. Ф. Талызина. – М. : Академия, 1998. – 288 с.
280. Теория и практика педагогического эксперимента / [Под ред. А. И. Пискунова]. – М. : Педагогика, 1979. – 208 с.
281. Теплицкий И. А. Введение в программирование систем искусственного интеллекта на языке Лисп : лабораторный практикум / Теплицкий И. А., Семериков С. А. – Кривой Рог : КГПУ, 2004. – 88 с.
282. Теплицький І. О. Дослідження дидактичних можливостей мови Лісп як засобу побудови інтелектуальних систем у шкільному курсі інформатики / Теплицький І. О., Семеріков С. О. // Проблеми сучасного підручника : зб. наук. праць / Ред. кол. – К. : Педагогічна думка, 2004. – Вип. 5., Ч. ІІ. – С. 183–191.
283. Теплицький І. О. Елементи комп’ютерного моделювання : навчальний посібник / І. О. Теплицький. – Кривий Ріг : КДПУ, 2005. – 208 с.
284. 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 / Триус Юрій Васильович ; Черкаський нац. ун-т ім. Б. Хмельницького. – Черкаси, 2005. – 649 с.
285. Триус Ю. В. Особливості створення методичної системи навчання основам програмування для підготовки майбутніх інженерів-програмістів / Три-

ус Ю. В., Богатирьов О. О., Гришко Л. В. // Вісник Черкаського університету. Серія педагогічні науки. – Випуск 35. – Черкаси, 2002. – С. 133–141.

286. Умрик М. А. Організація самостійної роботи майбутніх учителів інформатики в умовах дистанційного навчання інформатичних дисциплін : дис. ... канд. пед. наук : 13.00.02 – теорія та методика навчання (інформатика) / Умрик Марія Анатоліївна ; Національний педагогічний ун-т ім. М. П. Драгоманова. – К., 2009. – 210 с.
287. Унт И. Э. Индивидуализация и дифференциация обучения / И. Э. Унт. – М. : Педагогика, 1990. – 192 с.
288. Уоррен Г. Алгоритмические трюки для программистов : пер. с англ. / Генри Уоррен. – М. : Вильямс, 2003. – 228 с.
289. Фалина И. Н. Компетентностный подход в обучении и стандарт образования по информатике [Электронный ресурс] / И. Н. Фалина. – Режим доступа : http://inf.1september.ru/articlef.php?ID=200600703
290. Федорчук О. С. Формування аксіологічного компоненту інформатичної компетентності при підготовці майбутніх правознавців [Електронний ресурс] / О. С. Федорчук // Педагогічний дискурс. – 2008. – Випуск 4. – Режим доступу : http://www.nbuv.gov.ua/portal/soc_gum/peddysk/2008_04/fedorchuk.pdf
291. Филд А. Функциональное программированию : пер. с англ. / А. Филд, П. Харрисон. – М. : Мир, 1993. – 640 с.
292. Хараджян Н. А. Педагогічні умови підготовки фахівців з економічної кібернетики засобами комп'ютерного моделювання : дис. ... канд. пед. наук : 13.00.04 – теорія і методика професійної освіти / Хараджян Наталя Анатоліївна ; Черкаський національний ун-т ім. Б. Хмельницького. – Черкаси, 2011. – 286 с.
293. Харківська А. А. Формування інформатичної компетентності майбутнього вчителя інформатики в педагогічному ВНЗ / Харківська А. А. // Проблеми інженерно-педагогічної освіти. – 2009. – № 24–25. – С. 411–419
294. Хендерсон П. Функциональное программирование. Применение и реализа-

ция : пер. с англ. / П. Хендерсон. – М. : Мир, 1983. – 349 с.

295. Хомылева А. Н. Модульно-компетентностный подход в обучении преподавателей вуза / А. Хомылева // Высшее образование в России. – 2007. – № 9. – С. 127–131.

296. Хуторской А. В. Ключевые компетенции. Технология конструирования / Хуторской А. В. // Народное образование. – 2003. – № 5. – С. 55–61.

297. Хуторской А. В. Современная дидактика / Хуторской А. В. – СПб. : Питер, 2001. – 544 с.

298. Хювёнен Э. Мир Лиспа. В 2-х т. Т. 1. : Введение в язык Лисп и функциональное программирование : пер. с финск. / Э. Хювёнен, И. Сеппянен. – М. : Мир, 1990. – 447 с.

299. Хювёнен Э. Мир Лиспа. В 2-х т. Т. 2. : Методы и системы программирования : пер. с финск. / Э. Хювёнен, И. Сеппянен. – М. : Мир, 1990. – 319 с.

300. Черных Л. А. Теоретические основы разработки методической системы обучения / Л. А. Черных // Евристика та дидактика точних наук : збірник наукових робіт. – Вип. 3. – Донецьк : Донецька школа евристики та точних наук, 1995. – С. 15–19.

301. Чёрч А. Введение в математическую логику. Т. 1. / Чёрч А. – М. : Изд-во иностранной литературы, 1960. – 486 с.

302. Чорна Ж. А. Формування технологічної компетентності майбутнього вчителя інформатики у педагогічному університеті [Електронний ресурс] / Чорна Ж. А. // Теорія і методика навчання інформатики та математики : зб. наук. праць. – Випуск 3. – Мелітополь : МДПУ, 2004. – Режим доступу : http://www.conference.mdpu.org.ua/conf_all/confer/2004/inf_mat/articles/chernaya.pdf

303. Шадриков В. Д. Требования к программам профессионального педагогического образования, соответствующим государственным стандартам общего образования второго поколения, и результатам их освоения [Электронный ресурс]. – [2008]. – 312 с. – Режим доступа : http://standart.edu.ru/attachment.aspx?id=154

304. Шапар В. Б. Сучасний тлумачний психологічний словник / Шапар В. Б. – Харків : Прапор, 2007. – 640 с.

305. Шаповаленко И. В. Возрастная психология (Психология развития и возрастная психология) / И. В. Шаповаленко. – М. : Гардарики, 2005. – 349 с.

306. Швецкий М. В. Методическая система фундаментальной подготовки будущих учителей информатики в педагогическом вузе в условиях двухступенчатого образования : автореф. дисс. на соискание ученой степени доктора пед. наук : 13.00.02 – теория и методика обучения и воспитания (информатика) / Швецкий Михаил Владимирович. – СПб., 1994. – 36 с.

307. Шелудько В. С. Структура професійних компетентностей учителя інформатики [Електронний ресурс] / В. С. Шелудько // Збірник наукових праць Харківського національного педагогічного університету ім. Г. С. Сковороди «Педагогіка та психологія». – 2009. – № 35. – Режим доступу : http://www.nbuv.gov.ua/portal/soc_gum/znpkhnpu_ped/2009_35/index.html

308. Шилдт Г. Самоучитель С++, 3-е изд : пер. с англ. / Герберт Шилдт. – СПб. : BHV, 1998. – 668 с.

309. Шишов С. Е. Компетентностный подход к образованию как необходимость / Шишов С. Е., Агапов И. Г. // Мир образования – образование в мире. – 2001. – № 4.– С. 8–19.

310. Шокалюк С. В. Методичні засади комп’ютеризації самостійної роботи старшокласників у процесі вивчення програмного забезпечення математичного призначення : дис. ... канд. пед. наук : 13.00.02 – теорія та методика навчання (інформатика) / Шокалюк Світлана Вікторівна ; Національний педагогічний ун-т ім. М. П. Драгоманова. – К., 2009. – 261 с.

ДОДАТКИ

Додаток А

Навчальні заклади, в яких використовується мова програмування Scheme

№	Адреса	Короткий опис
1.	http://schemers.com/	список навчальних закладів, в яких використовується мова програмування Scheme
2.	http://www.cs.brandeis.edu/~cs21b/	Курс «Structure and Interpretation of Computer Programs», Університет Брандейса, м. Велтам, штат Массачусетс, США
3.	http://www.math.grin.edu/~stone/courses/scheme/	Курс «Fundamentals of computer science I», Гріннелл-коледж, м. Гріннелл, штат Айова, США
4.	http://cs.bilgi.edu.tr/pages/courses/year_1/comp_149/	Курс «The Fundamentals of Computer Science I», Стамбульський університет, м. Стамбул, Туреччина
5.	http://sicp.csail.mit.edu/Fall-2005/	Курс «Structure and Interpretation of Computer Programs», Массачусетський технологічний інститут, м. Кембридж, штат Массачусетс, США
6.	http://www.cs.northwestern.edu/academics/courses/111/	Курс «Fundamentals of Computer Programming», Північно-Західний університет, м. Еванстон, штат Іллінойс, США
7.	http://www.cs.trinity.edu/About/The_Courses/cs301/	Курс «Great Ideas in Computer Science», Трініті університет, м. Сан-Антоніо, штат Техас, США
8.	http://www.cs.trinity.edu/~meggen/Classes/2322/	Курс «Principles of Functional Languages», Трініті університет, м. Сан-Антоніо, штат Техас, США
9.	http://www.dccia.ua.es/dccia/inf/asignaturas/LPP/	Курс «Programming Languages and Paradigms», Університет Аліканте, м. Аліканте, Іспанія
10.	http://www.classes.cs.uchicago.	Курс «Introduction to Computer Science I», Чика-

№	Адреса	Короткий опис
	edu/archive/2009/fall/15100-1/	зький університет, м. Чикаго, штат Іллінойс, США
11.	http://zoo.cs.yale.edu/classes/cs201/	Курс «Introduction to Computer Science», Єльський університет, м. Нью-Хейвен, штат Коннектикут, США
12.	http://www.cs.caltech.edu/courses/cs1/	Курс «Introduction to Computation», Каліфорнійський університет, м. Берклі, Каліфорнія, США

Додаток Б

Робота з web-сервісом Dropbox

Папка Dropbox

Після встановлення Dropbox на комп'ютері буде створено папку Dropbox, що має певну особливість: будь-який файл, збережений в папці Dropbox також зберігається на всі інші комп'ютери і мобільні пристрої користувача та на веб-сайті Dropbox. На папці Dropbox є значок, що надає можливість дізнатись стан Dropbox: зелене коло з галочкою – всі файли завантажено, синє коло зі стрілками – файли в Dropbox в даний час оновлюються.

Додавання файлів у Dropbox

Крок 1. Перетягніть файли в папку Dropbox.

Крок 2. Синій значок означає, що відбувається синхронізація файлів з Dropbox.

Крок 3. От і все! Зелений значок означає, що файли синхронізовано. Тепер, коли файли завантажено в Dropbox, при внесенні в них будь-яких змін їх буде відображено на всіх комп'ютерах користувача.

Контекстне меню Dropbox

При клацанні правою кнопкою миші на папці Dropbox (або файлі, що знаходиться в ній) можливо:

– надати спільний доступ до папки;

– переглянути попередні версії та повернутися назад до попередньої версії файлу;

– переглянути файли на веб-сайті Dropbox.

– створити посилання на будь-який файл чи папку для надання до них доступу іншим користувачам.

Значок Dropbox на панелі задач

Значок Dropbox на панелі задач надає можливість перевірити налаштування та стан вашого Dropbox.

Відкривши контекстне меню для цього значка можна:

– відкрити папку Dropbox або зайти на сайт Dropbox;

– дізнатись, які файли нещодавно було змінено;

– переглянути, скільки часу займе оновлення файлів (якщо файл нещодавно було змінено, це буде завантаження лише його змін, а не всього файлу) та ін.

Розмір папки Dropbox

Безкоштовні облікові записи мають 2 Гб простору. Існує багато способів безкоштовного збільшення простору, один із найпростіших – запрошення друзів у Dropbox (500 Мб за друга, але не більше 16 Гб).

Dropbox для мобільних пристроїв

Додатки Dropbox доступні для iPhone, iPad, Android та Blackberry. Для встановлення Dropbox на мобільному пристрої необхідно відвідати http://www.dropbox.com/anywhere.

Додаток В

Проект 4. Експертна система

1. Структура ЕС

ЕС (система обробки знань) – це програмна система, базу знань та базу даних якої можна порівняти з уміннями фахівців у певній спеціальній галузі знань. ЕС разом з системами обробки природних мов є найбільш важливими в комерційному плані галузями використання штучного інтелекту.

У рамках досліджень штучного інтелекту створено численні ЕС для різних галузей знань, таких, наприклад, як медична діагностика та обстеження пацієнтів, генні та молекулярні дослідження, складання конфігурацій комп'ютерів, освіта, пошук несправностей у пристроях і системах та багато інших практичних застосувань. Мета цієї роботи – розробка невеликої ЕС, яку назвемо `Експерт`.

Типова ЕС складається з двох головних частин: машини виведення і бази знань. База знань містить факти і правила виведення з області застосування; спосіб вирішення проблем, що використовується в машині виведення, не пов'язаний із даними з предметної області. Крім того, в систему обов'язково входить якась мова взаємодії людини з машиною, за допомогою якого користувач-фахівець веде «діалог» з системою, а також до неї входять засоби, за допомогою яких інженер знань та експерт(и) підтримують базу знань.

2. Подання знань

Для подання знань використовують різні формалізми і мови подання даних. Найбільш часто зустрічається подання знань за допомогою продукційних правил типу `ЯКЩО-ТО`. У системі `Експерт` правила, що описують прийняття рішення, можна задавати у формі, що схожа на природну мову:

```
(ЯКЩО умова-1
І умова-2
...
І умова-і
ТО висновок-1
І висновок-2
```

```
...
I висновок-j)
```

Умови та висновки – це прості речення природної мови. Наприклад:

```
(ЯКЩО на лампочку подано напругу
I лампочка не горить
ТО лампочка, ймовірно, перегоріла)
(ЯКЩО читач перестав розуміти
I читач хоче вчитися
I читач ще може читати
ТО читачеві потрібно почати все спочатку
I читачеві слід бути уважнішим)
```

Вихідні дані і висновки, отримані в результаті прийняття рішень, надалі будемо називати відомими системі *фактами*.

Розглянемо більш детально структуру машини прийняття рішень програми Експерт.

3. Факти і правила

У системі Експерт факти подаються просто у вигляді списків символів. Наприклад, наступний список міг би описувати наші знання про яку-небудь тварину:

```
(
  (Тварина має шерсть)
  (Тварина смугаста)
  (Тварина жуйна)
)
```

База знань системи Експерт утворюється зі списків, подібних раніше описаним правилами ЯКЩО-ТО. Наприклад:

```
(ПРАВИЛО12 .
     (ЯКЩО тварина жуйна
      I тварина смугаста
      ТО тварина зебра)
)
```

Щоб правила можна було звести до фактів, їх умови і висновки необхідно представляти списками фактів. Для цього вирази, які є правилами, можна перетворити в структури, які складаються з імені правила, умов і висновків, представ-

лених у вигляді списку фактів:

```
; Перетворює правило у формі речення на структуру –
; список з 3-х компонентів
(define (аналізуй-правило правило)
  (list
     (car правило)
     (умови (cdr правило))
     (висновки (cdr правило))
   )
)
```

Якщо умови правила є елементами якого-небудь списку фактів, то застосування правила до цього списку фактів можна реалізувати шляхом додавання в список висновків з правила. Наприклад, застосувавши ПРАВИЛО12 до розглянутого вище списку даних, можна зробити висновок: тварина, про яку йде мова – зебра.

4. Правила виведення бази знань

Знання системи Експерт про тварин містяться в базі знань, яка містить 14 правил:

```
(define *база-знань* '(
  (ПРАВИЛО1 .
    (ЯКЩО тварина має шерсть
     ТО тварина ссавець)
   )
  (ПРАВИЛО2 .
    (ЯКЩО тварина годує дитинчат молоком
     ТО тварина ссавець)
   )
  (ПРАВИЛО3 .
    (ЯКЩО тварина має пір'я
     ТО тварина птах)
   )
  (ПРАВИЛО4 .
    (ЯКЩО тварина вміє літати
     І тварина несе яйця
     ТО тварина птах)
```

```
)
(ПРАВИЛО5 .
  (ЯКЩО тварина їсть м'ясо
   ТО тварина хижак)
)
(ПРАВИЛО6 .
  (ЯКЩО тварина має гострі зуби
   І тварина має пазурі
   І очі тварини посаджені прямо
   ТО тварина хижак)
)
(ПРАВИЛО7 .
  (ЯКЩО тварина ссавець
   І тварина має копита
   ТО тварина жуйна)
)
(ПРАВИЛО8 .
  (ЯКЩО тварина ссавець
   І тварина жує жуйку
   ТО тварина жуйна)
)
(ПРАВИЛО9 .
  (ЯКЩО тварина ссавець
   І тварина хижак
   І тварина жовто-коричнева
   І тварина має темні плями
   ТО тварина гепард)
)
(ПРАВИЛО10 .
  (ЯКЩО тварина ссавець
   І тварина хижак
   І тварина жовто-коричнева
   І тварина смугаста
   ТО тварина тигр)
)
(ПРАВИЛО11 .
```

```
    (ЯКЩО тварина жуйна
     І тварина довгошия
     І тварина довгонога
     І тварина має темні плями
     ТО тварина жираф)
  )
  (ПРАВИЛО12 .
     (ЯКЩО тварина жуйна
      І тварина смугаста
      ТО тварина зебра)
  )
  (ПРАВИЛО13 .
     (ЯКЩО тварина птах
      І тварина не вміє літати
      І тварина довгошия
      І тваринадовгонога
      І тварина чорно-біла
      ТО тварина страус)
  )
  (ПРАВИЛО14 .
     (ЯКЩО тварина птах
      І тварина не вміє літати
      І тварина плаває
      І тварина чорно-біла
      ТО тварина пінгвін)
  )
  )
)
```

Наступна функція `аналізуй` бере список правил (у нашому випадку – це `*база-знань*`), аналізує кожне правило і перетворює його до раніше описаної структури `(ПРАВИЛО)`, полями якої є: ім’я правила, умови і висновки:

```
;Функція для додавання елемента у кінець списка
(define (приєднай x y)
  (append x (list y))
)
```

```
;Аналізує правила та створює з них список
(define (аналізуй правила)
  (map аналізуй-правило правила)
)

;Функції доступа до елементів структури
(define (правило-ім'я правило)
   (car правило)
)

(define (правило-умови правило)
   (cadr правило)
)

(define (правило-висновки правило)
   (caddr правило)
)


;Повертає умови у вигляді списка
(define (умови речення)
  (речення-і (cdr речення) null null)
)

;Повертає висновки у вигляді списка
(define (висновки речення)
  (речення-і (cdr (member 'ТО речення)) null null)
)

;Виділяє речення, що розділяються прийменниками І, ТА, Й
(define (речення-і речення частина результат)
  (cond
    (
      (null? речення) ;; умова закінчення
      (приєднай результат частина)
```

```
    )
    (
      (member (car речення) '(ТО то)) ;; умова закінчення
      (приєднай результат частина)
    )
    (
      (member (car речення) '(І ТА Й і та й)) ;; нове речення
      (речення-і
        (cdr речення)
        null
        (приєднай результат частина)
      )
    )
    (
      else ; наступне слово
      (речення-і
        (cdr речення)
        (приєднай частина (car речення))
        результат
      )
    )
  )
)
```

Правила у вигляді записів легко обробляти, і функціям більш високого рівня нічого не треба знати про аналіз даних або деталі подання правил. Надалі це полегшує можливу модифікацію програм і їх подальший розвиток.

Проаналізовані варіанти правил збережемо у змінній `*правила*` наступною командою:

```
(define *правила* (аналізуй *база-знань*))
```

Значенням змінної `*правила*` буде список, що складається з структур.

Припустимо, що система `Експерт` зберігає відомі їй факти у списку `*факти*`, що складається з елементів даних. У цьому випадку застосовність правила можна перевірити функцією `перевірити-правило` і виведення правила можна при необхідності додати до фактів функцією `додати-висновки`:

```
;Перевіряє, чи можна застосувати правило
(define (перевірити-правило правило)
  (підмножина (правило-умови правило) *факти*)
)
;перетин множин
(define (перетин l1 l2)
  (cond
    (
      (or (not (list? l1)) (not (list? l2)))
      (error "Параметрами мають бути списки")
    )
    (
      (or (null? l1) (null? l2))
      null
    )
    (
      (not (member (car l1) l2))
      (remove-duplicates (перетин (cdr l1) l2))
    )
    (else
      (remove-duplicates (cons (car l1) (перетин (cdr l1) l2)))
    )
  )
)

;об'єднання множин
(define (об’єднання l1 l2)
  (cond
    (
      (or (not (list? l1)) (not (list? l2)))
      (error "Параметрами мають бути списки")
    )
    (
      (null? l1)
      l2
    )
```

```
    (
      (null? l2)
      l1
    )
    (
      (member (car l1) l2)
      (remove-duplicates (об'єднання (cdr l1) l2))
    )
    (else
      (remove-duplicates (cons (car l1) (об'єднання (cdr l1) l2)))
    )
  )
)

;Перевіряє, чи є множина підмножиною
(define (підмножина підмножина множина)
  (equal?
    підмножина
    (перетин підмножина множина)
  )
)

;Розширює список фактів висновками правила
(define (додати-висновки правило)
  (do ((висновки (правило-висновки правило) (cdr висновки)))
    ((null? висновки) *факти*)
    (if (member (car висновки) *факти*)
      null
      (begin
        (printf "Згідно правила ~a: " (правило-ім'я правило))
        (вивести-елементи (car висновки))
        (set! *факти* (cons (car висновки) *факти*))
      )
    )
  )
)
```

```
;Виводить елементи списка
(define (вивести-елементи список)
  (for ([елемент список])
    (printf "~a " елемент)
  )
  (newline)
  #t
)
```

5. Машина виведення

Машина виведення – це механізм (програма або апарат), який на основі фактів і правил виведення, що знаходяться у базі знань, будує нові факти, задає додаткові питання і так далі доти, доки не прийде до якого-небудь прийнятного кінцевого результату або відповіді.

У більш великих системах база знань містить окрім правил виведення знання й інших типів: факти про об’єкти проблемної області й їх властивості, дані про обставини та події, ієрархії понять, метадані і т.д.

Робота ЕС ґрунтується, в першу чергу, на великій базі знань. Робота машин виведення часто досить проста і прямолінійна.

6. Стратегія оберненого виведення

Машина виведення застосовує правила до відомих у поточний момент системі фактів для знаходження нових фактів доти, доки в результаті застосування не буде отриманий шуканий результат. Якщо у системи недостатньо відомостей для подальшого пошуку рішення, то вона запитує у користувача додаткові відомості і зберігає їх у своєму списку фактів, а потім намагається ще раз застосувати до своїх правил нові додаткові факти і т.д.

У дослідженнях зі штучного інтелекту створені різні способи знаходження рішення і виконання дій. У системі `Експерт` ми застосуємо *обернене виведення*, в якому рішення намагаються знайти, йдучи у зворотному напрямі від кінцевого результату.

7. Робота системи «Експерт»

Систему Експерт на самому верхньому рівні можна представити як розпізнавача тварин, що намагається на основі своїх правил довести деяку гіпотезу.

Подамо можливі кінцеві результати у вигляді списку *гіпотези*:

```
(define *гіпотези*
  '(
    (тварина пінгвін)
    (тварина страус)
    (тварина зебра)
    (тварина жираф)
    (тварина тигр)
    (тварина гепард)
  )
)
```

Всі ці гіпотези зустрічаються в частині виведення деяких правил. При зворотному виведенні умови цих правил можна інтерпретувати як нові гіпотези, якщо висновок є кінцевим результатом, і так далі. Так система породжує гіпотези нижчого рівня до тих пір, доки не виявить, що нових правил для породження гіпотез більше немає. Тоді система запитує умови правила безпосередньо у користувача, які він на цьому рівні вже, ймовірно, зможе поставити сам і не будучи фахівцем, потім система за допомогою нових відомостей намагається рухатися далі у своїх висновках:

```
(define *факти* null)
(define *запити* null)

(define (Експерт)  ; ЕС
  (set! *факти* null)
  (set! *запити* null)
  (display (машина-виведення *гіпотези*))
)

;Намагається перевірити яку-небудь гіпотезу
(define (машина-виведення гіпотези)
```

```
  (cond
    (
      (null? гіпотези)
      "Не можу довести жодну з відомих мені гіпотез"
    )
    (
      (доведи (car гіпотези))
      (car гіпотези)   ; результат
    )
    (
      else
      (машина-виведення (cdr гіпотези)) ; нова спроба
    )
  )
)
```

Доведення гіпотези або твердження, можна виконати за допомогою функції `доведи`:

Дано: гіпотеза, що доводиться.

Значення: значення функції – істина, якщо дану гіпотезу можна довести.

Дії:

1. Якщо гіпотеза вже є в списку фактів, значить вона доведена.

2. Якщо це не так, то зберемо всі правила, за допомогою яких можна було б довести гіпотезу, тобто ті правила, в частині виведення яких є ця гіпотеза. Якщо яке-небудь з цих правил можна прямо застосувати до списку фактів, то гіпотеза доведена.

Якщо жодне з вибраних правил не можна застосувати, то спробуємо довести застосовність будь-якого правила, довівши всі умови правила, взявши їх як нові гіпотези і рекурсивно застосувавши функцію `доведи`.

3. Якщо дії 1 і 2 не дають результату, то запитаємо у користувача, чи вірна гіпотеза (якщо тільки це ще не питалось), і якщо вона вірна, то приєднаємо твердження до списку фактів. Приєднаємо твердження до списку `*запити*` незалежно від відповіді, щоб потім не довелося повторно запитувати це ж саме.

Цю рекурсивну функцію можна безпосередньо запрограмувати. Всі три гілки описаної функції відмічені у визначенні відповідними коментарями:

```
(define (доведи гіпотеза)
    (cond
      ((member гіпотеза *факти*) #t)      ; гілка 1
      (
        (not (null? (можливі гіпотеза)))  ; гілка 2
        (if (прямо гіпотеза (можливі гіпотеза))
          #t
          (рекурсивно гіпотеза (можливі гіпотеза))
        )
      )
      (
        else                              ; гілка 3
        (гілка3 гіпотеза)
      )
    )
)

(define (гілка3 гіпотеза)
  (cond
    ((member гіпотеза *запити*) #f)
    (
      (and
        (display "Чи це правда, що: ")
        (вивести-елементи гіпотеза)
        (member (read) '(Так так))
      )
      (begin
        (set! *факти* (об'єднання (list гіпотеза) *факти*))
        #t
      )
    )
    (
      else
```

```
      (begin
        (set! *запити* (cons гіпотеза *запити*))
        #f
      )
    )
  )
)

(define (можливі гіпотеза)
  (let ((result null))
    (for ([правило *правила*])
      (when (member
               гіпотеза (правило-висновки правило) )
         (set! result (append result (list правило)))
      )
    )
    result
  )
)

;Перевіряє, чи можна довести гіпотезу безпосередньо
;за допомогою якого-небудь правила
(define (прямо гіпотеза можливі)
  (cond
    ((null? можливі) #f)
    ((null? *факти*) #f)
    (
      (перевірити-правило (car можливі))
      (додати-висновки (car можливі))
    )
    (
      else
      (прямо гіпотеза (cdr можливі))
    )
  )
)
```

```
;Рекурсивно перевіряє гіпотезу
(define (рекурсивно гіпотеза можливі)
  (cond
    ((null? можливі) #f)
    (
      (перевірити-непрямо (правило-умови (car можливі)))
      (додати-висновки (car можливі))
    )
    (
      else
      (рекурсивно гіпотеза (cdr можливі))
    )
  )
)

;Рекурсивно перевіряє всі умови
(define (перевірити-непрямо умови)
  (for/and ([умова умови])
    (доведи умова)
  )
)
```

Керуючись своїми правилами, програма може задавати користувачеві лише розумні з точки зору гіпотези питання. Проте першу гіпотезу доводиться вибирати навмання.

8. Приклади запитів

Тепер трохи «поговоримо» з програмою `Експерт`. У першому прикладі користувач бачить перед собою пінгвіна (рис. В.1), у другому – зебру (рис. В.2), але тварин він не знає. За допомогою програми він може визначити види цих тварин, даючи системі прості відповіді на її питання.

```
Мова: racket; memory limit: 128 MB.
> (Експерт)
Чи це правда, що: тварина має пір 'я
Так
Згідно правила ПРАВИЛО3: тварина птах
Чи це правда, що: тварина не вміє літати
так
Чи це правда, що: тварина плаває
так
Чи це правда, що: тварина чорно-біла
так
Згідно правила ПРАВИЛО14: тварина пінгвін
(тварина пінгвін)
```

Рис. В.1. Розпізнавання пінгвіну

```
Мова: racket; memory limit: 128 MB.
> (Експерт)
Чи це правда, що: тварина має пір 'я
Ні
Чи це правда, що: тварина вміє літати
ні
Чи це правда, що: тварина має шерсть
Так
Згідно правила ПРАВИЛО1: тварина ссавець
Чи це правда, що: тварина має копита
так
Згідно правила ПРАВИЛО7: тварина жуйна
Чи це правда, що: тварина смугаста
так
Згідно правила ПРАВИЛО12: тварина зебра
(тварина зебра)
>
```

Рис. В.2. Розпізнавання зебри

Додаток Г

Типи вікон:

контейнери – вікна, що можуть містити інші вікна:

– `frame%` – рамка – вікно верхнього рівня класу Вікон, які користувач може переміщувати та змінювати розмір;

– `dialog%` – діалогове вікно верхнього рівня класу Вікон, коли відображається діалогове вікно, всі інші вікна є недієздатними, доки це вікно не буде закрите;

– `panel%` – панель – підконтейнер всередині контейнерів. Існують такі підкласи класу panel%: vertical-panel% та horizontal-panel%;

підконтейнери – вікна, що мають міститись в інших вікнах:

– `panel%`;

– `pane%`;

– полотна:

– `canvas%` – полотно для малювання на екрані;

– `editor-canvas%` – редактор, допоміжне полотно для відображення текстового редактора або вклеєного редактора;

– елементи керування:

– `message%` – напис;

– `button%` – кнопка;

– `check-box%` – кнопка-прапорець, віконце, яке користувач клацанням миші може вмикати або вимикати; зазвичай кнопку «ввімкнуто», якщо це віконце має вигляд квадрата з буквою X або галочкою всередині; якщо квадрат порожній, то кнопку «вимкнуто»; стан такої кнопки не впливає на інші кнопки в діалоговому вікні;

– `radio-box%` – кнопка із залежною фіксацією; вибір (активізація) такої кнопки (зазвичай позначається точкою всередині неї) визначає одну з взаємовиключних функцій;

– `choice%` – вибір, пункт меню;

– `list-box%` – вікно з елементами списку, користувач обирає один або деякі-

лька елементів (в залежності від стилю списку);

– `text-field%` – текстове поле для введення тексту;

– `combo-field%` – комбінований елемент: поєднує текстове поле з вибором меню;

– `slider%` – бігунок, повзунок на лінійці зі шкалою з числами;

– `gauge%` – індикатор тільки вихідного контролю (користувач не може змінити значення) для подання цілого числа у заданому діапазоні.

На рис. Г.1 показана ієрархія класів графічного інтерфейсу.

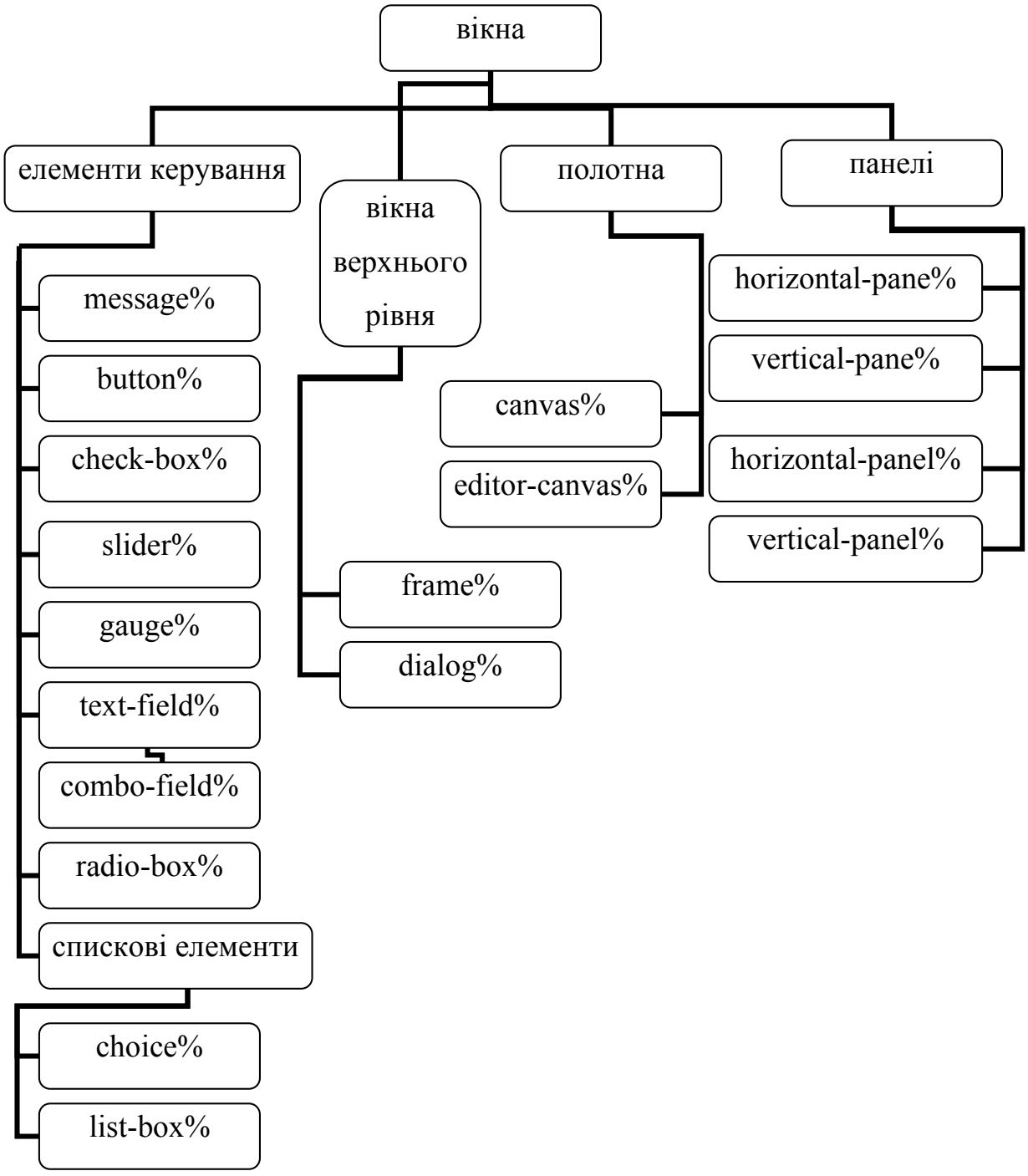


Рис. Г.1. Ієрархія класів графічного інтерфейсу

Додаток Д

Клас «кнопка-прапорець» – `check-box%`

Щоразу, коли користувач натискає кнопку-прапорець, на ній встановлюється / знімається прапорець і викликається функція оберненого виклику, передбачена в якості аргументу при створенні.

Поле	**Пояснення**	**Можливі значення** **# значення за замовчуванням**
обов'язкові поля		
label	заголовок	рядок або об'єкт класу `bitmap%`
parent	батьківський об'єкт	об'єкт класу `frame%`, `dialog%`, `panel%` або `pane%`
необов'язкові поля		
callback	функція оберненого виклику	
style	стиль	список з елементів: `deleted` – видалений # `null`
value	значення	логічний # `#f`
font	шрифт	об'єкт класу `font%` # `normal-control-font`
enabled	ввімкненість	логічний # `#t`
vert-margin	відстань по вертикалі до границі	ціле число (0; 1000) # 2
horiz-margin	відстань по гозизонталі до границі	ціле число (0; 1000) # 2
min-width	мінімальна ширина	ціле число (0; 10000) # мінімальна графічна ширина
min-height	мінімальна висота	ціле число (0; 10000)

Поле	Пояснення	Можливі значення # значення за замовчуванням
		# мінімальна графічна висота
stretchable-widtht	здатність розтягуватись у ширину	логічний # #f
stretchable-height	здатність розтягуватись у висоту (якщо (memq 'multiple style))	логічний # #f

Повідомлення для кнопок-прапорців:

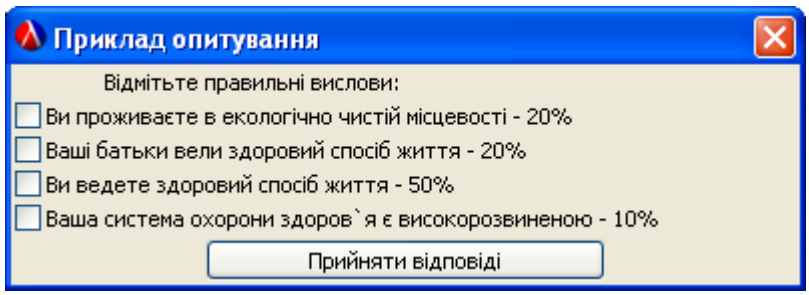
Загальна форма	Дія, результат
(send об'єкт-кнопка-прапорець get-value)	якщо встановлено прапорець – #t, інакше – #f
(send об'єкт-кнопка-прапорець set-value положення)	встановлення/зняття прапорця, в залежності від параметру положення (логічний)

Наприклад:

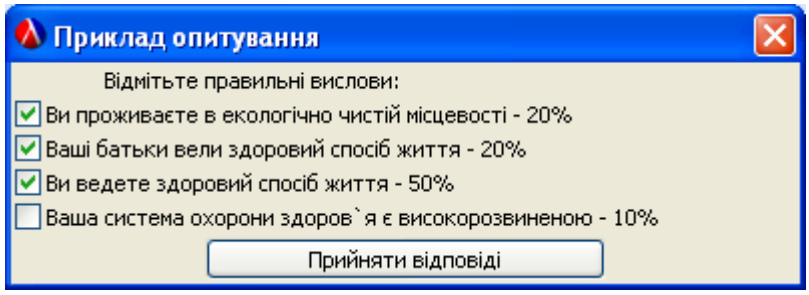
```
; створимо діалогове вікно,
; на якому розмістимо інші об'єкти
; вирівнювання всіх об'єктів,
; для яких воно буде батьківським об'єктом,
; - по центру по-вертикалі та по-горизонталі
; об'єкти на вікні Діалог
; будуть розташовані по вертикалі у тій послідовності,
; в якій вони будуть створені
  (define Діалог (new dialog%
      [label "Приклад опитування"]
      [width 400]
      [height 100]
      [x 100]
      [y 150]
      [alignment '(center center)]))

; створимо на діалозі напис
; спочатку виокремимо на напису місце під 100 знаків
  (define Повідомлення (new message%
      [parent Діалог]
      [label (make-string 100 #\ )]))
```

```
; пошлемо об’єкту Повідомлення повідомлення set-label
; з параметром "Відмітьте правильні вислови:"
  (send Повідомлення set-label "Відмітьте правильні вислови:")

; створимо на діалозі вертикальну панель
;- на ній розмістимо всі кнопки-прапорці
; вирівнювання об’єктів на панелі - зліва по-горизонталі ; та по-центру по-
вертикалі
  (define Панель (new vertical-panel%
      [parent Діалог]
      [alignment '(left center)]))

; кнопки-прапорці розміщуємо на панелі,
; а не на діалозі для того,
; щоб вони були вирівняні зліва
; якщо кнопка-прапорець була натиснута
; непарну кількість раз:
; - збільшуємо на 1 кількість разів натиснення кнопки
; та збільшуємо бали
; інакше (тобто, якщо користувач зняв позначку) –
; зменшуємо бали
  (define Питання1 (new check-box%
       [label "Ви проживаєте в екологічно чистій місцевості - 20%"]
      [parent Панель]
      [callback (lambda (check-box event)
        (if (odd? кількість1)
            (begin
              (set! кількість1 (add1 кількість1))
              (set! бали (+ бали 20)))
            (set! бали (- бали 20))))]))

  (define Питання2 (new check-box%
      [label "Ваші батьки вели здоровий спосіб життя - 20%"]
      [parent Панель]
      [callback (lambda (check-box event)
        (if (odd? кількість2)
            (begin
              (set! кількість2 (add1 кількість2))
              (set! бали (+ бали 20)))
            (set! бали (- бали 20))))]))

  (define Питання3 (new check-box%
      [label "Ви ведете здоровий спосіб життя - 50%"]
      [parent Панель]
      [callback (lambda (check-box event)
        (if (odd? кількість3)
            (begin
              (set! кількість3 (add1 кількість3))
              (set! бали (+ бали 50)))
            (set! бали (- бали 50))))]))

  (define Питання4 (new check-box%
      [label "Ваша система охорони здоров`я є високорозвиненою - 10%"]
```

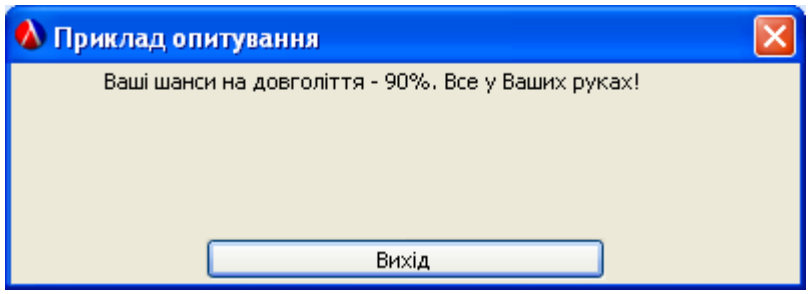
```
      [parent Панель]
      [callback (lambda (check-box event)
        (if (odd? кількість4)
            (begin
              (set! кількість4 (add1 кількість4))
              (set! бали (+ бали 10)))
            (set! бали (- бали 10))))]))

; останньою на діалозі створимо кнопку Кінець
; спочатку її заголовок - "Прийняти відповіді"
; якщо кнопка натиснута вперше (кількість = 1):
; - зникають усі кнопки-прапорці
; - кількість натискань кнопки  - 2
; - її назва змінюється на "Вихід"
; - напис на об’єкті Повідомлення
; - відомості про шанси на довголіття
; якщо кнопка Кінець натиснута двічі (кількість = 2):
; - діалогу посилається повідомлення on-exit
  (define Кінець (new button%
      [parent Діалог]
      [label "Прийняти відповіді"]
      [min-width 200]
      [callback (lambda (button event)
                  (if (= кількість 1)
                      (begin
                        (set! кількість 2)
                        (send Питання1 show #f)
                        (send Питання2 show #f)
                        (send Питання3 show #f)
                        (send Питання4 show #f)
                        (send Кінець set-label "Вихід")
                        (send Повідомлення set-label
            (string-append "Ваші шанси на довголіття - "
                           (number->string бали)
                           "%. Все у Ваших руках!")))
                      (send Діалог on-exit)))]))

; спочатку сума балів 0
 (define бали 0)
; змінна кількість рахує,
; скільки разів натиснута кнопка Кінець
  (define кількість 1)
; змінна кількістьі рахує,
; скільки разів натиснута кнопка-прапорець Питанняі
; (і - від 1 до 4)
  (define кількість1 1)


  (define кількість2 1)
  (define кількість3 1)
  (define кількість4 1)
; відобразимо Діалог
  (send Діалог show #t)
```



а) початковий вигляд



б) вибір вірних висловів



в) показ результатів

Рис. Д.1. Використання кнопок-прапорців

Повідомлення для вікон:

Загальна форма	**Дія, результат**
`(send об'єкт-вікно enable ввімкненість?)`	вмикання / вимикання вікна залежно від параметру ввімкненість?; коли вікно вимкнене повідомлення для нього ігноруються (це стосується і всіх об'єктів, для яких вікно є батьківським)

Загальна форма	Дія, результат
(send об’єкт-вікно focus)	вікну надається фокус введення
(send об’єкт-вікно get-cursor)	вікну надається курсор
(send об’єкт-вікно get-height)	ціле число (0; 10000) – висота вікна
(send об’єкт-вікно get-label)	рядок – заголовок вікна
(send об’єкт-вікно get-size)	розміри вікна
(send об’єкт-вікно get-x)	ціле число (-10000 10000) – позиція (по вісі x) лівого краю на екрані або на батьківському об’єкті
(send об’єкт-вікно get-x)	ціле число (-10000 10000) – позиція (по вісі y) верхнього краю на екрані або на батьківському об’єкті
(send об’єкт-вікно is-enabled?)	#t – якщо ввімкнене вікно і всі його батьківські об’єкти, інакше – #f
(send об’єкт-вікно is-shown?)	#t – якщо показане вікно і всі його батьківські об’єкти, інакше – #f
(send об’єкт-вікно on-focus надати?)	надходить, коли вікно отримує або втрачає фокус введення (#t – якщо отримує і #f – якщо втрачає)
(send об’єкт-вікно on-move x y)	надходить при переміщенні вікна
(send об’єкт-вікно refresh)	перерисування вікна
(send об’єкт-вікно set-label рядок)	надання заголовку вікна значення рядок
(send об’єкт-вікно get-label)	рядок – заголовок вікна

Додаток Е

Анкета для виявлення початкового рівня сформованості компетентностей з програмування

Оберіть та підкресліть одну із запропонованих відповідей:

1. Чи вивчали Ви програмування?

а) ні б) намагався в) так

2. З наведеного списку оберіть, де Ви найбільше поглибили рівень знань з програмування:

а) на уроках інформатики б) на факультативних заняттях

в) під час самостійних занять г) на додаткових курсах

3. Оцініть рівень власної компетентності з програмування:

а) низький б) достатній в) середній г) високий

4. Ви:

а 1. знаєте основні конструкції мови програмування

а) ні б) деякі в) всі

а 2. знаєте основні типи даних

а) ні б) деякі в) всі

а 3. знаєте основні етапи розробки програми

а) ні б) деякі в) всі

а 4. знаєте способи визначення змінних та функцій

а) ні б) деякі в) всі

б 1. вмієте визначати змінні та функції

а) ні б) деякі в) всі

б 2. вмієте утворити похідні типи даних із простих

а) ні б) деякі в) всі

б 3. вмієте створити документацію до програми

а) ні б) деякі в) всі

б 4. вмієте налагодити програму

а) ні б) деякі в) всі

в 1. програмуєте на тиждень протягом

а) менше 1 год. б) 2 год. в) більше 2 год.

в 2. створили корисних програм

а) жодної б) 1–2 в) більше двох

в 3. плануєте пов’язати своє життя з програмуванням

а) ні б) можливо в) так

в 4. прагнете опанувати новими середовищами програмування

а) ні б) так в) хочу створити власне

г 1. співпрацюєте з іншими студентами в процесі створення програми

а) ніколи б) інколи в) завжди

г 2. дотримуєтесь думки про необхідність розподілу обов’язків в команді програмістів

а) ні б) так в) сам розподіляю

г 3. користуєтесь засобами миттєвого зв’язку

а) ні б) для спілкування в) для спілкування та співпраці

г 4. в команді програмістів взяли б на себе роль

а) виконавця б) постановника завдання в) менеджера

Дякуємо за обдумані відповіді!