ЧЕРКАСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ БОГДАНА ХМЕЛЬНИЦЬКОГО

На правах рукопису

Гришко Людмила Веніамінівна

УДК 378.147:004.25(07)

МЕТОДИЧНА СИСТЕМА НАВЧАННЯ ОСНОВ ПРОГРАМУВАННЯ МАЙБУТНІХ ІНЖЕНЕРІВ-ПРОГРАМІСТІВ

13.00.02 - теорія і методика навчання (інформатика)

Дисертація на здобуття наукового ступеня
кандидата педагогічних наук

Науковий керівник
Триус Юрій Васильович
доктор педагогічних наук, професор

Черкаси – 2009

ЗМІСТ

ВСТУП..... 6

РОЗДІЛ 1

ТЕОРЕТИЧНІ ТА ПСИХОЛОГО-ПЕДАГОГІЧНІ ОСНОВИ ФАХОВОЇ ПІДГОТОВКИ МАЙБУТНІХ ІНЖЕНЕРІВ-ПРОГРАМІСТІВ..... 18

1.1. Роль і місце курсу з основ програмування в системі підготовки фахівців у галузі комп'ютерних наук..... 18

1.2. Особливості навчання дисциплін циклу комп'ютерних наук у ВНЗ України та за кордоном..... 19

1.3. Вимоги до професійних якостей майбутніх інженерів-програмістів..... 31

1.4. Психолого-педагогічні основи навчання основ програмування..... 37

1.4.1. Психологічні основи навчання основ програмування..... 37

1.4.2. Діяльністний підхід у навчанні основ програмування..... 48

1.4.3. Компетентністний підхід у навчанні основ програмування..... 51

1.4.4. Диференційований підхід у навчанні основ програмування..... 60

1.4.5. Модульно-рейтингова система навчання основ програмування..... 62

Висновки до розділу 1..... 67

РОЗДІЛ 2

МЕТОДИЧНА СИСТЕМА НАВЧАННЯ ОСНОВ ПРОГРАМУВАННЯ..... 69

2.1. Мета, завдання і функції навчання курсу з основ програмування..... 69

2.2. Зміст курсу з основ програмування…..... 76

2.2.1. Принципи визначення змісту курсу з основ програмування для майбутніх інженерів-програмістів..... 77

2.2.2. Теоретичні основи курсу..... 79

2.2.3. Зміст лабораторного практикуму..... 80

2.3. Сучасні принципи, підходи і методи навчання основ програмування..... 82

2.3.1. Принципи навчання основ програмування..... 82

2.3.2. Сучасні підходи до навчання основ програмування 89

2.3.2.1. Задачний підхід у навчанні основ програмування.... 89

2.3.2.2. Навчання стилю програмування, як складова формування професійної культури майбутніх інженерів-програмістів..... 93

2.3.2.3. Диференційований підхід як засіб підвищення мотивації студентів при навчанні основ програмування..... 102

2.3.2.4. Індивідуалізація процесу навчання основ програмування..... 105

2.3.3. Методи навчання основ програмування..... 106

2.3.3.1. Традиційні методи навчання основ програмування 106

2.3.3.2. Метод парного програмування…….. 110

2.3.3.3. Колективний проект як метод активного навчання основ програмування..... 111

2.3.3.4. Метод паралельного навчання мов програмування..... 116

2.4. Засоби навчання основ програмування……..... 120

2.4.1. Аналіз підручників, посібників і збірників задач з основ програмування..... 120

2.4.2. Система завдань з основ програмування як засіб підвищення практичної значущості результатів навчання..... 125

2.4.3. Використання інформаційно-комунікаційних технологій при навчанні основ програмування..... 144

2.4.4. Системи програмування..... 151

2.5. Форми організації навчальної діяльності студентів при навчанні основ програмування..... 152

2.6. Елементи модульно-рейтингової системи навчання основ програмування..... 159

2.7. Експериментальна перевірка ефективності розроблених компонентів методичної системи навчання основ програмування..... 161

2.7.1. Організація і проведення педагогічного експерименту... 161

2.7.2. Аналіз результатів педагогічного експерименту..... 167

Висновки до розділу 2..... 173

ВИСНОВКИ..... 175

ДОДАТКИ..... 179

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ……..... 255

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ВНЗ	- вищій навчальний заклад;
ПВНЗ	- педагогічний вищій навчальний заклад;
ППЗ	- педагогічні програмні засоби

ВСТУП

Актуальність дослідження. Інформаційно-комунікаційні технології, які сьогодні використовуються практично в усіх сферах людської діяльності, стали невід'ємною частиною сучасного суспільства, рушійною силою його економічного зростання. Існуючі тенденції розвитку інформаційного суспільства обумовлюють зростання потреби у фахівцях у галузі комп'ютерних наук. Якщо в 1970 році, за даними робочої групи Міжнародної федерації з обробки інформації (IFIP) під егідою ЮНЕСКО [84], професіонали в галузі комп'ютерних наук складали 0,5% від числа всіх фахівців, то на початок нового тисячоліття це співвідношення становить близько 4% і тенденція на збільшення частки цих фахівців зберігається. Також, якщо в перші 35 років існування обчислювальних машин основна увага приділялася розвитку апаратних засобів і, зокрема, наприкінці 50-х років XX-го століття 90% витрат становили витрати на апаратуру, то сьогодні акцент змішується в бік розробки програмного забезпечення і саме на розробку програмного забезпечення витрачається близько 90% всіх коштів [223]. Більш того, „вартісні показники галузей, що виробляють програмне забезпечення, стають визначальними для економіки – як національної, так і міжнародної" [191]. Тому перед системою вищої освіти нашої держави стоїть задача забезпечення зростаючих потреб суспільства високопрофесійними фахівцями в галузі інформаційно-комунікаційних технологій, зокрема, інженерами-програмістами. Від інтелекту програміста, його готовності до інтелектуально-творчої діяльності, від якості рішень, що приймаються програмістом, залежить „інтелект суспільства" [190].

Основою теоретичних знань, практичних умінь і навичок, які використовують у своєї професійної роботі програмісти, є комп'ютерні науки. Фундаментом для навчання комп'ютерних дисциплін з циклу професійної і практичної підготовки є курс програмування, який вивчає алгоритми і програми; їхні властивості; методи побудови алгоритмів та способи їх подання в ЕОМ; питання, що пов'язані з аналізом алгоритмів і доведенням їхньої

правильності, побудовою структур даних та алгоритмами їх опрацювання. Поняття „програмування” поєднує в собі розуміння поставленої проблеми; написання програми, знаходження помилок у програмі (налагодження); перевірку того, що програма задовольняє своєму призначенню (тестування); зміну програми для того, щоб вона була придатна для розв’язування іншого завдання (модифікація); оволодіння новими навичками й знаннями програмування (навчання) [73], [121], [191], [223]. Метою програмування є „розробка програм для розв’язування поставлених задач” [183].

Вітчизняна вища школа має значний досвід щодо підготовки інженерів-програмістів. Разом із тим, сьогодні в цій галузі освіти залишається багато невирішених проблем. Серед проблем, що спричиняють зниження якості підготовки інженерів-програмістів, їх конкурентоспроможності на міжнародному ринку праці, слід зазначити наступні:

– певна неузгодженість освітньо-кваліфікаційних характеристик та освітньо-професійних програм державних стандартів з напрямів підготовки „Комп'ютерні науки”, „Прикладна математика”, „Програмна інженерія” з світовими та європейськими стандартами підготовки фахівців у цій галузі;

– недостатня кількість україномовної навчальної і методичної літератури з фахових дисциплін, що вивчаються в межах комп'ютерних спеціальностей;

– недостатня кількість педагогічного програмного забезпечення для підтримки процесу навчання дисциплін з комп’ютерних наук, зокрема, з основ програмування;

– значне розшарування рівня знань зі шкільного курсу інформатики у студентів молодших курсів;

– низький рівень навчально-пізнавальної активності студентів;

– невміння й, часто, небажання студентів працювати самостійно, тощо.

Як показали опитування вчителів інформатики і учнів, викладачів комп’ютерних дисциплін і студентів комп’ютерних спеціальностей, основними причинами, що обумовили виникнення цих проблем, є:

– відсутність гнучких механізмів управління освітою, які б швидко реагували на зміни, що постійно відбуваються в галузі інформаційно-комунікаційних технологій, і, як наслідок, *невідповідність державних стандартів сучасному рівню розвитку цієї галузі*;

– *незадовільний стан навчання інформатики в середніх навчальних закладах,* пов'язаний з недостатньою кількістю і плинністю висококваліфікованих вчителів інформатики, які йдуть працювати в бізнес-структури;

– превалювання у вищій школі традиційних методів навчання і *обмежене застосування новітніх педагогічних підходів, технологій і методів навчання,* зокрема, компетентістного, особистістно-орієнтованого та диференційованого підходів, проблемного і продуктивного навчання;

– недостатній рівень використання нових інформаційно-комунікаційних технологій у навчанні комп'ютерних дисциплін, зокрема, програмування.

Як свідчить світовий досвід, досягнення якісно нового рівня в підготовці фахівців із вищою освітою неможливо без забезпечення розвитку вищої школи на основі нових прогресивних концепцій, науково-методичних досягнень, запровадження сучасних педагогічних та інформаційно-комунікаційних технологій.

Одним із реальних шляхів підвищення якості професійної підготовки майбутніх інженерів-програмістів на рівні ВНЗ, на нашу думку, є розробка науково-обґрунтованих методичних систем навчання (МСН) з фахових дисциплін, які б сприяли активізації навчально-пізнавальної, науково-дослідної діяльності студентів, розкриттю їхнього творчого потенціалу, збільшенню ролі самостійної та індивідуальної роботи і ґрунтувалися б на широкому впровадженні в навчальний процес новітніх педагогічних та інформаційних технологій ([24], [202]).

Аналіз робіт у галузі загальної психології, вікової психології, дидактики (Л.С. Виготський [39], О.М. Леонтьєв [115, 116], С.Л. Рубінштейн [173], Н.Ф. Тализіна [196, 197] та ін.) показує, що навчання, як основне джерело

розумового розвитку людини, може бути розвиваючим за умови тісної взаємодії учасників навчального процесу – вчителя і учня.

Роботи, присвячені питанням формування прийомів розумової діяльності, дидактичним закономірностям формування знань, умінь і навичок (Ю.К. Бабанський [15-18], П.Я. Гальперін [40-41], З.І. Слєпкань [186, 187], Н.Ф. Тализіна [196, 197], І.С. Якиманська [228] та ін.), надають можливість осмислити існуючі проблеми щодо методів, прийомів, організаційних форм навчального процесу.

У даний час у вітчизняній системі вищої освіти спостерігається тенденція до зменшення аудиторного навантаження з одночасним збільшенням частки самостійної навчальної діяльності студентів. Цей підхід вимагає таких форм навчальної роботи, що активізують пізнавальну діяльність студентів, їхню самостійність. Саме активізація пізнавальної діяльності студентів є одним із пріоритетних напрямів досліджень педагогіки вищої школи.

Різні аспекти проблеми активізації пізнавальної діяльності студентів розглядали А.М. Алексюк [3], С.І. Архангельський [12], Ю.К. Бабанський [16-18], Є.Н. Кабанова-Меллер[87], С.Л. Рубінштейн [173], З.С. Сейдаметова [178], С.О. Семеріков [179], Н.Ф. Тализіна [196], Т.І. Шамова [244] та інші.

Дослідження вчених (М.І. Жалдак [76], Н.В. Морзе [134], С.О. Семеріков [179], Ю.В. Триус [201] та ін.) присвячені питанням використання ІКТ у навчальному процесі середньої і вищої школи, свідчать про позитивний вплив ІКТ на активізацію пізнавальної діяльності учнів та студентів як при вивченні окремих навчальних предметів, так і в навчальному процесі в цілому.

Останнім часом з розвитком інформаційних технологій у світі поширюється мобільне навчання. С.О. Семеріков визначає мобільне навчання «як сучасний напрямок розвитку систем дистанційної освіти із застосуванням мобільних телефонів, смартфонів, КПК, електронних книжок. … В порівнянні з традиційним мобільне навчання надає можливість моніторингу навчання в реальному часі та забезпечує високу насиченість контенту, що дозволяє

розглядати його не лише як засіб навчання, а й як інструмент спільної роботи, призначений для підвищення якості навчання» [180].

Окремим аспектам навчання основ програмування у вищій школі присвячені роботи А.М. Гуржія, М.С.Львова, О.В. Співаковського [66], А.П. Єршова [73], М.І. Жалдака [75], А.Г. Кушніренка, Г.В. Лебедєва [114], В.М. Заварикіна, В.Г. Житомирського, М.П. Лапчика [77], А.В. Могільова, М.І. Пака, Є.К. Хьоннера [130], Л.Ф. Монастирського, Д.Г. Румянцева [174], З.С. Сейдаметовой [178], С.О. Семерікова [179], та ін., дисертаційні роботи Т.Ю. Китаєвської [92], Ф.В. Толкачова [199]. Слід зазначити, що у своїй більшості дослідження і роботи присвячені питанням навчання інформатики і програмуванню майбутніх вчителів інформатики.

Питанням підготовки майбутніх інженерів-програмістів присвячена робота З.С. Сейдаметовой [178]. В роботі наведено ретельний аналіз документа СС2001, також проаналізовані ОКХ та ОПП для напряму «Прикладна математика». В результаті зроблено висновок про «фундаментальні відмінності в підходах» освітніх стандартів вищої освіти України і СС2001 [178]. Автор зазначає, що «Дуже важливими для розвитку професійних навичок є дисципліни програмістского циклу», до яких відносить і дисципліну «Програмування».

Разом із тим у результаті наших досліджень зроблено висновок про те, що поза увагою дослідників залишаються питання методики навчання основ програмування майбутніх інженерів-програмістів.

В Україні підготовка фахівців у галузі інформаційних і телекомунікаційних технологій у ВНЗ здійснюється за такими напрямами освітньої діяльності (згідно переліку напрямів, за якими здійснюється підготовка фахівців у вищих навчальних закладах за освітньо-кваліфікаційним рівнем бакалавра [162]):

6.040301 – Прикладна математика

6.040302 – Інформатика

6.050101 – Комп'ютерні науки

6.050102 – Комп'ютерна інженерія

6.050103 – Програмна інженерія

6.050201 – Системна інженерія

6.050202 – Автоматизація та комп'ютерно-інтегровані технології

6.170101 – Безпека інформаційних і комунікаційних систем

6.170103 – Управління інформаційною безпекою

В освітньо-професійній програмі підготовки бакалавра кожного із цих напрямів є курс з програмування, засвоєння якого передбачається на першому та другому році навчання. Наприклад, у галузевих стандартах вищої освіти з напряму „Прикладна математика" цей курс включено до циклу професійної підготовки і має назву „Програмування" [143], у напрямі „Комп'ютерні науки" – до циклу природничо-наукової підготовки і має назву „Основи програмування та алгоритмічні мови" [144], у напрямі „Комп'ютерна інженерія" цей курс включено до циклу природничо-наукової підготовки і також має назву „Програмування" [145]. Надалі називатимемо цей курс „Основи програмування".

Оскільки саме при навчанні основ програмування закладається як теоретична, так і практична база підготовки майбутніх інженерів-програмістів, *актуальною є проблема визначення особливостей і створення науково-обґрунтованої методичної системи навчання основ програмування майбутніх інженерів-програмістів,* яка б сприяла активізації навчально-пізнавальної, науково-дослідної діяльності студентів, розкриттю їхнього творчого потенціалу, розвитку самостійності та індивідуальних здібностей особистості й ґрунтувалася на широкому впровадженні у навчальний процес новітніх педагогічних та інформаційно-комунікаційних технологій і враховувала сучасні вимоги до підготовки фахівців у галузі інформаційно-комунікаційних технологій і комп'ютерних наук.

Вище названі чинники зумовили вибір теми нашого дослідження: „Методична система навчання основ програмування майбутніх інженерів-програмістів".

Зв'язок роботи з науковими програмами, планами, темами. Обраний напрямок досліджень входить до плану науково-дослідних робіт Черкаського національного університету імені Богдана Хмельницького як складова колективної теми кафедри прикладної математики (протокол №7 від 28 лютого 2002 року засідання вченої ради Черкаського державного університету) та погоджена в бюро Ради з координації наукових досліджень в галузі педагогіки та психології в Україні (протокол № 8 від 29 жовтня 2002 р.).

Об'єктом дослідження є процес навчання основ програмування майбутніх інженерів-програмістів.

Предметом дослідження є методична система навчання основ програмування студентів комп'ютерних спеціальностей.

Мета дослідження полягає в створенні науково-обґрунтованої методичної системи навчання основ програмування майбутніх інженерів-програмістів, яка сприяє активізації навчально-пізнавальної, пошуково-дослідної діяльності студентів, розкриттю їх творчого потенціалу, розвитку самостійності та індивідуальних здібностей особистості й ґрунтується на широкому впровадженні в навчальний процес новітніх педагогічних та інформаційно-комунікаційних технологій.

Відповідно до мети дослідження поставлені наступні *завдання:*

1. Узагальнити вітчизняний та зарубіжний досвід навчання основ програмування у ВНЗ на комп'ютерних спеціальностях.

2. Уточнити мету і зміст навчання основ програмування для спеціальностей „Програмне забезпечення автоматизованих систем" (напрям підготовки „Програмна інженерія") та „Прикладна математика" (напрям підготовки „Прикладна математика") на сучасному етапі розвитку інформатики, комп'ютерної техніки та інформаційно-комунікаційних технологій; визначити методи, засоби та організаційні форми навчання основ програмування, які сприяють інтелектуальному розвитку студентів, формуванню в них активної життєвої і професійної позиції, навичок самостійної роботи.

3. Визначити умови та можливі шляхи реалізації диференційованого підходу при навчанні основ програмування студентів комп'ютерних спеціальностей у ВНЗ.

4. Виявити і реалізувати шляхи активізації навчально-пізнавальної діяльності студентів молодших курсів, яка орієнтована на поглиблення та розширення теоретичних знань з основ програмування, підвищення якості набутих навичок розв'язування практичних задач та вміння їх застосовувати при вирішенні професійних завдань.

5. Розробити методику організації і проведення самостійної роботи студентів із використанням новітніх інформаційно-комунікаційних технологій навчання.

6. Створити ефективну систему контролю знань, умінь і навичок студентів з основ програмування на основі модульно-рейтингової системи навчання.

7. Організувати експериментальну перевірку результативності запропонованих компонентів методичної системи навчання основ програмування майбутніх інженерів-програмістів; розроблених методичних, навчальних друкованих і електронних матеріалів та педагогічних програмних засобів.

Для розв'язання поставлених завдань застосовано такі *методи досліджень:*

– теоретичні: аналіз наукової, психолого-педагогічної та навчально-методичної літератури з проблеми дослідження; вивчення досвіду навчання основ програмування для студентів комп'ютерних спеціальностей в Україні та за кордоном;

– емпіричні: педагогічне спостереження навчального процесу; анкетування, бесіди зі студентами та викладачами; тестування студентів; аналіз досвіду роботи викладачів комп'ютерних дисциплін в Україні та за кордоном; педагогічний експеримент (констатувальний, пошуковий і формувальний етапи) зі статистичним аналізом його результатів.

Наукова новизна одержаних результатів визначається тим, що:

– *вперше запропоновано науково обґрунтовану і експериментально апробовану методичну систему навчання основ програмування студентів комп'ютерних спеціальностей, в якій враховано Рекомендації Комп'ютерного товариства Інституту Інженерів з Електротехніки і Електроніки та Асоціації з Обчислювальної Техніки щодо змісту університетських програм навчальних дисциплін у галузі інформатики* і яка орієнтована на широке використання в навчальному процесі новітніх педагогічних та інформаційно-комунікаційних технологій;

–*визначено* принципи та засади побудови навчального курсу з основ програмування для комп'ютерних спеціальностей ВНЗ (п. 2.2.1);

–*уточнено* зміст компетентностей студентів комп'ютерних спеціальностей з основ програмування (п. 1.4.3);

–*запропоновано* класифікацію навчальних завдань з основ програмування, яка орієнтована на вид і характер діяльності студентів (п. 2.4.2);

–*визначено* способи активізації навчально-пізнавальної діяльності студентів, їх творчої самостійної та індивідуальної роботи, які орієнтовані на поглиблення та розширення теоретичних знань з основ програмування, підвищення якості набутих навичок розв'язування практичних задач та вміння застосовувати їх при вирішенні професійних завдань.

Практичне значення одержаних результатів визначається тім, що**:**

– *розроблено* і *реалізовано* на практиці компоненти методичної системи навчання основ програмування (цілі, зміст, методи, засоби та організаційні форми навчання) для майбутніх інженерів-програмістів;

–*розроблено* навчальну та робочу програми з основ програмування (п. 2.2.1);

–*розроблено та обґрунтовано* методику створення студентами навчального програмного продукту (п. 2.3.3.3);

–*створено* лекційний курс і лабораторний практикум з основ програмування та методичний посібник «Початки програмування»;

–*розроблено* педагогічний програмний засіб «Linears Rolls» для моделювання роботи зі списками та електронний посібник «Програмування мовою С».

Результати дослідження можуть бути використані викладачами ВНЗ при навчання алгоритмізації і програмування студентів спеціальностей, навчальними планами яких передбачено вивчення курсу з програмування, а також вчителями інформатики, які працюють у школах фізико-математичного, природничого та технологічного профілів, для поглибленого навчання учнів основ програмування.

Особистий внесок здобувача полягає у розробці компонентів методичної системи навчання основ програмування студентів комп'ютерних спеціальностей, яка орієнтована на широке використання в навчальному процесі новітніх педагогічних та інформаційно-комунікаційних технологій.

Результати дослідження впроваджено під час експериментального навчання студентів спеціальностей „Програмне забезпечення автоматизованих систем" та „Прикладна математика" Черкаського національного університету імені Богдана Хмельницького, спеціальності „Інформаційні управляючі системи і технології" Черкаської філії ПВНЗ „Європейський університет" (довідка №2322/01 від 12.09.2007 р.), спеціальності „Інформатика" Полтавського державного педагогічного університету імені В.Г. Короленка (довідка №1285/01-37/02 від 03.04.2009 р.).

Особистий внесок здобувача у роботах в співавторстві:

1. У роботі [24] здобувачем визначено мету пропедевтичного курсу з програмування у підготовці майбутніх інженерів-програмістів.

2. У роботі [64] здобувачем висвітлено проблему створення інструментальних програмних засобів, які призначені для надання можливості суб'єктам навчального процесу розв'язувати задачі з програмування.

3. У роботі [65] здобувачем запропоновано принципи розробки генератора індивідуальних завдань з програмування.

4. У роботі [202] здобувачем визначено завдання курсу з основ програмування у вищій школі при підготовці фахівців з комп’ютерних спеціальностей.

Здобувач брав участь в обговоренні і аналізі результатів та написанні текстів всіх статей, що написані у співавторстві.

Результати дослідження доповідались на Першій Всеукраїнській конференції молодих науковців „Інформаційні технології в науці та освіті” (Черкаси, 1997); на Другій Всеукраїнській конференції молодих науковців „Інформаційні технології в науці та освіті” (Черкаси,); на Всеукраїнській науково-методичній конференції „Теорія та методика навчання фундаментальних дисциплін у вищій технічній школі” (Кривий Ріг, 2003); на V Всеукраїнській науково-практичній конференції „Комп’ютерне моделювання та інформаційні технології в науці, економіці та освіті” (Черкаси, 2003); на IV науково-методичному семінарі „Інформаційні технології в навчальному процесі” (Одеса, 2003); на IV Всеукраїнській конференції молодих науковців „Інформаційні технології в науці, освіті і техніці” (Черкаси, 2004); на Всеукраїнській науково-практичній конференції „Інформатика та комп'ютерна підтримка навчальних дисциплін у середній та вищій школі” (Бердянськ, 2004); на V Всеукраїнській конференції молодих науковців „Інформаційні технології в науці, освіті і техніці” (Черкаси, 2006); на науково-практичній конференції „Громадянське суспільство України у площині економічного, правового, політичного та духовного розвитку” (Черкаси, 2006); на Всеукраїнській науково-методичній конференції „Проблеми математичної освіти” (Черкаси, 2007), на VI Всеукраїнській конференції молодих науковців „Інформаційні технології в науці, освіті і техніці” (Черкаси, 2008); на семінарах кафедри теоретичних основ інформатики Національного педагогічного університету імені М. П. Драгоманова; на науково-практичних семінарах кафедр інформатики та сучасних інформаційних технологій і прикладної математики Черкаського національного університету імені Богдана Хмельницького.

Апробація результатів дисертаційного дослідження проводилась під час експериментального навчання студентів спеціальностей „Програмне забезпечення автоматизованих систем” та „Прикладна математика” Черкаського національного університету імені Богдана Хмельницького, спеціальності „Спеціалізовані комп’ютерні системи” Інституту Соціального Управління економіки і права (м. Черкаси), спеціальності „Інформаційні управляючі системи і технології” Черкаської філії ПВНЗ „Європейський університет”, спеціальності „Інформатика” Полтавського державного педагогічного університету імені В.Г. Короленка.

Публікації. За темою дослідження опубліковано 16 наукових праць, з них 1 методичний посібник [58], 5 статей – у фахових виданнях ВАК України ([55], [57], [58], [64], [65]), 10 робіт – у збірниках матеріалів і тез конференцій ([24], [52-54], [56], [60-62], [65], [202]).

Структура дисертації. Дисертаційна робота складається із вступу, 2 розділів, висновків, списку використаних джерел (236 найменування) та 10 додатків (обсягом 77 сторінок). Загальний обсяг тексту становить 276 сторінок, основний текст викладений на 178 сторінках, робота містить 21 таблицю, з яких додатки містять 6 таблиць, та 18 рисунків.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ТА ПСИХОЛОГО-ПЕДАГОГІЧНІ ОСНОВИ ФАХОВОЇ ПІДГОТОВКИ МАЙБУТНІХ ІНЖЕНЕРІВ-ПРОГРАМІСТІВ

1.1. Роль і місце курсу з основ програмування в системі підготовки фахівців у галузі комп'ютерних наук

Аналіз державних стандартів вищої освіти, навчальних планів і робочих навчальних програм зі спеціальностей, де здійснюється підготовка фахівців у галузі інформатики, прикладної математики, комп'ютерної інженерії, інформаційно-телекомунікаційних технологій показує, що знання, вміння, і навички, здобуті студентами з основ програмування, потрібні їм при вивченні дисциплін циклу фундаментальних, професіонально-орієнтованих дисциплін і дисциплін спеціалізації.

Як зазначає С.І. Архангельській: „... кожний навчальний предмет у вищий школі має подвійну значимість – внутрішню змістову і прикладну, яка розповсюджується на інші види навчання і практику" [12]. Дійсно, наприклад, для засвоєння навчального матеріалу дисциплін „Об'єктно-орієнтоване програмування", „Комп'ютерна математика", „Алгоритми: розробка та аналіз", „Інженерна графіка", „Чисельні методи", „Чисельні методи математичної фізики", „Сучасні технології програмування", „Обчислювальна геометрія та комп'ютерна графіка", „Теорія алгоритмів", „Комп'ютерне моделювання складних систем", „Комп'ютерне моделювання фізичних процесів" необхідне вміння програмувати однією з мов програмування високого рівня, зокрема, мовою Pascal або C. Вміння читати і розуміти готові програми і алгоритми необхідні у курсі „Системне програмування", „Системне програмне забезпечення", „Операційні системи", „Основи системного аналізу об'єктів і процесів комп'ютеризації", „Теорія інформації і кодування", „Програмування на Асемблері". Дисципліни „Проектування і розробка компіляторів", „Програмне забезпечення ЕОМ", „Теорія захисту даних в інформаційних

системах”, „ЕОМ і мікропроцесорні системи”, „Паралельні обчислювальні процеси” передбачають знання студентами мови програмування С. Розуміння аспектів проектування програмного забезпечення, яке закладається в курсі з основ програмування, потрібне в курсі „Організація баз даних”, „Комп’ютерні мережі”, „Методологія розробки програмних продуктів та програмних систем”, „Введення в сучасні інформаційні технології”, „Засоби мультимедіа у нових інформаційних технологіях” та ін.

Підсумовуючи вищесказане можна зробити висновок: *курс з основ програмування є ключовим компонентом системи підготовки фахівців у галузі комп’ютерних наук*, головним завданням якого є забезпечення теоретичного базису й практичних умінь і навичок у професійній підготовці майбутніх програмістів. Тому знання основ програмування є необхідною умовою для засвоєння більшості дисциплін комп’ютерних наук, оскільки:

– ця галузь знань складається з тих концепцій, які важливі для практики програмування незалежно від парадигми програмування, яка використовується у навчальному процесі;

– успішне засвоєння студентами навчального матеріалу з комп'ютерних дисциплін і, як наслідок, їхня майбутня професійна діяльність залежить від якості засвоєння навчального матеріалу курсу з основ програмування;

– курс з основ програмування є вступним курсом [230] і повинен вивчатися на молодших курсах, перед вивченням усіх інших дисциплін з циклу комп’ютерних наук.

1.2. Особливості навчання дисциплін циклу комп’ютерних наук у ВНЗ України та за кордоном

Структура і зміст вищої освіти істотно відрізняється в різних країнах. Зупинимося на деяких особливостях навчання дисциплін циклу комп’ютерних наук у різних ВНЗ Європи і США.

Освітні стандарти США регулюються такими нормативними документами, як „Акт про освіту в цілях зміцнення економічної безпеки", „Laws for Education. Department of Education, Wash., D.C., 1990", „Америка у 2000 році: стратегія освіти" (1991), Програмна концепція пріоритетних цілей освіти Національної асоціації губернаторів США „Цілі 2000: досягнення національних цілей освіти" (1991) [182].

Для освітніх стандартів США характерні такі особливості:

– стандарт розглядається як сукупність кінцевих і проміжних цілей у галузі реформування освіти;

– стандарт передбачає досягнення фіксованих показників;

– випускні випробування проводяться по завершенню кожного освітнього рівня;

– здійснюється фундаменталізація основних академічних предметів із вказівкою на неприпустимість їхнього прискореного вивчення;

– стандарти спрямовані на уникнення безсистемності, випадковості у виборі навчальних предметів.

Навчальний план 4-річної підготовки бакалавра складається з 120 кредитів. Звичайна практика, коли одна навчальна дисципліна оцінюється в три кредити, тобто навчальний план бакалавра включає до 40 дисциплін, з них загальна гуманітарна підготовка становить близько 20%. Більшість студентів освоюють навчальну програму за 5-6 років. На перших курсах студент має право вибору навчальних дисциплін, які не мають відношення до обраної програми підготовки.

Навчальні плани підготовки є індивідуальними для кожного студента й складаються за його бажанням, але під керівництвом викладача-наставника. При цьому не існує загального правила, відповідно до якого студент повинен визначитися з напрямом підготовки. Крім того, „ ... американський абітурієнт поступає не на конкретний факультет того або іншого університету, а до університету або коледжу. Зазвичай, лише наприкінці другого або навіть

третього року навчання студент обирає свою спеціалізацію” [230]. За семестр студент повинен засвоїти навчальний матеріал 4-5 навчальних дисциплін, складаючи після виконання програм екзамен.

Основою навчального процесу в американських вищих навчальних закладах є самостійна робота студента. Тому організації самостійної роботи студента приділяється дуже велика увага: доступ до комп’ютерних класів для студентів не обмежений за часом; викладачами вищих навчальних закладів розроблені підручники з усіх дисциплін циклу комп’ютерних наук; існують бази даних із конспектами навчального матеріалу в електронному вигляді; студенти мають вільний доступ до мережі Internet.

Навчальне навантаження студента США складається з аудиторної, до 15 годин на тиждень, і самостійної роботи - до 45 годин на тиждень, тобто співвідношення аудиторного навантаження й самостійної роботи – 1:3.

Аудиторні заняття тривають від 50 хвилин до 1 години.

У деяких вищих навчальних закладах США, які готують фахівців з комп’ютерних наук, передбачені завершальні проектні курси, які тривають один-два семестри. У цей час студенти розробляють проект, працюючи в команді. Проект повинен враховувати значущі для реальних задач чинники, такі як вартість, безпеку, продуктивність, задоволеність замовників цього проекту і його користувачів. Проект може розроблятися виключно в навчальних цілях або ж мати реальних замовників, якими можуть бути, наприклад, інші факультети ВНЗ.

Аналіз існуючих державних стандартів і навчальних планів деяких ВНЗ нашої країни з напрямів підготовки „Комп'ютерні науки” і „Прикладна математика” показує, що навчання програмування майбутніх інженерів-програмістів обмежується часто курсом з основ програмування та курсом з об'єктно-орієнтованого програмування. Як правило, у лабораторних практикумах цих дисциплін студенти розв’язують невеликі навчальні задачі, тексти програм яких не перевищують 150-200 рядків. В окремих випадках

навчальні плани окремих ВНЗ у циклі вибіркових дисциплін містять курси, присвячені інженерії програмного забезпечення. На старших курсах державними стандартами передбачене виконання курсових, кваліфікаційних і дипломних робіт. Традиційно ці види робіт передбачають лише індивідуальну роботу студента. Відзначимо, що навчальні плани з підготовки фахівців у галузі комп'ютерних наук передбачають виробничу практику після третього курсу навчання. Звичайно під час цієї практики студенти проходять ознайомлення з існуючим на підприємстві або в організації програмним забезпеченням і його супроводом. До розробки програмного забезпечення студентів, як правило, не залучають через різні причини: коротку тривалість практики; конфедиційність даних підприємства; задачі, що вимагають розв’язування, найчастіше досить складні для студентів і т.п.

Ми вважаємо, що в державні стандарти відповідних напрямів, крім виконання студентами індивідуальних курсових, кваліфікаційних і дипломних робіт, слід включити такі види навчальної діяльності, які допоможуть студентам, майбутнім програмістам, здійснити перехід від написання невеликих програм до створення програмного продукту, що передбачає колективну роботу, а саме, розробку колективного програмного продукту.

Слід зазначити, що університети США орієнтовані на дослідницьку роботу студентів. Метою є навчання всіх студентів на рівні, достатньому для наступної професійної або наукової кар’єри. Більшість курсів, що прослуховують студенти університетів у США і Канаді, зазвичай перебувають за межами їхньої основної спеціалізації.

В інших країнах такий узагальнюючий підхід застосовується досить рідко. Замість цього, очікується, що студенти будуть концентруватися на якійсь одній галузі, можливо доповненої декількома курсами із близьких дисциплін. Такий метод є типовим, наприклад, для Великобританії, де навчальні програми розраховані на три роки.

У Франції академічний рік складається із двох семестрів [102]. Заняття

проводяться традиційно, але перевага віддається самостійній роботі студентів. Фундаментальні дисципліни закінчуються екзаменами.

У Росії всі дисципліни ділять на цикли загальних гуманітарних і соціально-економічних дисциплін, загальних математичних і природничо-наукових дисциплін, загальних професійних дисциплін напряму, спеціальних дисциплін і факультативні дисципліни. У перших трьох циклах присутній федеральний компонент, тобто обов'язкові дисципліни, національно-регіональний компонент (або компонент ВНЗ) і курси за вибором студента, які визначаються ВНЗ. Курс з програмування відносять до федерального компоненту циклу загальних математичних і природничо-наукових дисциплін.

У державних стандартах бакалавра, що сьогодні діють в Україні, усі навчальні дисципліни ділять на:

– нормативні навчальні дисципліни, серед яких виділено цикл гуманітарних і соціально-економічних дисциплін, цикл природничо-наукової підготовки і цикл професійної ї практичної підготовки;

– вибіркові навчальні дисципліни, серед яких виділено цикл дисциплін самостійного вибору вищого навчального закладу і цикл дисциплін самостійного вибору студента.

Як зазначено раніше, курс з основ програмування напряму „Прикладна математика" у стандартах вищої освіти включено до циклу професійної підготовки. У той же час аналогічний курс „Основи програмування та алгоритмічні мови" напряму „Комп'ютерні науки" включено до дисциплін природничо-наукової підготовки розділу нормативних навчальних дисциплін.

Відзначимо, що для різних напрямів підготовки кількість годин (кредитів), відведених на курс з основ програмування, різна. Наприклад, для напряму „Прикладна математика" - 378 годин (7 кредитів), для напряму „Комп'ютерні науки" - 270 годин (5 кредитів), для напряму „Комп'ютерна інженерія" - 216 годин (4 кредити).

В Україні з дисциплін комп'ютерних наук передбачені такі форми проведення аудиторних занять як лекції, лабораторні роботи, практичні заняття.

Щодо часу вивчення і тривалості курсу з основ програмування відзначимо, що, згідно з державними стандартами вищої освіти різних комп'ютерних спеціальностей, ця дисципліна вивчається на першому курсі, або на першому та другому курсах навчання і є двосеместровою.

Виконання індивідуальних курсових робіт з програмування передбачено на 2-4-ому курсах навчання дисциплін циклу комп'ютерних наук.

У наший країні, згідно з Болонською декларацією, відбувається реформування вищої освіти, що передбачає зміни в організації навчального процесу, зокрема, посилення ролі самостійної роботи студентів і створення для цього відповідних умов [162] і, відповідно, зменшення аудиторного навантаження на студента до 18-24 годин на тиждень.

Освітня концепція Болонського процесу полягає у формуванні на перспективу загальноєвропейської системи вищої освіти, яка ґрунтується на спільності фундаментальних принципів функціонування, сформульованих у шістьох ключових позиціях [147]:

1. Введення двоциклового навчання;
2. Запровадження кредитної системи;
3. Контроль якості освіти;
4. Розширення мобільності;
5. Забезпечення працевлаштування випускників;
6. Забезпечення привабливості європейської системи освіти.

Щодо змісту курсу з основ програмування та стратегії навчання, то Комп'ютерне товариство Інституту Інженерів з Електротехніки і Електроніки і Асоціація з Обчислювальної Техніки спільно розробили документи „Computing Curricula 2001" [230] („CC 2001" надалі) та „Software Engineering 2004" [235], у яких наведено перелік рекомендацій щодо змісту університетських програм

навчальних дисциплін у галузі інформатики. В цих документах пропонується декілька стратегій складання навчального плану підготовки фахівців у галузі комп'ютерних наук. Зокрема, усі навчальні курси розбиваються на три рівні: вступні курси, основні курси і поглиблені курси.

Дисципліни, що пропонуються на першому або другому курсі навчання, визначені як вступні. Дисципліни, визначені як основні, є дисциплінами другого та третього років навчання і закладають фундамент для подальшого навчання за фахом. Дисципліни, визначені як поглиблені, концентруються на темах, що вимагають значної попередньої підготовки на ранніх курсах. Серед них є обов'язкові і факультативні (вибіркові).

Курс з основ програмування тут віднесено до вступних курсів.

Для вступного курсу з інформатики запропоновано шість стратегій навчання: підхід з орієнтацією на імперативне програмування, підхід з орієнтацією на об'єктно-орієнтоване програмування, підхід з орієнтацією на функціональне програмування, підхід із максимальним охопленням матеріалу, підхід з орієнтацією на алгоритми і підхід з орієнтацією на апаратну складову. Розглянемо більш детально ці стратегії.

Підхід з орієнтацією на *імперативне програмування* – це найтрадиційніший з усіх перелічених підходів, який передбачає вивчення однієї із процедурних мов програмування. Розробники документа відзначають, що при цьому підході можна використовувати і одну з об'єктно-орієнтованих мов для навчання основ програмування. Відмінність цього підходу від об'єктно-орієнтованої моделі, у такому випадку, 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 об'єктно-орієнтованого програмування (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 об'єктно-орієнтованого програмування.

Відзначимо, що для спеціальностей „Програмне забезпечення автоматизованих систем" і „Прикладна математика" вітчизняними навчальними планами передбачений окремий курс з об'єктно-орієнтованого програмування.

Підхід *з орієнтацією на об'єктно-орієнтоване програмування* також концентрується на програмуванні, але при цьому із самого початку робиться акцент на принципах об'єктно-орієнтованого проектування і програмування.

При об'єктно-орієнтованому підході до навчання програмуванню здійснюється раннє ознайомлення із принципами об'єктно-орієнтованого програмування, яке останніми роками стало надзвичайне важливим і для академічного середовища, і для промисловості. Але й об'єктно-орієнтована модель має свої недоліки. Зокрема, багато мов, які використовуються для навчання об'єктно-орієнтованого програмування (C++, Java, C#), набагато складніші за класичні процедурні мови. При виборі цієї моделі викладачу слід ретельно обміркувати методику навчання навчального матеріалу, інакше деталі обраної мови можуть затінити суть об'єктно-орієнтованого програмування.

Підхід *з орієнтацією на функціональне програмування* був вперше використаний у Масачусетському технологічному інституті в 80-х роках XX століття. Цей підхід характеризується використанням функціональних мов програмування (Lisp, Scheme). 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.

Підхід *із максимальним охопленням* матеріалу дає студентам більш цілісний погляд на дисципліну, що вивчається, уявлення про цілий ряд важливих тем, замість того щоб відразу занурювати їх у деталі однієї конкретної проблеми. Студенти, які навчаються за таким підходом, одержують більш широке уявлення про програмування, що надає їм можливість рухатися далі, упевнено і цілеспрямовано, опановуючи інші комп'ютерні курси.

До недоліків такого підходу можна віднести те, що розгляд широкого спектру теоретичних питань з програмування на ранній стадії навчання, який передбачає такий підхід, ускладнює засвоєння навчального матеріалу студентами молодших курсів.

Підхід з *орієнтацією на алгоритми* при вивченні основних концепцій програмування передбачає використання псевдокоду замість реальної мови програмування. 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. Це надає можливість студентам працювати із широким діапазоном типів даних і структур управляючої логіки. Після того як студенти опанують основні типи алгоритмів і типи даних, вони можуть починати використовувати одну з мов програмування. Завдяки виключенню із програми

навчання часу, відведеного на синтаксис і особливості певного середовища програмування, вступний курс, який побудований на основі цього підходу, може включати додаткові теоретичні теми. У 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 навиками налагоджування програм необхідне для подальшої успішної роботи студентів і тому, бажано, щоб вони практикувалися в цьому при навчанні якомога раніше.

Підхід *з орієнтацією на апаратну складову* передбачає навчання основ інформатики і, зокрема, основ програмування, починаючи з машинного рівня. 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, 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 може бути використаний при підготовці фахівців із проектування комп’ютерної техніки, коли необхідно досягнути раннього ознайомлення з апаратною частиною обчислювальної системи.

Незалежно від обраної стратегії навчання, розробники документів „CC 2001” та „Software Engineering 2004” пропонують набір вимог у вигляді списку розділів і тем, які повинен охопити курс з основ програмування (табл. 1.1).

Таблиця 1.1

Список розділів і тем, які повинен охопити курс з основ програмування, згідно з документом „CC 2001”

№	Назва розділу	Перелік тем розділу
1	Основні конструкції програмування	Основи синтаксису і семантики мов високого рівня; змінні, типи, вирази і присвоювання; основи введення-виведення; оператори розгалуження і циклу; функції і передавання параметрів; структурна декомпозиція
2	Алгоритми і розв’язування задач	Стратегії розв’язування задач; роль алгоритмів у процесі розв’язування задач; стратегії налагодження програм; концепція і властивості алгоритмів
3	Фундаментальні структури даних	Стандартні типи; масиви; записи; рядки і опрацювання рядків; подання даних у пам’яті комп’ютера; статичне, автоматичне і динамічне виділення пам’яті; управління пам’яттю під час виконання програми; посилання і покажчики; зв’язані списки; методи реалізації стеків, черг і хеш-таблиць; методи реалізації графів і дерев; стратегії вибору відповідної структури даних
4	Основи аналізу алгоритмів	Асимптотичний аналіз поведінки алгоритмів у середньому і крайніх випадках; нотація О-великого, о - малого, Ω (омега), θ (тетта); стандартні класи складності; емпіричні вимірювання ефективності; компроміс між часом і обсягом пам’яті в алгоритмах; використання рекурентних відношень для аналізу рекурсивних алгоритмів

Продовж. табл. 1.1

5	Фундаментальні обчислювальні алгоритми	Прості чисельні алгоритми; алгоритми послідовного і двійкового пошуку; алгоритми сортування з квадратичною складністю і з складністю O(n•log(n)); хешування; дерева двійкового пошуку; подання графів: обходи в ширину і глибину; алгоритми пошуку найкоротшого шляху; транзитивне замикання; мінімальне остовне дерево; топологічне сортування
	Алгоритмічні стратегії	Алгоритми повного перебору; „жадібні алгоритми”; алгоритми „розділяй і володарюй”; перебір з поверненням; метод гілок і меж; евристики; зіставлення із зразком і алгоритми опрацювання тексту; алгоритми чисельної апроксимації
7	Проектування ПЗ	Фундаментальні концепції і принципи проектування; об’єктно-орієнтований аналіз і проектування; проектування з метою повторного використання
8	Специфікації і вимоги до ПЗ	Важливість специфікації у процесі створення програм
9	Перевірка відповідності ПЗ	Основи тестування; генерація тестових прикладів

Також запропоновані варіанти розподілу розділів і тем для двосеместрового і трисеместрового курсу з інформатики. Вибір того або іншого варіанту залежить від навчальних планів спеціальностей і конкретних вищих навчальних закладів. В обох варіантах передбачаються наступні форми занять – курс лекцій, практичні заняття (лабораторні роботи) з програмування, самостійна робота, що цілком збігається з навчальними планами, передбаченими державними стандартами в нашій країні.

Російські програми з інформатики, в основному, відповідають документу „CC 2001” [83], [140].

Як показав аналіз вітчизняних освітніх стандартів вищої школи, навчальних програм з курсу основ програмування вищих навчальних закладів, в Україні немає чітко визначених стратегій навчання основ програмування, що є значним недоліком у системі підготовки фахівців у галузі комп'ютерних наук.

Відзначимо, що в Україні недостатньо вітчизняних підручників з дисциплін циклу комп'ютерних наук, а в тих, які є, в основному пропонується вивчення синтаксису і семантики мов програмування; ліцензоване програмне забезпечення майже недоступно через високу ціну.

1.3. Вимоги до професійних якостей майбутнього програміста

У роботі [140] поняття "професіоналізм" по відношенню до програміста визначається як інтегральна особистісна характеристика людини, яка:

– оволоділа нормами професійної діяльності і спілкування та здійснює їх на високому рівні, добиваючись професійної майстерності в галузі програмування;

– слідує професійній ціннісній орієнтації, дотримуючись професійної етики;

– розвиває себе засобами професії;

– прагне зробити творчий внесок у професію, збагативши її досвід;

– прагне і вміє викликати інтерес суспільства до результатів своєї професійної діяльності, сприяє підвищенню ваги і престижу своєї професії в суспільстві, гнучко враховує нові запити суспільства до неї.

Аналізом специфіки задач у галузі програмування і питанням професійних якостей програмістів у різний час займалися психологи і педагоги Ф. Брукс [28], Г. Вейнберг, Н. Вірт [35], Е. Дейкстра [70], С. Макконнелл [120], М.Л. Смульсон [190], Б. Шнейдерман [222] та ін. Дослідники стверджують, що професійні програмісти мають свої риси, якості та визначають психологічні і загальнолюдські риси, здібності і особливості мислення, які повинні бути характерними для програмістів (табл. 1.2 і табл. 1.3).

Таблиця 1.2

Якості, які властиві програмістові, що пов'язані безпосередньо із створенням програмного продукту

	Перелік якостей
За Е.Дейкстрою [69]**:**	– *здатність визначити архітектуру програми*, тобто розбивати складну задачу на елементарні складові і задати варіанти їх комбінування; – *уміння бачити задачу одночасно на різних рівнях деталізації*, тобто програміст повинен вільно переходити від опису задачі у загальних поняттях до суті нижчого рівня, що стоїть за цими поняттями; – *уміння уявляти собі процес, який проектується, у динаміці*, оскільки, дані, що оброблюються, у деякий момент часу можуть мати одні значення і взаємозв'язки, а у наступний момент деякі з них можуть змінитися; – *уміння бачити далі за одну програму, що розробляється в даний момент*, тобто вміти враховувати широке оточення даної частини задачі і можливість її включення до деякої загальної системи; – *уміння узагальнювати типові ситуації*, тобто необхідно уміти знаходити в програмі ідейно однорідні ділянки; – *уміння застосовувати і комбінувати добре відомі прийоми програмування і типові алгоритми*, тобто більшість нових ідей повинна знаходитися в тісній взаємодії з вже відомими ідеями і методами; – *уміння модифікувати програму*, оскільки постановка задачі в процесі розв'язування уточнюється і розроблена програма, яка вже написана, може не розв'язувати правильно поставлену задачу;

Продовження Таблиці 1.2

За Б. Шнейдерманом [222]:	– *здатність розуміти програми;* у програмуванні розуміння має нижній рівень (розуміння кожного рядка коду), середній рівень (розуміння структури алгоритму і даних) і вищий рівень (розуміння загального призначення програми). – *здатність налагоджувати програми*, тобто знаходити помилки в програмі. Така робота містить елементи розв'язування головоломок. Складність такої роботи обумовлена психологічними чинниками – посиленим занепокоєнням і небажанням допускати помилки; – *здатність модифікувати програми*: модифікація програм є досить поширеною задачею програмування. Успішність у вирішенні задачі модифікації можна вважати одним з критеріїв розуміння програм; – *уміння запам'ятовувати і відтворювати текст програм*: здатність відтворення правильної програми „по пам'яті" знаходиться у прямій залежності від досвіду програмування;
За М.Л. Смульсон [190]:	– *гнучкість і стратегічність мислення*; – *творчі властивості мислення*; – *уважність*, що виявляється у вмінні припускати якнайменше помилок; – *логічний характер мислення*.

Що стосується характеристик особистості програміста, Б. Шнейдерман відзначає, що „Особливості особистості відіграють критичну роль у визначенні як взаємодії між програмістами, так і робочого стилю окремих програмістів" [222] (табл. 1.3).

Таблиця 1.3

Психологічні і загальнолюдські риси, які повинні бути притаманні програмістові

	Перелік якостей
За Е. Дейкстрою [69]:	– *наявність комплексного мислення* - уміння наперед визначати етапи, які потрібно пройти, щоб вирішити те або інше питання; – *культура власної праці* - уміння забезпечити себе необхідним інструментарієм для роботи; – *здатність аналізувати власні помилки* (ця риса вказує на вимогливість програміста до себе з одного боку і застосування стилю програмування, що зменшує кількість помилок, з іншого боку); – *уміння працювати в колективі*, оскільки практично будь-яка серйозна розробка носить колективний характер, при цьому успіх всієї роботи залежить від взаєморозуміння, розподілу функцій і взаємостосунків у колективі; – *уміння працювати з користувачем*: професійний програміст повинен розуміти потреби користувача, уміти оцінити зручність конкретних форм інтерфейсу, мати змогу і здатність привчити користувача до нових засобів і систем; програміст повинен володіти психологічними знаннями, що допомагають йому у спілкуванні з користувачем; – *володіння інтелектуальними засобами*, які застосовуються для розуміння програми; – *дотримання правил загальнолюдської етики*; – *здатність чітко бачити дійсні труднощі і відкидати все, що не відноситься до справи;*

	– здатність виявляти всі випадки, де можна застосувати теорію, самостійно зважитися на її застосування або звернутися за порадою до більш досвідченого програміста; *– здатність при невдачі подолати самолюбство і знайти інший підхід*.
За Б.Шнейдерманом [222]:	*Наполегливість/пасивність*. Наполеглива людина володіє необхідною для виконання роботи ініціативою. *Інтровертність/екстровертність*. При колективному співробітництві й при взаємодії з користувачем сприятливим є дружній стиль відносин. *Внутрішня/зовнішня керованість*. Особистості з вираженою внутрішньою керованістю намагаються підкорити собі обставини. *Висока/низька збудливість*. Помірний рівень збудливості сприяє підвищенню продуктивності *Висока/низька мотивація*. Особистості з високим рівнем мотивації здатні виконувати дуже складні завдання. *Висока/низька терпимість до невизначеності*. Розробники повинні вміти працювати в умовах, коли: –чітко встановлене невелике число фактів або компонентів; –потрібно приймати рішення при обмежених вхідних даних; –для прийняття чергового рішення потрібно мати схильність до деякого ризику. *Уміння бути точним.* На завершальних етапах розв'язування завдання необхідні особлива увага до подробиць і готовність перевірити й урахувати кожну деталь.

	Скромність. Гарний програміст не повинен занадто самовпевнено ставитися до якості своїх програм. *Здатність переносити стрес*. У випадках відставання від планових строків потрібна здатність добре працювати у стресовій ситуації.
За М.Л.Смульсон [190]:	– висока працездатність і ретельність у праці; – оперативність мислення; – вміння приймати рішення в умовах обмеженого часу; – вміння створювати собі робоче місце, яке сприяє підвищенню продуктивності праці.

Також М.Л. Смульсон відзначає, що „…характерною рисою програміста є глибоке і гнучке володіння можливостями комп'ютера, інакше кажучи, засобами діяльності. Ефективність праці програміста багато в чому пов'язана з ефективністю системи засобів і вмінням обрати серед них найбільш адекватні тій задачі, що розв'язується" [190].

У роботі [26] зазначається, що сьогодні без самостійності, готовності швидко пристосовуватися до змін, переорієнтуватися, без уміння знайти підхід до розв'язування проблеми або терміново розв'язати нестандартне питання важко досягти професійного успіху. Ринок і конкуренція потребують фахівців-професіоналів із неординарним творчим мисленням.

Погоджуючись із наведеними вище вимогами до професійних якостей програміста, ми вважаємо, що до числа особливо важливих характеристик програміста варто віднести *здатність до навчання протягом усього періоду професійної діяльності*. На підставі власного досвіду також вважаємо, що деякі з перерахованих якостей можна і треба сформувати у майбутніх інженерів-програмістів вже в курсі з основ програмування.

1.4. Психолого-педагогічні основи навчання основ програмування

1.4.1. *Психологічні основи навчання основ програмування студентів*. Одним з основних завдань освітньої системи є завдання виховання творчої особистості. Для цього необхідно впроваджувати такі форми організації навчальної діяльності, які сприяють різнобічному розвитку кожного студента. З метою розв'язування цього важливого завдання психологи й дидакти досліджують загальні психологічні закономірності розумового процесу (А.М. Леонтьєв [115] та ін.); структуру мислення (Л.С. Виготський [38], Г.С. Костюк [103], С.Л. Рубінштейн [173] та ін.); системи розумових дій, що формують основу мислення (П.Я. Гальперін [40-41], Є.М. Кабанова-Меллєр [87], Н.Ф. Тализіна [196-197] та ін.).

У роботах з психології ([6-9], [38-40], [103-104], [115-116], [128], [161], [173]) розвиток студента як людини розглядається як процес, у якому поєднуються біологічні, фізіологічні, психічні і соціальні аспекти. Результатом є формування індивідуальності, суттю якої є єдність і взаємозв'язок властивостей людини як особистості й суб'єкта діяльності.

Дослідження А.М. Алексюка [3], Б.Г. Ананьєва [8], М.В. Буланової-Топоркової, А.В. Духавнєвої, С.І. Самигіна [152] та ін. присвячені вивченню особливостей розвитку особистості студента. Для студента, як людини, характерні психічні властивості індивіда, від яких залежить перебіг психічних процесів, виникнення психічних станів; суспільні відносини, зумовлені належністю до певної соціальної групи; якості, які зумовлені спадковістю і вродженими задатками: тип вищої нервової діяльності, безумовні рефлекси, інстинкти, фізична сила і т.п.; щодо морального розвитку у цьому віці, то він характеризується з одного боку, посиленням свідомих мотивів поведінки, закріпленням таких якостей як цілеспрямованість, рішучість, наполегливість, самостійність, ініціатива, вміння володіти собою; підвищенням інтересу до моральних проблем (мети і образу життя, обов'язку тощо), а з іншого боку,

недорозвиненість свідомої регуляції поведінки, наявність внутрішньої невпевненості у собі; щодо діяльності студента, то вона вимагає інтенсивності функціонування психіки, високої інтелектуальної напруженості, перевантажень [152].

Ці самі дослідники виділяють три основні підходи до пізнавальної діяльності студентів:

1) широкий підхід до цілей і задач навчання, тобто орієнтація на широку спеціалізацію, на різнобічну професійну підготовку, коли інтереси студентів спрямовані на більше широку область знань, ніж передбачено програмою;

2) чітка орієнтація на вузьку спеціалізацію, при якій пізнавальна діяльність студентів характеризується виходом за рамки навчальної програми в межах окремих дисциплін;

3) засвоєння знань і здобуття вмінь лише в межах навчальної програми.

Психологи розрізняють два основних види діяльності, пов'язані з пізнавальними процесами людини – учіння і навчання. Згідно з [205], „учіння - цілеспрямований процес засвоєння учнями знань, оволодіння уміннями й навичками", „навчання - цілеспрямований процес передачі і засвоєння знань, умінь, навичок і способів пізнавальної діяльності людини".

Польський вчений О.В. Долженко вважає [111], що „... навчання детерміноване метою, змістом і діями, за допомогою яких суб'єкт, що навчається, здобуває певні знання, уміння й навички; воно розвертається в результаті власної активності суб'єкта, що ніхто інший „за нього" не виявить, носить, головним чином, процесуальний характер, може протікати в різних формах, базується на пізнанні (прямому або непрямому) та індивідуальному досвіді, викликає зміни в поведінці особистості".

У роботі [223] зазначено, що „розрізняють дві основні спрямованості процесу навчання: навчання поняттям і формування вмінь (навичок)". При навчанні поняттям основна увага надається поданню і роз'ясненню нового

концептуального і фактичного навчального матеріалу: визначенню понять, опису їх властивостей і відношень, загальному і розгорненому опису процесів і т.п. При формуванні вмінь головна увага надається створенню проблемних ситуацій, що повторюються і поступово ускладнюються і вимагають від студента виконання певних дій.

Л.М. Фрідман [214] виділяє наступні етапи навчальної діяльності: 1) початково-мотиваційний, що містить такі основні навчальні дії: а) усвідомлення учнем основної навчально-проблемної ситуації, що вводить його в предмет майбутньої роботи з вивчення навчального матеріалу; б) формулювання основного навчального завдання; в) самоконтроль і самооцінка можливостей майбутньої діяльності по вивченню навчального матеріалу; 2) операціонально-пізнавальний, на якому учень вивчає навчальний матеріал, опановує знаннями, уміннями й навичками в процесі розв’язування основного навчально-пізнавального завдання; 3) контрольно-оцінюючий, на якому учень узагальнює вивчений навчальний матеріал, проводить корегування помилкових дій, аналізує роботу, що виконана, оцінює свою діяльність у цілому.

Умовою успішного навчання у ВНЗ є інтерес студентів до навчального предмету, ходу навчання і його результату. Цей інтерес пов'язаний з багатьма факторами: змістом навчального предмету, рівнем його складності; організацією процесу навчання; системою контролю й оцінювання знань; особистісними якостями самого викладача; системою цінностей студента, його найближчого оточення; взаєминами в студентському колективі. Тому одним з важливих завдань дидактики вищої школи є активізація навчально-пізнавальної діяльності студентів ([2], [85], [101], [180], [187], [224], [227] та ін.), яка “розуміється як цілеспрямована діяльність викладача, спрямована на розробку і використання такого змісту, форм, методів, прийомів і засобів навчання, які сприяють підвищенню пізнавального інтересу, активності, творчої самостійності студентів у засвоєнні знань, формуванні навичок і вмінь,

застосуванні їх на практиці” [187]. У курсі з основ програмування творчий підхід студентів до навчальної діяльності формується шляхом залучення їх до розв’язування різноманітних завдань, що поступово ускладнюються, на лабораторних заняттях, під час самостійної роботи, при підготовці рефератів і доповідей, при виконанні курсових робіт, розробці колективних проектів.

Точкою відліку початку підвищеного інтересу психологів щодо програмування є вихід першого видання книги Вейнберга „The Psychology of Computer Programming” більше 30 років тому. Предметом дослідження психології програмування є діяльність програміста, психологічні закономірності прийому і обробки інформації, функціонування психічних прийомів пам’яті і мислення у програміста, його працездатності, питання професійного відбору [160].

У дослідженнях J.G. Greeno [231], які засновані на інформаційному підході до психології навчання, запам’ятовування і розв’язування задач, запропоновано схему взаємодії компонентів пам’яті, які використовуються у роботі програміста і до яких відносять короткотривалу пам’ять, довготривалу пам’ять, робочу пам’ять. Згідно з цим дослідженням, постановка задачі (проблеми), яка повинна бути запрограмована, надходить до короткочасної пам’яті, у якій заздалегідь аналізуються відомості, що надійшли. Постійні знання програміста (знання синтаксису мови, методів побудови алгоритмів, опису структур даних і т.п.) знаходяться в довготривалій пам’яті. У робочій пам’яті відомості з короткочасної і довготривалої пам’яті об’єднується в нові структури. У процесі розв’язування задачі, тобто генерації програми, нові відомості, що є у короткочасній і довготривалій пам’яті, і поняття, що відносяться до проблеми, об’єднуються в робочій пам’яті, а результат використовується для генерації розв’язку або, у разі навчання, запам’ятовується в довготривалій пам’яті для подальшого використання. Як відмічено в [22]: „Хороші програмісти завжди трошки ледачі. Вони сидять і чекають осяяння, замість того, щоб втілювати в життя ту ідею, що першою прийшла їм у голову”.

Це відбувається завдяки тому, що досвідчений програміст зберігає в пам’яті складні багаторівневі знання про поняття і методи програмування і в процесі осмислення задачі визначає, які із цих знань слід застосувати. Частину цих знань називають семантичними знаннями [222]. Ці знання належать до загальних істотних для програмування понять, але не пов’язані з конкретною мовою програмування. До семантичних знань входять визначення понять нижнього рівня, наприклад: як зберігаються данні в пам’яті комп’ютера, як працює оператор розгалуження, як відбувається виклик підпрограми, які бувають типи даних. До них входять визначення понять середнього рівня, наприклад, обмін значень двох змінних, алгоритм пошуку більшого з двох значень, а також складніші алгоритми, такі як алгоритми сортування, бінарний пошук, обхід бінарного дерева пошуку т.п. Ці знання зберігаються у вигляді загальних змістовних фреймів, значною мірою не залежних від знань синтаксису конкретних мов програмування або засобів програмування, таких як пакети підпрограм або середовища програмування.

У довготривалій пам'яті зберігаються й синтаксичні знання. Ці знання більш конкретні й докладні, а тому легше забуваються у порівнянні з семантичними знаннями. До синтаксичних знань входять такі елементи як синтаксис оператора присвоєння, розгалуження, повторення, зарезервовані слова, імена бібліотечних функцій і стандартних модулів

На малюнку (рис. 1.1) за допомогою конкретного прикладу задачі, яку треба розв’язати (побудувати математичну модель, обрати метод розв’язування задачі, побудувати алгоритм, визначити типи даних, написати код програми, інтерпретувати отримані результати), зображено схему взаємодії компонентів пам’яті при розв’язуванні задач з програмування.

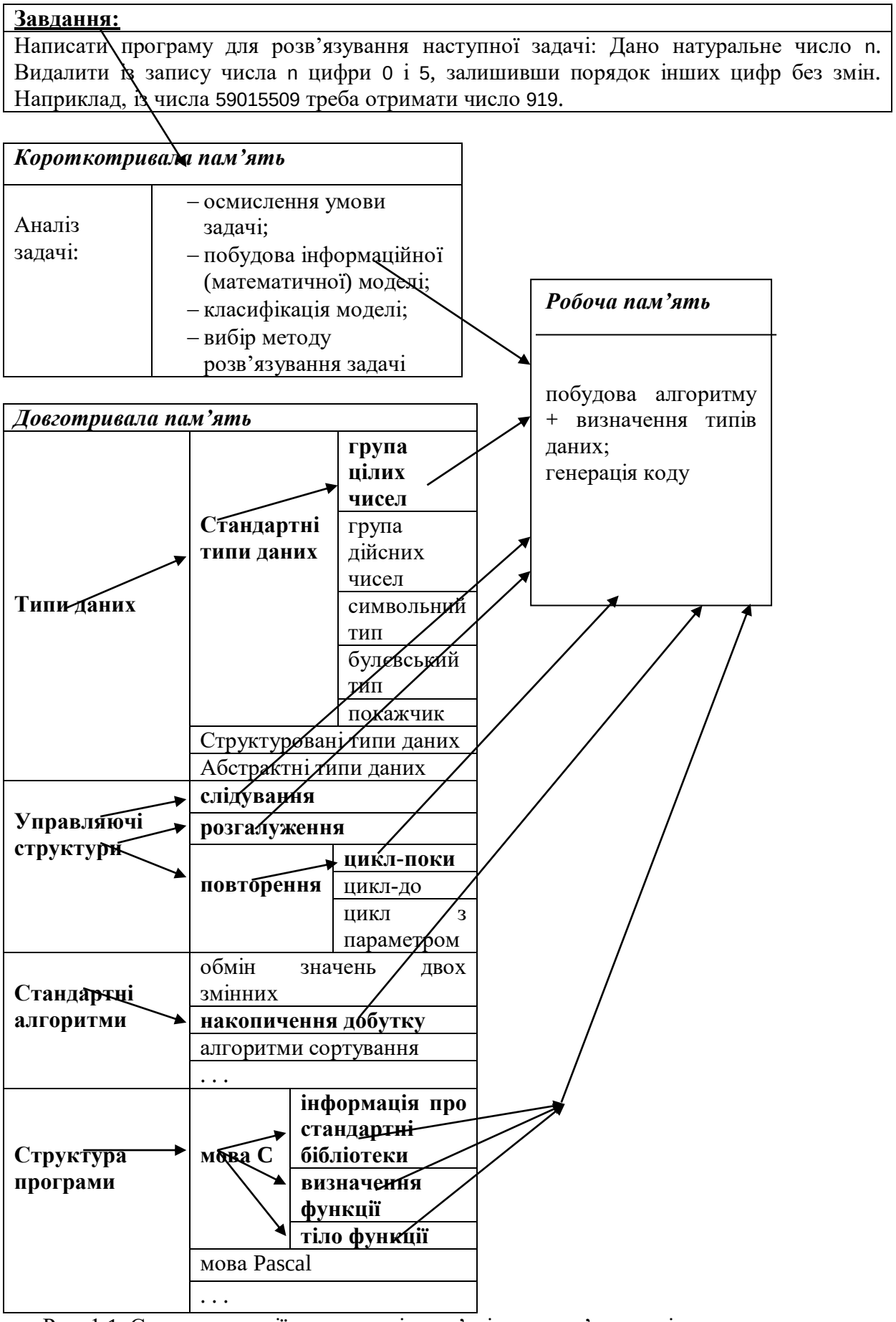


Рис. 1.1. Схема взаємодії компонентів пам'яті при розв'язуванні задач з програмування.

Програмістам, психологам і педагогам відомо, що звичайно важко вивчити першу мову програмування, а другу подібну вивчити відносно легко тому, що легше вивчити нове синтаксичне подання, якщо вже відома відповідна семантична структура.

Б. Шнейдерман указує на відмінності між семантичними і синтаксичними знаннями в довготривалій пам'яті програміста. Семантичні знання отримуються в процесі, який є інтелектуальною потребою осмисленого навчання, що включає розв'язування задач і вивчення навчальної літератури (підручників, довідників), що допомагає закріпити нові поняття сумісно з наявними семантичними знаннями. Синтаксичні знання, що носять інструктивний характер, заучуються механічно і не дуже перетинаються із вже наявними системами семантичних знань. Отриманню нових синтаксичних знань можуть перешкоджати одержані раніше синтаксичні знання, оскільки відбувається додавання нових відомостей, а не їх інтеграція. Дослідження психологів підтверджують, що при вивченні нових мов програмування програмісти дуже легко розуміють семантику основних конструкцій мови, але часто помиляються при кодуванні. Наш досвід роботи показує, що, наприклад, студенти, що вже мають досвід написання програм мовою Pascal, при програмуванні мовою C не ставлять крапку з комою перед службовим словом „`else`" в операторі розгалуження й не звертають увагу на розходження ідентифікаторів, у які входять однакові символи (букви алфавіту) в одній і тій же послідовності, але в різних регістрах (рядкові й прописні).

Дослідники припускають, що процес написання програми розв'язування поставленої задачі складається із двох фаз. На першій фазі відомості про умову задачі надходять до системи мислення програміста і через короткотривалу пам'ять потрапляють до робочої пам'яті, де вони аналізуються. Аналогічно з довготривалої пам'яті програміста пересилаються узагальнені відомості (семантичні і синтаксичні) для подальшого аналізу задачі. На другій фазі, при побудові загального плану розв'язування задачі, відбувається її покрокове

уточнення. Спочатку розв’язок задачі уявляється в узагальнених термінах. Потім це уявлення розвивається до більш докладних планів аж до генерації конкретних кодів, що відображають самі дрібні деталі.

Однією з характеристик професійних якостей програміста є розуміння програм. Розуміння програм необхідне при налагодженні і при модифікації програм, а також при навчанні. Б. Шнейдерман у своїх дослідженнях припускає, що для розуміння програми програміст будує багаторівневу внутрішню семантичну структуру, використовуючи свої знання синтаксису мови програмування. На найвищому рівні програміст повинен розуміти, яка задача розв’язується за допомогою програми. На більш низьких семантичних рівнях програміст може розуміти знайомі послідовності операторів. Можлива ситуація, коли програміст може зрозуміти деталі нижнього рівня, не розуміючи загальної задачі. „Головне полягає в тому, що програмісти виробляють внутрішню семантичну структуру для подання синтаксису програми, але не запам’ятовують і не розуміють програму як послідовність рядків” [222]. Процес запам’ятовування, за допомогою якого програміст перетворює програму у внутрішню семантичну структуру, має „кусковий” характер [233]. Замість того, щоб розбирати програму символ за символом, програміст розпізнає призначення груп операторів, потім збирає ці фрагменти разом для формування більш великих фрагментів доти, поки не буде зрозуміла вся програма. Тому, що внутрішня семантична структура програми розробляється самим програмістом, ці знання не швидко забуваються і доступні для різних цілей: використання інших структур даних для розв’язування задачі, написання програми розв’язування задачі іншою мовою програмування, роз’яснення призначення програми кому-небудь.

У процесі навчання майбутніх інженерів-програмістів також є синтаксичні і семантичні аспекти. Так, у курсах навчання другої або третьої (не першої) мови програмування однієї парадигми основна увага приділяється

синтаксичним аспектам семантичних понять. Курси з розробки й аналізу алгоритмів зосереджені на семантичних знаннях.

Згідно [222], семантичні і синтаксичні знання розділені. Багаторівневі семантичні знання, отримані в основному завдяки цілеспрямованому навчанню, знаходять своє втілення в багаторівневому методі розробки внутрішньої семантики для деякої конкретної задачі. Відносні синтаксичні знання, придбані, в основному, завдяки механічному заучуванню, цілком обмежені мовою. Семантичні знання необхідні для аналізу задачі, у той час як синтаксичні знання використовуються на етапах кодування і виконання.

Багато вчених (Ю.І. Машбиць [124], Н.Ф. Тализіна [196-197] та ін.) підкреслюють неприпустимість зведення навчальних цілей тільки до засвоєння предметних знань і вмінь. На їхній погляд у систему цілей необхідно включати вміння вчитися. Ми цілком підтримуємо цю точку зору й у цьому контексті вважаємо, що у процесі навчання основ програмування треба визначити систему завдань, за допомогою якої можна буде сформувати у студентів знання, уміння й навички, необхідні й для їх подальшого навчання й у наступній професійній діяльності. Система завдань з програмування як засіб навчання є одним з найважливищих компонентів методичного забезпечення в навчанні основ програмування (див. п. 2.3.2.2).

Також необхідно вирішити питання про складність змісту навчального матеріалу та його обсяг. Зміст навчального матеріалу, з одного боку, повинен бути максимально наближеним до сучасного рівня розвитку комп'ютерних наук, з іншого боку, повинен бути доступним студентам. Також треба систематизувати й визначити послідовність його викладання. Важливо визначити міру складності навчального матеріалу, тому що занадто спрощений або занадто складний зміст ведуть до зниження інтересу й пізнавальної активності студентів у процесі навчання. Обсяг навчального матеріалу не повинен бути надмірним, щоб не перевантажувати студентів у процесі навчання (див. п. 2.2).

Так само важливим є визначення необхідної кількості завдань для оволодіння навчальним матеріалом і формування практичних умінь і навичок. Ю.К. Бабанський відзначає, що „спеціальні дослідження показали, що якщо дотримуватися принципів систематичності і послідовності, то для формування визначених вмінь і навичок досить виконати п’ять вправ при середньому рівні складності навчального матеріалу. Для визначення кількості вправ у навчально-методичних комплексах необхідна побудова цих вправ таким чином, щоб на мінімальному числі вправ учні опановували б загальним принципом виконання подібного роду дій” [15]. Зведення кількості завдань до мінімуму приводить до того, що їх виявляється недостатньо для забезпечення цілей навчання. З іншого боку, занадто велика кількість завдань ставить перед студентом нездоланний бар’єр.

Зазначимо, що для забезпечення якісного засвоєння студентами знань з основ програмування, варто забезпечувати не тільки нарощування числа досліджуваних понять, але і вивчення тих самих понять на різних рівнях деталізації, наукової строгості. Ми вважаємо, що дидактичні принципи систематичності, послідовності навчання при побудові курсу з основ програмування доцільно реалізовувати у вигляді дидактичної спіралі – багаторазове повернення на новому рівні [230]. Мається на увазі оволодіння студентами знаннями й уміннями в контексті, що ускладнюється, за допомогою збагачення, розвитку й узагальнення досліджуваних понять. Наприклад, для поняття „алгоритм” виділяють наступні рівні складності розгляду – лінійний, розгалужений, циклічний, комбінований, з використанням підпрограм, рекурсивний. На прикладі типів даних – стандартні, структуровані, абстрактні типи даних. Дидактична спіраль при цьому утворюється в результаті комбінації рівнів цих параметрів, з почерговим їхнім ускладненням.

У даний час психологи і педагоги приділяють значну увагу проблемі адаптації студентів до умов навчання у вищій школі. Ця проблема досліджується на психофізіологічному [20], індивідуально-

психологічному [185], соціально-психологічному рівнях [9]. Специфіка процесу адаптації у ВНЗ визначається відмінністю в організації навчального процесу в середній і вищій школах. Звідси виникають відмінності в результатах діяльності при навчанні однієї і тієї ж людини у школі і у ВНЗ. Першокурснику необхідно немало часу для того, щоб пристосуватися до вимог навчання у ВНЗ. Як стверджується в [172], „однією з причин низьких темпів адаптації студентів є неузгодженість у педагогічній взаємодії між викладачем і студентом при організації способів навчання". Взаємодія викладача і студента в процесі навчання довгий час розглядалася як система, у якої управляючі функції належать викладачу, що затримувало формування активної позиції студента в навчальному процесі [109]. Від того, як довго за часом і за різними витратами відбувається процес адаптації, залежать поточні і майбутні успіхи студентів у навчанні, процес їхнього професійного становлення ([17], [25], [73], [91]).

З метою подолання проблеми адаптації до умов навчання у вищій школі і задоволення вимогам до рівня знань шкільного курсу інформатики, ми вважаємо за потрібне включити *пропедевтичний курс* у процес навчання основ програмування. Цей курс повинен передувати курсу з основ програмування. Змістом такого курсу повинні бути основні розділи шкільного курсу інформатики.

Результати проведеного нами педагогічного експерименту підтверджують правильність наших припущень: наявність пропедевтичного курсу допомагає студентам першого курсу подолати бар'єр, який виникає у декого з них при переході від шкільної системи навчання до системи навчання у ВНЗ, і надає можливість забезпечити однаковий стартовий рівень знань, умінь і навичок з алгоритмізації і основ програмування у студентів для вивчення розділів курсу з основ програмування.

1.4.2. Діяльнісний підхід у навчанні основ програмування.

Аналіз психолого-педагогічної літератури (П.Я. Гальперін [40], Д.Б. Ельконін, Г.С. Костюк [103-104], О.М. Леонтьєв [116], Н.Ф. Тализіна [197] та ін.) показує, що діяльністна теорія розглядає процес навчання як формування пізнавальної діяльності учнів. Відповідно до діяльністного підходу засвоєння знань людиною здійснюється в процесі власної активності. Процес діяльності одночасно є процесом формування людських здатностей і функцій, одиницею діяльності є предметна дія [116].

Розглядаючи навчання студента у ВНЗ як підготовку до майбутньої професійної діяльності, розрізняють змістове наповнення складових структури навчальної та професійної діяльності. Вважається, що метою навчання є загальний і професійний розвиток особистості, а метою праці - виробництво матеріальних і духовних цінностей; пізнавальні, переважно інтелектуальні, дії в навчанні й практичні (у тому числі теоретично-практичні) у професійній діяльності; засобами навчання є засоби психічного відбиття дійсності, а засобами праці - перетворення реальної дійсності; результатами діяльності навчання виступають діяльні здатності особистості, а результатами професійної діяльності - товари, нові знання, освіченість людей і т.п.

Структурною складовою професійної діяльності інженера-програміста є процес розв’язування задачі. Тому метою курсу з основ програмування є формування деяких ланок цілісної структури майбутньої професійної діяльності студента, а саме, формування у студентів вміння складати ефективні алгоритми для розв'язування різноманітних задач та писати програми згідно складеного алгоритму.

П.Я. Гальперін розрізняє дві основні частини процесу розв’язування задач: аналітичну, котра пов'язана з дослідженням задачі і здогадів щодо шляхів її розв’язування і є необхідною передумовою творчого мислення, і конструктивну, котра пов'язана з побудовою того, „що потрібно знайти або

придумати щоб одержати розв’язок задачі” [41] і яка являє собою творчий компонент мислення.

Застосування діяльнісного підхіду в навчанні висуває певні вимоги до методів навчання і послідовності їх використання [197].

Методологічні основи методів навчання в різний час розглядали в своїх роботах А.М. Алексюк [3-4], Ю.К. Бабанський [15-18], В.П. Беспалько [23], М.І. Махмутов [123], З.І. Слєпкань [186-187], Н.Ф. Тализіна [196-197] та ін.

Більшість дидактів, у загальному випадку, розглядають методи навчання як способи взаємопов'язаної діяльності вчителя і учнів, спрямовані на розв’язування комплексу задач навчального процесу. Під методом навчання у вищій школі розуміють способи роботи викладача і студентів, за допомогою яких досягається оволодіння знаннями, навичками і уміннями, формується світогляд студентів, розвиваються їхні здібності [187]. Кажучи про функції методів навчання, виділяють спонукаючу (стимулюючу), організуючу і контролюючу функції, а також освітню, виховну і розвиваючу функції [187].

Критерієм ефективності методів навчання, згідно з Ю.К. Бабанським [16], вважають відповідність досягнутих результатів в освіті, вихованні і розвитку учнів тим задачам, які необхідно було розв’язати на відповідному етапі навчання з урахуванням реальних навчальних можливостей учнів.

Існують різні класифікації методів навчання: за принципом наростання ступеня дослідницького підходу до навчання, за джерелами передачі і здобування знань, за способами організації навчальної діяльності учнів, за характером навчально-пізнавальної діяльності учнів. М.І. Махмутов визначає методи навчання залежно від дидактичної мети, змісту навчання і характеру взаємодії вчителя і учнів [123]. Класифікація методів навчання, яка запропонована академіком Ю.К. Бабанським [17], заснована на ідеї цілісного підходу з розділенням усієї сукупності методів навчання на три групи методів: 1) методи організації й здійснення навчально-пізнавальної діяльності; 2) методи

стимулювання і мотивації навчально-пізнавальної діяльності; 3) методи контролю і самоконтролю ефективності навчально-пізнавальної діяльності.

У даний час у системі вищої освіти однією з актуальних залишається проблема пошуку ефективних технологій навчання, що забезпечать не тільки засвоєння студентами знань і формування професійних умінь і навичок, але і розвиток творчих і комунікативних здібностей особистості, формування і розвиток самодостатньої творчої особистості. Залучення студентів до активної навчально-пізнавальної діяльності в ході навчального процесу пов'язане із застосуванням прийомів і методів, що одержали узагальнену назву методи активного навчання.

Наукове обґрунтування методів активного навчання запропоноване Є.С. Полат [138], С.Л. Рубінштейном [173], А.М. Смолкіним [188] та ін.

Методи активного навчання вимагають використання такої системи методів, що спрямована, головним чином, не на подання викладачем готових знань і їхнє відтворення, а на самостійне оволодіння студентами знань у процесі активної пізнавальної діяльності; внесення елементів професійної діяльності в навчальний процес. Саме в активній діяльності, що направляється викладачем, студенти опановують необхідними знаннями, уміннями, навичками для їхньої майбутньої професійної діяльності, розвиваються їхні творчі здібності. Методи активного навчання спрямовані на залучення студентів до самостійної пізнавальної діяльності, сприяють розвитку особового інтересу студентів до розв’язування пізнавальних задач, забезпечують можливість застосування студентами отриманих у процесі навчання знань і вмінь.

Розрізняють імітаційні методи активного навчання, тобто форми проведення занять, у яких навчально-пізнавальна діяльність побудована на імітації професійної діяльності, і не імітаційні, тобто способи активізації пізнавальної діяльності студентів на лекційних заняттях [188].

У курсі з основ програмування доцільно використовувати такий метод активного навчання як *індивідуальна або колективна розробка студентами невеликого програмного продукту* (див. п. 2.3.2.4).

Застосування методів активного навчання створює умови, при яких студенти одержують ефективні мотиваційні стимули, що сприяють розвиткові їхніх здібностей, елементів творчої, пізнавальної активності, самостійності, тобто проявові позиції суб'єкта навчально-виховного процесу. Викладач, при цьому, є організатором і координатором діяльності студента, допомагає йому в оволодінні способами і прийомами навчальної діяльності.

Розвитку взаємодії і співробітництва студентів у процесі навчальної діяльності сприяють інтерактивні методи навчання. Суть інтерактивних методів навчання полягає в тому, що навчальний процес організований таким чином, що практично всі студенти залучаються в процес пізнання. У кожного є можливість розуміти і рефлектувати із приводу того, що вони знають і думають. При цьому відбувається обмін знаннями, ідеями, способами діяльності.

Відмінною рисою інтерактивних методів навчання є ініціативність студентів у навчальному процесі, що стимулює педагог із позиції партнера помічника. Інтерактивне навчання характеризується також груповою взаємодією всіх учасників навчального процесу, емоційною включеністю і активністю кожного в процесі роботи. Хід і результат навчання набуває особистої значимості для всіх учасників процесу і надає можливість розвинути у студентів здатність самостійного вирішення проблеми.

У процесі навчання основ програмування інтерактивні методи навчання доцільно застосовувати при проведенні лабораторних робіт (див. п. 2.3.2.).

1.4.3. *Компетентністний підхід у навчанні основ програмування*. Сьогодні конкурентноздатність фахівця на сучасному ринку праці багато в чому залежить від його здатності опановувати нові технології, адаптуватися до інших умов праці. Тому провідна ідея системи освіти виражена в спробі

зв’язати результуючу складову освіти з її планованими результатами. Впровадження компетентністного підходу в процес навчання припускає можливість реалізації цієї ідеї.

У сучасному уявленні компетентністного підходу маються на увазі ідеї загального й особистісного розвитку, сформульовані у контексті психолого-педагогічних концепцій розвиваючої і особістістно-орієнтованої освіти, де зміст освіти визначається четирьохкомпонентною моделлю: знання, уміння, досвід творчої діяльності і досвід ціннісного відношення [164].

Усередині компетентністного підходу виділяють два базових поняття: компетенція і компетентність, при цьому перше з них „включає сукупність взаємозалежних якостей особистості, що задаються стосовно визначеного кола предметів і процесів”, а друге співвідноситься з „володінням людиною відповідною компетенцією, що включає його особистісне відношення до неї і предмету діяльності” [216]. У цьому ж контексті функціонує і поняття „освітньої компетенції”, яка розуміється як „сукупність значеннєвих орієнтацій, знань, умінь, навичок і досвіду діяльності учня стосовно визначеного кола об'єктів реальної дійсності, необхідних для здійснення особистісної і соціально-значимої продуктивної діяльності”. Освітні компетенції диференціюються за рівнями, як і зміст освіти [164]:

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

- загально предметні (реалізовані на змісті, що є інтегративним для сукупності предметів, освітньої галузі);

- предметні (формовані в рамках окремих предметів).

Існують різноманітні підходи до формулювання ключових компетенцій і їхніх класифікацій. Наприклад, російські класифікації включають загальнокультурну, навчально-пізнавальну, інформаційну, комунікативну, соціально-трудову компетенції і компетенцію особистісного самовдосконалення. У системі сертифікації компетенцій, що розроблена OCR

(екзаменаційною комісією Оксфорда Кембриджу), виділяються ключові компетенції декількох рівнів, що, на думку розробників, необхідно враховувати при підготовці компетентного висококваліфікованого фахівця незалежно від професійної спрямованості: 1 - комунікація; 2 - операції з числами; 3 - інформаційні технології; 4 - робота з людьми; 5 - удосконалення здібностей до навчання і підвищення результативності; 6 - вирішення проблем; 7 - розвиток особистісних компетенцій [93].

Відповідно, під компетентністним підходом у навчанні основ програмування ми розуміємо підхід, що акцентує увагу на результатах навчання, причому як результат будемо розглядати не суму засвоєних знань, а здатність студента використовувати отримані знання в процесі навчання й у наступній професійній діяльності. У цілому, „успішність людини у майбутньому професійному й соціальному житті визначається рівнем розвитку ключових компетентностей” [131].

Компетентності з основ програмування – це наявність у студента здатностей:

загальних:

– використовувати в практичній діяльності засвоєні знання, уміння та навички з основ програмування у процесі вивчення дисциплін з комп'ютерних наук, у навчальному процесі в цілому та у наступній професійній діяльності

– організовувати й планувати свою, зокрема навчальну, діяльність;

– знаходити й аналізувати відомості з різних джерел;

– формалізувати здобути знання, накопичувати їх;

– працювати в колективі;

– використовувати знання іноземних мов у професійної діяльності.

спеціальних:

– складати алгоритми поставлених задач и писати програми;

– застосовувати й комбінувати типові алгоритми й відомі методи програмування;

– налагоджувати й тестувати навчальні програми в середовищі розробки мови програмування;

– розуміти галузі застосування здобутих знань з основ програмування в інших професійно-орієнтованих і спеціальних дисциплінах та в сфері майбутньої професійної діяльності

– вивчати нові мови програмування;

Звернемося до проблеми вибору парадигми програмування і, відповідно, вибору мови програмування в процесі навчання основ програмування.

Існують різні парадигми програмування, що представляють собою незалежні напрямки: функціональна, імперативна, об'єктно-орієнтована, декларативна та ін. (див. п.1.2).

Зараз широко використовуються комп'ютерні системи, що надають можливість використання *паралельного програмування*. У мультипроцесорній системі є декілька центральних процесорів, які спільно використовують загальну пам'ять. У розподіленій комп'ютерній системі є декілька комп'ютерів, у кожного з яких є своя пам'ять і свій центральний процесор, причому всі ці комп’ютери за допомогою лінії зв'язку з'єднані в одну мережу, через яку вони можуть взаємодіяти один з одним. При виконанні програми на одному комп'ютері в паралельному програмуванні програма ділиться на кілька окремих завдань, що одночасно виконуються на віртуальному, умовно – багатопроцесорному, комп'ютері, хоча, звісно, реально у даному випадку в кожний момент часу може виконуватися тільки одна задача.

Для керування інформаційними потоками – *мережеве програмування* – використовуються нові технології, що забезпечують способи навігації в глобальній мережі Internet.

Як показує аналіз навчальних планів і робочих програм, освітніх стандартів вищої школи напрямів з комп'ютерних наук у більшості вищих навчальних закладах України в курсі з основ програмування вивчають структурне програмування, яке є розвитком імперативного програмування.

Структурні принципи побудови програм були сформульовані у фундаментальних роботах В.М. Глушкова з теорії дискретних перетворень, у яких він визначив основні структури і правила їхніх композицій і довів фундаментальну теорему про можливість перетворення програми в структуровану форму без зміни інформаційної бази, з якою вона працює [45]. У роботі Е. Дейкстри [69] визначено, що структурна організація програм задовольняє вимогам доведення правильності програм і допускає внесення змін до програми без ускладнень.

Ми вважаємо, що *навчання основ програмування повинне здійснюватися саме за парадигмою структурного програмування*.

При виборі парадигми програмування було враховано два чинника:

1) використання структурної організації програм надає можливість формувати і розвивати алгоритмічне мислення ([73], [97], [99]). Алгоритмічне мислення, яке розглядається як визначення послідовності дій, і яке, поряд з образним і логічним мисленням, визначає інтелектуальний рівень людини, її творчий потенціал. Саме навички планування, звичка до точного і повного опису своїх дій допомагають професійним програмістам розробляти алгоритми розв'язування різноманітних задач;

2) наступність – традиційно в Україні в шкільному курсі інформатики в розділі „Алгоритмізація і програмування" вивчають саме структурне програмування.

Що стосується вибору мови програмування для навчання основ програмування, ми вважаємо, що майбутніх інженерів-програмістів навчати розв'язуванню задач необхідно мовою програмування С (див. п. 2.2.3).

Але, за для забезпечення якості навчання основ програмування, треба спочатку навчити майбутніх інженерів-програмістів складати програми мовою програмування Pascal, який був розроблений Н. Віртом саме „для навчання новачків тому, як писати програми". Примітно, що „ ... структурне програмування задавало низхідний принцип розробки (покрокова декомпозиція), передбачало структурування логіки й даних, за рахунок

простоти й математичної основи підвищувало надійність програмного забезпечення. Все це органічно вписувалося в можливості лаконічного Паскаля" [232].

Невід'ємним засобом навчання основ програмування є *середовище мови програмування,* що призначено для розробки програм. Середовище складається з набору засобів підтримки і команд, що активізують ці засоби. Звичайно засобами підтримки є редактори, компілятори або інтерпретатори, налагоджувачи**,** верифікатори, генератори тестових даних і засоби виведення результатів [159]. Кожен із засобів підтримки також є програмою і може бути використано і використовується програмістом на різних стадіях складання програми.

Для навчання основам програмування у ВНЗ, залежно від цілей навчання, традицій, можливостей і переваг, використовуються такі системи програмування як Turbo Pascal, Borland Pascal, FreePascal, Borland C, Borland C++, Delphi, Borland C++ Builder, MS Visual Studio, Visual BASIC та ін.

Більшість з цих систем програмування орієнтовані на професійних програмістів, не призначені для навчальних цілей і ускладнюють процес навчання тому, що перевантажені надаваними можливостями, у яких немає необхідності при навчанні основ програмування На наш погляд, ці програмні продукти не зовсім придатні для використання у навчальному процесі тому що:

1) перевантажені можливостями, що надаються (наприклад, команда Ё у середовищі BC, деякі пункти меню команди Options т.і.);
2) мають англомовний інтерфейс;
3) містять часто англомовну довідкову систему.

Крім того, більшість з цих середовищ не є вільнорозповсюдженими.

Як показує наш досвід, у багатьох студентів молодших курсів існує психологічний бар'єр при роботі на комп'ютері. До того ж, існуюче середовище програмування, яке переобтяжене можливостями, що надаються, посилює цю проблему: увага студентів розсівається на сприйняття, осмислення інформації

на екрані, їх відволікають і лякають значна кількість команд, до того ж англомовних, які найчастіше не використовуються у навчальному процесі.

Сьогодні існують середовища програмування, які є як повноцінними промисловими системами розробки програмного забезпечення, так і вирішують завдання навчально-методичного призначення. До таких програмних продуктів належить компонентне середовище програмування Blackbox Component Builder (Blackbox) [229]. Цей програмний продукт розроблений фірмою Oberon Microsystems і, з метою некомерційного використання, поширюється вільно. Мовою програмування в цьому середовищі є компонентний Pascal. BlackBox Component Builder розповсюджується за Open Source ліцензією.

В Україні також приділяють увагу розробці середовищ програмування, які призначені для використання у навчальному процесі. До таких програмних продуктів належить система АЛГОРИТМ, що розроблена під керівництвом О.В. Міленького [232]. Цей програмний продукт надає можливість користувачам оперувати безпосередньо із блоками алгоритму, їхніми взаємними зв'язками й перевіряти правильність функціонування складеної блок-схеми алгоритму. Вона створена у вигляді інтегрованого інтерактивного середовища, що забезпечує легкість освоєння, зручність діалогу й багатоступінчастий контроль помилок. Для роботи із системою не потрібне знання мови програмування. Вона базується на основних алгоритмічних поняттях, спільних для більшості мов, і тому є досить універсальною й придатною для розв’язування широкого кола обчислювальних задач. Разом з тим, у структурному й синтаксичному відношеннях система АЛГОРИТМ ближче до мови програмування Pascal, ніж до інших мов. Програма надає можливість будувати лінійні алгоритми, алгоритми з розгалуженням й циклічні алгоритми; об'єднувати блоки алгоритму в процедури; використовувати процедури як без параметрів, так і з параметрами (параметрами-значеннями й параметрами-змінними), а також задавати всередині процедури локальні змінні й масиви, що надає можливість опанувати принципи структурування алгоритму

й ієрархічного доступу до його об'єктів; використовувати не тільки стандартні математичні функції, але й визначати власні функції користувача.

Програмно-методичний комплекс „Відеоінтерпретатор алгоритмів пошуку й сортування”, що розроблено на кафедрі інформаційних технологій Херсонського державного університету під керівництвом О.В. Співаковського, призначений для використання при вивченні розділу „Основи алгоритмізації й програмування” курсу інформатики загальноосвітньої школи як засіб вивчення алгоритмів, мови програмування, налагодження програм [66]. Клас завдань Відеоінтерпретатора – різні алгоритми обробки масивів даних, у тому числі сортування, пошук унікальних елементів (максимуми, мінімуми й ін.). Робочою мовою програмування в навчанні основ програмування обрана мова програмування Pascal. Цей програмно-методичний комплекс має гриф Міністерства освіти й науки України.

Навчальні програми „Паскаль - крок за кроком” і „Си - крок за кроком”, які розроблені російськими педагогами [139], призначені для навчання основ програмування школярів старших класів і студентів.

У навчальній програмі „Паскаль - крок за кроком” викладаються основи мови Pascal на прикладі реалізації Turbo Pascal фірми Borland. Курс складається із семи уроків: 1) структура Pascal-програми: призначення мов програмування; структура Pascal-програми; константи й змінні; алфавіт і службові слова; найпростіші типи даних; 2) операції, вирази і оператори: оператор присвоювання й арифметичні вирази; порядок виконання арифметичних операцій; процедури виведення `Write` і `WriteLn`; стандартні функції; поняття алгоритму; повтори й порівняння; цикл `while do`; 3) управляючі конструкції мови Pascal: оператори циклів; операції відношень; тип `boolean` і логічні вирази; оператор розгалуження `if-then-else`; 4) типи даних: інтервальні й перелічувальні типи; визначення власного типу користувача; перетворення типів; стандартні процедури введення `Read` і `ReadLn`; робота з бітами; оператор вибору `case of`; 5) масиви і рядки: регулярні типи (масиви); операції із

символьними масивами; помилки в роботі з регулярними типами; рядковий тип і операції над ним; введення/виведення даних; робота з текстовими файлами; 6) процедури і функції: процедури й функції користувача; передача параметрів у процедуру або функцію; область видимості імен; рекурсія й випереджальний опис; використання модулів; 7) покажчики і записи: записи; робота з динамічними змінними; множини; основні відмінності авторської версії мови програмування від мови Turbo Pascal фірми Borland.

У навчальній програмі „Си - крок за кроком" подається ANSi-стандарт мови С. Курс також складається із семи уроків: 1) структура програми на С: призначення мов програмування; структура найпростішої програми на С; константи і змінні; типи даних; 2) операції, вирази і оператори: операція присвоювання і арифметичні вирази; порядок виконання арифметичних операцій; функція виведення `printf()`; поняття алгоритму; повторення і порівняння; цикл `while`; 3) управляючі конструкції мови С: оператори циклів у С; операції відношень; умовні й логічні вирази; оператор розгалуження `if-else`; знайомство із препроцесором мови С; 4) додаткові операції: операція умови `?:` ; додаткові операції присвоювання; операції збільшення й зменшення; операції приведення типів; операції для роботи з окремими бітами; 5) масиви і рядки: реалізація перемикачів у С (`switch`) ; призначення і використання операторів `break`, `continue` і `goto`; масиви й рядки; деякі стандартні функції введення/виведення; 6) функції і структура програми: функції користувача; передача аргументів; типи функцій; використання покажчиків; класи пам'яті і області видимості імен; можливості С-препроцессора; 7) масиви, покажчики, структури: покажчики й масиви; рядки й багатовимірні масиви; структури й об'єднання; стандартні бібліотеки для роботи з рядками.

Обидві програми містять велику кількість прикладів, контрольних питань, що дає можливість організувати самостійну роботу студентів з вивчення мов програмування Pascal і С.

У цілому, існуючих програмно-методичних комплексів, розроблених для засвоєння курсу з основ програмування, на наш погляд, недостатньо.

У Черкаському національному університеті розроблено програмно-педагогічне забезпечення на підтримку окремих тем курсу з основ програмування (див. п. 2.4.2).

1.4.4. Диференційований підхід у навчанні основ програмування.

Важливою умовою успішного навчання у ВНЗ є інтерес студентів до предмету, що вивчається, ходу навчання та його результатів. Цей інтерес обумовлюється багатьма чинниками: змістом предмету, що вивчається, рівнем його складності; організацією процесу навчання; системою контролю і оцінювання знань, що використовує викладач; особистими якостями самого викладача (його педагогічною майстерністю, фаховою підготовкою, вмінням зацікавити студентів предметом); системою цінностей студента, його найближчого оточення; взаємостосунками в студентському колективі; соціальним замовленням на підготовку фахівців з галузі науки, що представлена даним предметом. Тому одним з важливих завдань дидактики вищої школи є активізація навчально-пізнавальної діяльності студентів ([2], [85], [101], [180], [186], [227] та ін.), яка „розуміється як цілеспрямована діяльність викладача, спрямована на розробку і використання такого змісту, форм, методів, прийомів і засобів навчання, які сприяють підвищенню пізнавального інтересу, активності, творчої самостійності студентів у засвоєнні знань, формуванні навичок і вмінь, застосуванні їх на практиці” [186]. Для формування творчої особистості необхідно застосовувати такі форми навчальної діяльності, які сприяють різнобічному розвитку особистості й переходу на рівень творчості як стилю діяльності.

Одним з шляхів активного впливу на процес навчання є покращення функціональних характеристик самого процесу навчання, що надають можливість активізувати пізнавальну активність й інтерес студентів до

навчального процесу, наслідком чого стане підвищення їхньої успішності. На наш погляд, у процесі навчання основ програмування ця мета досягається впровадженням диференційованого підходу ([107], [132], [193], [206] та ін.).

Ю.К. Бабанський визначає поняття „диференційований підхід" як спосіб оптимізації, який передбачає оптимальне поєднання класних, групових та індивідуальних форм навчання [17]. З.І. Слєпкань визначає, що „диференційований підхід – це особливий підхід викладача до різних груп або до окремих студентів, що полягає в організації з ними на основі різної за змістом, обсягом і складністю (хоч за одними навчальними програмами) роботи з урахуванням однакових, в основному, властивостей особистості" [188].

У методичному аспекті проблема диференціації навчання розглядається в роботах О.А. Кузнєцова, Т.Б. Захарової [107], В.М. Монахова В.О. Орлової, В.В. Фірсова [132] та ін. Диференціація навчання в цих дослідженнях розуміється як організація і методика навчання, при якій кожен учень, опановуючи деяким мінімумом загальноосвітньої підготовки, що є загальнозначущою і забезпечує можливість адаптації в життєвих умовах, що постійно змінюються, одержує право і гарантовану можливість приділяти переважну увагу тим напрямам, що найбільшою мірою відповідають його схильностям.

М.М. Шахмаєв виділяє зовнішню і внутрішню диференціації [219]. *Зовнішня диференціація* передбачає врахування особливостей учнів за допомогою їхнього об'єднання в окремі групи, торкається самої структури навчання і може бути реалізована в таких формах як організація груп, для яких навчальний план пристосований до інтересів і потреб учнів; навчання за вибором (обов'язкове або добровільне), при якому можливе поглиблене вивчення обов'язкових предметів, вивчення додаткових предметів, факультативне вивчення предметів; перерозподіл звичайних класів відповідно до рівня успішності учнів і вимог до навчання; організація спеціальних класів

і шкіл для навчання учнів, що мають глибокий інтерес і здібності до визначеної галузі знань. *Під внутрішньою диференціацією* розуміють використання на заняттях з учнями однієї групи різних методів і засобів, що забезпечують максимальний розвиток здібностей, схильностей, задоволення пізнавальних потреб та інтересів кожного учня. Розглядаючи внутрішню диференціацію, прийнято враховувати, що при цьому враховуються індивідуальні особливості, які притаманні групам учнів, і організація варіативного навчального процесу в цих групах.

Під *диференційованим підходом* у навчанні основ програмування майбутніх інженерів-програмістів будемо розуміти організаційний аспект навчання, що передбачає використання різних форм внутрішньої диференціації. А саме, дозована допомога студентам з боку викладача, диференційовані завдання за рівнями складності, групова робота зі студентами (див. п. 2.3.2.5.).

Диференційований підхід у навчанні основ програмування надає можливість враховувати прагнення студентів до здобування нових знань, розкрити індивідуальні здібності до навчання за допомогою ефективного поєднання різних методів, засобів і організаційних форм навчання; сприяти формуванню творчої, інтелектуальної особистості.

1.4.5. Модульно-рейтингова система навчання основ програмування.

Модульно-рейтингова система навчання складається з двох взаємозв'язаних частин, що доповнюють одна одну, модульної і рейтингової, які можуть функціонувати як разом, так і окремо і є зведенням правил і положень, на основі яких в ході навчання здійснюється оцінювання навчально-пізнавальної діяльності студента ([10, 11], [18], [67], [88], [119], [136], [156], [175], [200], [226] та ін.).

О.І. Смолюк визначає модуль як „цільовий функціональний вузол, у якому навчальний зміст професійної освіти і технологія оволодіння ним об’єднуються в систему високого рівня об’єктивної цілісності” [189];

А.М. Алексюк визначає модуль як відносно самостійну частину програми, яка об’єднує певні науково-пізнавальної ідеї і містіть одне або декілька фундаментальних понять, закономірностей, близьких за змістом, або, в окремих випадках, законів [3]. У [119] навчальний модуль розглядають як цілісну одиницю навчального процесу, що має відносну самостійність в рамках навчального курсу і що охоплює основні компоненти навчання. Кожен модуль виступає як одинична частина навчального методичного комплексу і завершується проміжним контролем.

Як вказують І.Ф. Прокопенко, В.І. Євдокимов, „модульна форма організації навчання передбачає поділ матеріалу курсу на навчальні модульно-самостійні розділи або теми курсу з відповідною системою організації навчання, методичного забезпечення, діагностики та контролю навчального процесу” [167].

Модульність навчання передбачає побудову процесу навчання на наступних принципах ([11], [67], [119], [156], [177], [202], [226]):

- побудова навчального матеріалу на засадах формування функціональних змістових блоків-модулів із чітко визначеними конкретними дидактичними цілями, організаційними формами і методами подання визначеного змісту;
- структурування змісту на окремі блоки-модулі як відносно самостійні структури, тісно пов’язані між собою дидактичними цілями, наступністю і послідовністю змісту, методами його оволодіння, а також поступовим ускладненням завдань самостійності учіння;
- гнучкість побудови модулів передбачає можливість пристосування змісту навчального матеріалу та шляхів оволодіння ним до індивідуальних можливостей і потреб студентів;
- усвідомленість перспективи співпраці викладача і студента вимагає розуміння необхідності постійного й системного оволодіння змістом, методами учіння, різноманітністю, методами навчання-консультування, координування розвитку пізнавальних інтересів.

Рейтингова система оцінювання навчальних досягнень ґрунтується на інтегральній оцінці результатів усіх видів навчальної діяльності студента (виконання і захист лабораторних робіт, написання контрольних робіт, самостійна робота, створення проекту, і т.п.) і надає можливість на основі об'єктивних критеріїв оцінити рівень знань, вмінь і навичок студента з дисципліни, що вивчається, у вигляді рейтингу. У загальному значенні, рейтинг це оцінка, індивідуальний показник оцінки популярності, авторитету, цінності кого-, чого-небудь, який ґрунтується на результатах голосування, соціологічних опитувань, а також власне сама така оцінка, ступінь популярності, цінності кого-, чого-небудь [170]. Щодо навчальної діяльності найбільше точне означення рейтингу дане Ю.В. Триусом: „*Рейтинг* (англ. rating – оцінка, порядок) – це комплексний кількісний показник, який враховує результати навчально-пізнавальної діяльності студента з усіх видів навчальних занять і самостійної роботи з конкретної дисципліни протягом семестру, навчального року і за весь період навчання. У рейтинговому показнику можна враховувати будь-яку форму діяльності студента, характер праці на заняттях, якість аудиторної і самостійної роботи, рівень мислення, інтелектуальний потенціал, здатність творчо реалізувати його на практиці” [201].

Рейтинг є активним показником навчальної діяльності студента. Рейтингова система оцінювання призначена для підвищення об'єктивності і достовірності оцінювання рівня підготовки фахівців і використовується як один з елементів управління навчальним процесом у ВНЗ. За допомогою рейтингової системи оцінювання навчальних досягнень не лише контролюється рівень знань студентів, але й активізується їхня пізнавальна діяльність у навчанні, стимулюється якісніше виконання аудиторних і позааудиторних навчальних завдань, під час навчання, а не лише в короткий час перед іспитом.

На сучасному етапі використання інноваційних технологій в освітньому процесі практикується об'єднання модульної системи навчання і рейтингової системи контролю і оцінювання навчальних досягнень студентів.

Застосування модульно-рейтингової системи навчання надає можливість ([11], [67], [200], [226] та ін.):

- одержувати об'єктивні показники рівня знань з окремих модулів курсу;
- підвищити об'єктивність процедури оцінювання рівня знань студентів;
- активізувати навчально-пізнавальну діяльність студентів;
- позитивно впливати на навчальну дисципліну студентів;
- підвищити якісний рівень засвоєння знань студентами;
- діагностувати здібності студентів;
- своєчасно вносити корективи в організацію навчального процесу за результатами поточного рейтингового контролю;
- організовувати систематичну роботу студентів по засвоєнню навчального матеріалу, що, в свою чергу, сприяє підвищенню міцності знань і побудові фундаменту для подальшого навчання;
- своєчасно визначати прогалини в засвоєнні навчального матеріалу;
- враховувати індивідуальні особливості студента;
- здійснювати диференційований підхід у навчанні;
- визначати особистий рейтинг кожного студента;
- прогнозувати підсумкову оцінку з дисципліні.

Відзначимо, що модульно-рейтингова система навчання спонукає до активної продуктивної праці зі зворотним зв’язком і до творчого пошуку як студентів, так і викладачів.

Наш досвід роботи показує, що модульно-рейтингова система навчання забезпечує дотримання основних принципів контролю і оцінки знань студентів [3]:

- принцип індивідуального характеру перевірки й оцінювання знань студентів, який передбачає індивідуальну роботу викладача з кожним студентом;

- принцип системності перевірки й оцінки знань, якій вимагає здійснення контролю протягом усього періоду навчання студента;
- принцип тематичності, якій передбачає оцінку навчальної діяльності студента не тільки за її кінцевими результатами за семестр чи навчальний рік, а й за кожною темою, що вивчається;
- принцип диференційованої оцінки успішності навчання студентів, який передбачає здійснення її на основі визначення однакових рівнів засвоєння знань;
- принцип єдності вимог викладачів до студентів, згідно якому викладач зобов'язаний ставити єдині вимоги до аналізу й оцінювання ходу вивчення і засвоєння майбутніми спеціалістами навчального матеріалу в цілковитій відповідності із кваліфікаційними характеристиками, чинними навчальними планами і програмами, сприяючи посиленню особистої відповідальності студента за якість своєї навчальної праці;
- принцип об'єктивності, якій передбачає постійну увагу викладача до підтримання необхідного рівня підготовки майбутніх спеціалістів, до системного аналізу результатів міжсесійного контролю і показників успішності з метою своєчасного здійснення заходів для покращення організації і змісту навчально-виховного процесу, підвищення ефективності і якості аудиторних і самостійних занять студентів.

Відомо, що оцінювання рівня знань має виховний характер ([12], [37], [110], [210], [227]). Сприйняття оцінки студентом може бути неадекватно тим об'єктивним показникам, які характеризують міру підготовленості студентів на кожному рівні навчання при використанні модульно-рейтингової системи навчання. Тому у процесі навчання, у цілому, та при контролі знань, зокрема, слід враховувати індивідуально-психологічні особливості студента; проблеми адаптації студента до навчального процесу у ВНЗ, оскільки курс з основ програмування вивчається на першому і другому курсах навчання.

Висновки до розділу 1

У результаті аналітичної роботи в межах дисертаційного дослідження зроблено такі висновки:

1. Курс з основ програмування для майбутніх інженерів-програмістів становить теоретичну та практичну базу для комп’ютерних дисциплін; знання, вміння і навички з основ програмування є фундаментом для навчання інших комп'ютерних дисциплін і основою професійної підготовки майбутнього програміста.

2. Фахівцям у галузі комп'ютерних наук властиві риси, якості, здібності й особливості мислення, деякі з котрих можна і треба сформувати при навчанні основ програмування, зокрема здатність визначити архітектуру програми, здатність налагоджувати програми, здатність розуміти програми, уміння бачити задачу одночасно на різних рівнях деталізації, уміння застосовувати типові алгоритми, здатність модифікувати програму.

3. З метою охоплення всієї сукупності необхідних знань з основ програмування, для поліпшення якості навчання і для надання студентам достатньої кількості часу на опанування матеріалу, курс з основ програмування повинен складатися з пропедевтичної та основної частин при тій же кількості відведених під цю дисципліну кредитів (годин). При цьому: а) пропедевтичне навчання основ програмування повинне здійснюватися за допомогою мови програмування Pascal; б) основне – з використанням мови програмування С; в) навчати треба структурним принципам побудови програм; г) зміст курсу доцільно погодити зі рекомендаціями, що запропоновані у документах „Computing Curricula 2001” та „Software Engineering 2004”.

4. З метою покращення функціональних характеристик процесу навчання, які надають можливість активізувати пізнавальну активність студентів і підвищити їхню успішність у навчанні, доцільно використовувати диференційований підхід у навчанні основ програмування.

5. Для формування у студентів здатності використовувати отримані знання в процесі навчання й у наступній професійній діяльності треба у процесі навчання основ програмування акцентувати їхню увагу на результатах навчання, що забезпечується використанням компетентністного підходу.

6. Для забезпечення активізації навчально-пізнавальної діяльності студентів, позитивного впливу на навчальну дисципліну студентів, підвищення якісного рівня засвоєння знань студентами, систематичної роботи студентів щодо засвоєння навчального матеріалу, діагностики здібності студентів до навчання, об'єктивності процедури оцінювання навчальних досягнень студентів, своєчасної корекції організації навчального процесу треба впроваджувати модульно-рейтингову систему навчання основ програмування.

Основні результати першого розділу опубліковано у роботах [24], [54], [57], [60], [63], [202].

РОЗДІЛ 2

МЕТОДИЧНА СИСТЕМА НАВЧАННЯ ОСНОВ ПРОГРАМУВАННЯ

2.1. Мета, завдання і функції навчання курсу з основ програмування

У роботі С.У. Гончаренка дано означення методики навчального предмету [46]: „Методика конкретного навчального предмета – це галузь педагогічної науки, що досліджує зміст навчального предмета й характер навчального процесу, який сприяє засвоєнню учнями необхідного рівня знань, умінь та навичок, розвитку мислення школярів, формуванню світогляду і виховання якостей громадянина своєї країни. … До завдань методики входить дослідження змісту навчання, процесу викладання й процесу учіння”. Також автор відзначає, що „методика глибоко пов'язана з відповідною наукою, оскільки в навчанні треба відображати особливості відповідної науки, її зміст і методи дослідження”. На наш погляд, таке розуміння методики навчання предмету є прийнятним і для дисциплін, що вивчаються у ВНЗ, у тому числі й для курсу з основ програмування.

А.М. Пишкало визначив методичну систему навчання як структуру, що складається з наступних основних елементів: цілей навчання, змісту навчання, методів, форм і засобів навчання [171]. Ефективність методичної системи навчання визначається ступенем досягнення поставлених перед нею цілей ([71], [151], [208]).

Цілі навчання основ програмування випливають із загальних цілей освіти, а також з особливостей програмування, як дисципліни, його ролі й місця в системі навчання інженерів-програмістів (див. п. 1.1). *Головною метою навчання* майбутніх інженерів-програмістів є різнобічний розвиток особистості, її здібностей, формування фахівця з комп’ютерних наук, конкурентноздатного на міжнародному ринку інтелектуальної праці громадянина України, здатного працювати на користь своєї батьківщини.

Програмування – перша серед професійно-орієнтованих дисциплін, яку вивчають майбутні програмісти. Від змісту курсу, від системи знань, умінь і навичок, яка буде сформована у студентів у процесі навчання цього курсу, залежить якість засвоєння ними дисциплін, заснованих на знаннях із програмування, успішність у майбутній професійній діяльності.

Ми вважаємо, що *головним завданням* при навчанні основ програмування майбутніх програмістів повинно бути навчання методам побудови ефективних алгоритмів, їх аналізу та реалізації сучасними засобами програмування на комп'ютері.

Виходячи із зазначеного, визначимо *основні функції* навчання основ програмування [202].

Загальноосвітня функція навчання основ програмування полягає в тому, щоб забезпечити в студентів:

формування знань про:

– фундаментальні концепції і парадигми програмування;

– методологію побудови інформаційних моделей;

– базові принципи побудови алгоритмів і програм;

– модульний принцип розробки програм;

– типові алгоритмічні конструкції: слідування, розгалуження, повторення;

– синтаксис та семантику мов програмування високого рівня;

– етапи обробки програм на ПК, таких як: редагування, трансляція, компонування;

– типи і структури даних та алгоритми їх обробки;

– типи підпрограм, механізм передачі параметрів-значень та параметрів-змінних;

– методи структурного низхідного програмування;

– рекурсивні алгоритми та способи їх побудови;

– методи динамічного розподілу пам'яті;

– критерії ефективності алгоритмів і програм;

– засоби для розробки, супроводу та організації багатофайлових прикладних програм;

формування вмінь і навичок практичного програмування мовами програмування високого рівня Pascal та C, а саме:

- складати лінійні, розгалужені та циклічні програми з використанням стандартних та структурованих типів даних;
- розробляти алгоритми і програми методом покрокового уточнення;
- налагоджувати й тестувати навчальні програми в середовищі розробки мови програмування;
- використовувати підпрограми при складанні програм мовою Pascal та складати програми мовою C, які містять одну або декілька функцій;
- використовувати й адаптовувати стандартні алгоритми;
- будувати алгоритми, ефективні за часом та пам'яттю;
- реалізовувати багатомодульні програми;
- виконувати роботу, що включає опис проблеми, її аналіз, проектування, розробку й модифікацію невеликих програмних проектів, а також підготовку необхідної документації.

Важливу роль відіграє курс з основ програмування у формуванні загальнопізнавальних, організаційних, комунікативних умінь і навичок студентів, а також у формуванні і розвитку в них потреби постійно розширювати і поглиблювати свої знання. Тому, *розвиваюча функція* навчання основ програмування полягає в тому, щоб сприяти:

розвитку:

– алгоритмічного стилю мислення;

– інтелектуальних якостей і творчих здібностей, які потрібні для розуміння програм;

– критичного та творчого мислення;

– інформаційної культури;

– професійної інтуїції;

– здатності до порівняльного/критичного аналізу;

формуванню здатності:

– бачити задачу одночасно на різних рівнях деталізації;

– узагальнювати типові ситуації;

– до вивчення нових мов програмування;

– використовувати знання і навички в новій ситуації;

формуванню:

– розуміння зв'язку теорії програмування із практикою програмування;

– навичок самостійної роботи з навчальною та науковою літературою при розв’язуванні завдань з програмування;

– розуміння галузі застосування здобутих знань в інших професійно-орієнтованих і спеціальних дисциплінах та в сфері майбутньої професійної діяльності студентів;

– наукового світогляду;

– готовності до навчання протягом усього життя.

Виховна функція навчання основ програмування полягає у формуванні в студентів потреби і уміння використовувати науковий зміст основ програмування для усвідомлення цілісної картини процесу розв’язування пізнавальних і професійних задач або проблем, а також, пов’язана з формуванням у студентів якостей, які притаманні фахівцям із комп’ютерних наук та інженерам-програмістам (див. п. 1.3.) у тому числі:

– педантичність, дисциплінованість, акуратність;

– наполегливість;

– внутрішня керованість;

– низька нетерпимість до невизначеності;

– усвідомлення особистої відповідальності за результати своєї роботи;

– прагнення до самоутвердження через творчу діяльність;

– уміння працювати як індивідуально, так і як лідер або член команди;

– бережливе відношення до техніки;

– системність і цілеспрямованість у професійній діяльності.

Для досягнення поставленої мети в курсі „Програмування” передбачається ознайомлення студентів з:

– історією розвитку мов програмування;

– парадигмами програмування;

– поняттям синтаксису та семантики мов програмування;

– об’єктами даних (зміни та константи, атрибути та зв'язування);

– елементарними типами даних (числові типи, символьний тип, булевський тип, покажчик), засобами їхнього представлення та алгоритмами обробки;

– структурованими типами даних (рядки, масиви, асоціативні масиви, записи, об'єднання, множини, файли, списки), засобами їхнього представлення та алгоритмами обробки;

– контролем та перетворенням типів;

– структурами управління на рівні операторів (оператор присвоювання, складений оператор, оператор розгалуження, оператор циклу, безумовний перехід);

– методами побудови алгоритмів;

– основами аналізу алгоритмів;

– технологіями побудови програмних комплексів.

Курс повинен забезпечити оволодіння методологією структурного програмування. Із цією метою розглядаються:

– засоби декомпозиції програм;

– методологія покрокової розробки програм та проектування “з гори до низу”.

Студенти після вивчення курсу повинні знати:

– етапи обробки програм на ПЕОМ, такі як: редагування, трансляції, компонування;

– структуру програми мовою Pascal;

– структуру програми мовою С;

– методи представлення алгоритмів;

– синтаксис та семантику мов програмування Pascal та С;

– управляючі конструкції алгоритмів: послідовність, вибір, повторення;

– властивості алгоритмів;

– класифікацію операцій (арифметичні операції, операції відношення, логічні операції, порозрядні бітові та операції зсуву, операції над рядками, операції над множинами);

– пріоритет виконання операцій;

– структурну методологію розробки програм;

– класифікацію структур даних;

– скалярні типи даних;

– структуровані типи даних: масиви, рядки, записи, множини, файли, списки;

– оператори (прості, структуровані);

– методи побудови алгоритмів;

– бібліотечні підпрограми мови Pascal;

– графічні засоби мови Pascal та С;

– бібліотечні функції мови С;

– типи підпрограм у мові Pascal, механізм передачі параметрів-значень та параметрів-змінних;

– механізми використання функцій у мові С;

– класи пам'яті мови С;

– форми рекурсивних алгоритмів;

– засоби препроцесорної обробки програми;

– модульний принцип розробки програм;

– методи оцінки швидкодії алгоритмів;

– методи аналізу алгоритмів;

– програмування модулів мови Pascal;

– парадигми програмування;

– технології проектування програмних комплексів;

уміти:

– формалізувати задачу;

– визначати структури даних для розв'язування даної задачі;

– будувати інформаційну модель задачі;

– розробляти алгоритми методом покрокового уточнення;

– будувати декілька різних алгоритмів для розв'язування однієї задачі;

– подавати алгоритми в різних формах;

– складати лінійні програми, програми з розгалуженням, циклічні програми мовою Pascal та C;

– використовувати скалярні типи даних: цілі, дійсні, символьні, логічні;

– налагоджувати програму в середовищі Borland Pascal та C;

– користуватися довідковою системою середовища програмування;

– підключати процедури та функції стандартної бібліотеки модулів мови Pascal;

– підключати функції стандартної бібліотеки мови C;

– використовувати структуровані типи даних при складанні програм;

– ділити складну задачу на елементарні складові і задавати їхнє комбінування;

– використовувати власні процедури та функції при складанні програм мовою Pascal;

– використовувати власні функції при складанні програм мовою C;

– оцінювати швидкодію алгоритмів;

– будувати алгоритми з використанням мінімального об’єму пам'яті;

– застосовувати і комбінувати відомі прийоми програмування і типові алгоритми;

– застосовувати методи побудови алгоритмів;

– реалізувати багатомодульні програми;

– складати і використовувати макрозасоби та засоби препроцесорної обробки програм у мові С;

а також:

– організовувати пошук відомостей, необхідних для розв’язування задачі;

– писати програми, правильно і добре документовані;

– аналізувати власні помилки правильно і однозначно висловлювати думку в зрозумілій співбесіднику формі;

– розуміти готові програми;

– розробляти програмний продукт мікрогрупами та колективно.

Багато педагогів (Ю.І. Машбиць [124], Н.Ф. Тализіна [196] та ін.) підкреслюють неприпустимість зведення навчальних цілей тільки до засвоєння предметних знань і вмінь. На їхній погляд, у систему цілей необхідно включати вміння вчитися. Це твердження особливо актуально для дисциплін з комп'ютерних наук. Відомо, що комп'ютерні науки в даний час продовжують стрімко розвиватись – “нові технології, що з'являються, застарівають практично відразу після виникнення” [29].

2.2. Зміст курсу з основ програмування

Проблема змісту навчального курсу є найважливішою дидактичною проблемою. У змісті курсу з основ програмування повинні відображатися не лише поняття, закономірності, певні теорії й факти науки програмування, але і сучасні тенденції і напрями її розвитку. Також повинна проектуватися й діяльність студента щодо засвоєння виділених основ через комплекс спеціально

підібраних завдань, які моделюють основні типи професійних завдань фахівця ([58], [62]). При підготовці майбутнього інженера-програміста важливо звертати увагу на соціальний аспект його професійної діяльності, на виробничі відносини, які передбачають здатність до колективної роботи над одним великим завданням.

У світлі сучасних тенденцій та інтеграції європейських країн у сфері вищої освіти для побудови загальноєвропейського освітнього простору й приєднання України до Болонського процесу, метою якого є створення умов для розвитку і модернізації систем вищої освіти європейських країн, розробці єдиних освітніх стандартів [147], при формуванні змісту курсу з основ програмування для майбутніх інженерів-програмістів ми орієнтуємось на документи „Рекомендації з викладання інформатики 2001” [230] і „Рекомендації з викладання програмної інженерії і інформатики” [235]. При цьому, ми вважаємо за потрібне:

– узгодити зміст робочої програми курсу із суміжними дисциплінами навчального плану;

– враховувати зміни, які відбуваються в інформаційних технологіях, інноваційні підходи і педагогічні технології, які використовуються у вищій школі, а також зростаючу важливість навчання протягом всього життя.

Виходячи з вищесказаного, визначимо основні принципи побудови курсу з основ програмування, врахування яких надасть можливість певним чином структурувати і відбирати навчальний матеріал.

2.2.1. Принципи визначення змісту курсу з основ програмування для майбутніх інженерів-програмістів .

Виходячи з того, що знання, уміння й навички, одержані студентами, майбутніми інженерами-програмістами, у курсі з основ програмування, по-перше, є фундаментом для вивчення дисциплін циклу комп'ютерних наук, тобто визначають рівень ефективності й результативності вивчення професійно-орієнтованих дисциплін, по-друге, вони необхідні в професійній

діяльності фахівця в галузі комп'ютерних наук та інформаційно-комунікаційних технологій, сформулюємо основні принципи визначення змісту курсу з основ програмування:

1. *Принцип відповідності навчальним цілям.* Цілі навчання основ програмування визначаються виходячи із загальних цілей освіти – навчання, розвиток, виховання, а також з особливостей програмування, як дисципліни, її ролі і місця в системі навчання майбутніх інженерів-програмістів.

2. *Принцип науковості.* 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 обраної парадигми програмування.

3. *Принцип фундаментальності.* У 1992 році, у статті під назвою „Ми можемо навчати інформатиці краще" Мері Шоу висловила це міркування таким чином: „Давайте організуємо наші курси навколо ідей, а не навколо артефактів. Це зробить наші цілі яснішими, як для студентів, так і для викладачів. Машинобудівні інститути не викладають проектування бойлера – вони викладають термодинаміку" [236]. При цьому зміст повинний бути настільки складним, щоб допускати конструювання на його базі нового змісту потрібної складності.

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

5. *Принцип сучасності.* Швидкий розвиток інформаційно-комунікаційних технологій вимагає регулярного перегляду навчальної програми дисципліни з метою модернізації застарілих компонентів.

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.

7. *Принцип вирівнювання знань.* Зміст курсу з основ програмування повинен включати пропедевтичний модуль з основ алгоритмізації і програмування. Призначенням такого модулю є зменшення розшарування студентів, що вступили до ВНЗ, за знаннями з алгоритмізації і програмування, одержаними в школі.

2.2.2. Теоретичні основи курсу.

У методичній системі навчання основ програмування, що пропонується, для різних видів занять – лекцій і лабораторних робіт – обрано різні стратегії опанування навчального матеріалу.

Зміст модулів і тем лекційного курсу визначається на основі підходу „із максимальним охопленням матеріалу” (див. п. 1.2.). Перелік модулів лекційного курсу подано в таблиці 2.1.

Таблиця 2.1

Зміст теоретичної частини курсу

Модуль 1. Вступ до програмування	*Змістовий модуль 1.* Програмування як наука
	Змістовий модуль 2. Основи алгоритмізації і програмування. Навчальна мова програмування Pascal
Модуль 2. Мова програмування С	*Змістовий модуль 3.* Основи мови програмування С
Модуль 3. Алгоритми: побудова і аналіз	*Змістовий модуль 4.* Сортування і порядкові статистики
	Змістовий модуль 5. Абстрактні типи даних
	Змістовий модуль 6. Методи побудови і аналізу алгоритмів
Модуль 4. Основи розробки програмного забезпечення	*Змістовий модуль 7.* Основи розробки програмного забезпечення
Модуль 5. Огляд парадигм програмування	*Змістовий модуль 8.* Парадигми програмування

Кількість годин, відведених на вивчення кожного модуля і кожної теми, залежить від загальної кількості лекційних годин, передбачених для дисципліни і від спеціальності, на якій вивчається курс з основ програмування. Розподіл годин за темами для спеціальності «Прикладна математика» наведено у додатку А.

Теми модулів для різних спеціальностей можуть розрізнятися. Наприклад, для спеціальності „Прикладна математика” при розробці робочої програми слід більше часу відвести математичному апарату аналізу алгоритмів, методам розробки ефективних алгоритмів. Для інженерної спеціальності „Програмне забезпечення автоматизованих систем” більше уваги слід приділити питанням подання даних у пам'яті віртуальної машини і на апаратному рівні. Зауважимо, що окремі теми для деяких спеціальностей можуть не розглядатися. Наприклад, для спеціальності „Комп'ютерно-інтегровані технологічні процеси і виробництва”, на наш погляд, можна не включати до робочої програми розділ „Технологія побудови програмного комплексу”, а також не включати теми „Вступ до логічного програмування”, „Вступ до функціонального програмування”, „Вступ до програмування під Internet”.

Але повинна існувати *інваріантна частина змісту курсу* з основ програмування, яка є необхідним фундаментом для вивчення суміжних дисциплін і для майбутньої професійної діяльності (див. 1.2, табл. 1.1).

2.2.3. Зміст лабораторного практикуму.

Програмування – це один із видів людської діяльності, що потребує надзвичайної точності, педантичності. Прищеплення студентам цих якостей при створенні власних програмних продуктів є найбільш складним у процесі навчання програмування. Сформувати у майбутніх інженерів-програмістів ці важливі риси можна шляхом навчання програмування на базі структурного підходу (див. п. 1.4.).

Мови програмування – це основний засіб „спілкування” програміста з комп’ютером. На сучасному етапі розвитку інформаційних технологій від програміста вимагається вміння програмувати багатьма мовами і, навіть, стилями програмування одночасно. Володіння різними парадигмами програмування, знання декількох мов, розуміння головних відмінних рис мов програмування значно полегшує програмісту засвоєння нових мов програмування і дозволяє крокувати разом з усією індустрією створення програмного забезпечення.

При підготовці майбутніх інженерів-програмістів пропонується така послідовність навчання: від засвоєння мов програмування Pascal і C, що підтримують імперативну парадигму програмування, до об’єктно-орієнтованих (C++, Java).

Отже, у курсі з основ програмування пропонується вивчення двох мов програмування: *мови Pascal і мови C.*

Вибір мови програмування Pascal обумовлений тим, що, по-перше, ця мова спеціально створювалася для навчання програмуванню; по-друге, основи цієї мови, як правило, відомі студентам зі шкільного курсу інформатики, оскільки в більшості регіонів України в шкільному курсі інформатики навчають алгоритмізації і програмуванню саме на прикладі мови програмування Pascal.

Завдяки цьому підходу можна, 1) забезпечити однаковий стартовий рівень знань, умінь і навичок з алгоритмізації у студентів для вивчення основних розділів курсу з основ програмування; 2) удосконалити знання студентів з мови Pascal; 3) починаючи вже з першого семестру навчання розв’язувати алгоритмічно складні задачі; 4) використовуючи те, що в конструкціях мов програмування Pascal і C є багато спільного, забезпечити вивчення мови програмування C у порівнянні з мовою Pascal, що сприяє більш глибокому засвоєнню кожної із цих мов.

Мова програмування C обрана з міркувань перспективності основних практичних галузей застосування програмування:

1) С – універсальна мова програмування, яка тісно пов'язана із системним програмуванням, що є суттєвим для майбутніх розробників програмного забезпечення;

2) у даний час кількість професійних програмістів, що використовують мову С, швидко зростає;

3) синтаксис і семантика більшості нових мов (таких як Java і C#), частково засновані на концепціях, характерних для мови С;

4) знання синтаксису і семантики мови С послужать основою для навчання мови С++. Опитування, яке було проведене серед керівників американських комп'ютерних компаній показало, що 46% респондентів вважають найбільш перспективною мовою програмування мову С++ [140]. Автор мови програмування С++ Б. Страуструп визначає, що „за виключенням несуттєвих деталей, С++ є надмножиною мови С" [195];

5) мова С, разом із мовами С++ і Java, які розроблялися на його основі, найбільше впливають на розвиток програмування;

6) знання мови програмування С може використовуватися в різних прикладних галузях, тому що ця мова має достатню кількість управляючих операторів і можливість структуризації даних.

Слід також зазначити, що знання мови С слугуватимуть студентам базою для вивчення дисциплін „Об'єктно-орієнтоване програмування", „Системне програмування", „Операційні системи".

У додатку А наведено зміст лекційного курсу з розподілом годин для спеціальності «Прикладна математика».

2.3. Сучасні принципи, підходи і методи навчання основ програмування

2.3.1. Принципи навчання основ програмування. Принципи навчання – це керівні ідеї, нормативні вимоги до організації і здійснення освітнього процесу [133]. Для успішної структуризації процесу навчання слід керуватися загально

дидактичними принципами навчання (С.І. Архангельський [12], Ю.К. Бабанський [16], І.Я. Лернер [117], М.І. Махмутов [123], М.М. Скаткін [184] та ін.).

У вищій школі принципи навчання розділяють на дві групи [3]. До першої групи відносять принципи, які відображають закономірності зв'язку мети і змісту навчання: розвиваючого і виховного характеру навчання; науковості навчання; активності і самостійності студентів у навчанні; систематичності навчання; зв'язку змісту навчання з життям, практикою. До другої групи відносять принципи моделювання умов продуктивного навчання: принципи наочності в навчанні; врахування вікових та індивідуальних особливостей студентів; міцності знань; об'єднання колективних, групових і індивідуальних форм організації навчання; зв'язку лекційної і поза лекційної навчально-виховної роботи зі студентами; керівної ролі викладача, кафедри та ін.

Розглянемо питання реалізації принципів навчання для організації навчальної діяльності при навчанні основ програмування.

Принцип спрямованості процесу навчання на гармонійний розвиток особистості. Згідно з цим принципом, процес навчання основ програмування покликаний формувати у студента різносторонні знання, уміння і навички, які відповідають рівню розвитку сучасних комп'ютерних наук і комп'ютерної техніки; розвивати вольову, інтелектуальну, емоційну сфери особистості; формувати суспільно значущі мотиви діяльності та інтереси, які істотним чином впливають на майбутню професійну діяльність і широкий розумовий кругозір.

Досвід роботи показує, що навчання основ програмування сприяє формуванню і розвитку пізнавальних інтересів студентів у тому випадку, якщо існує взаємозв'язок між теоретичним матеріалом і практичними завданнями; лекційні заняття чергуються із лабораторними заняттями; здійснюється підбір завдань з урахуванням індивідуальних здібностей студентів.

Принцип науковості навчання. Слід зазначити, що комп'ютерні науки і сьогодні знаходяться в стадії розвитку і що в цій галузі, як ні в якій інший, студенти зустрічаються з новими науковими і технічними термінами, з новими поняттями.

Власний досвід підтверджує, що для реалізації принципу науковості навчання основ програмування доцільно використовувати метод наукового пізнання; знайомити студентів з історією розвитку обчислювальної техніки і комп'ютерних наук; надавати можливість студентам переживати радість власного відкриття при розв’язуванні задачи; привчати студентів читати періодичну літературу, науково-популярні журнали і робити повідомлення про прочитане; сприяти участі студентів в олімпіадах з програмування і наукових студентських конференціях.

Разом із принципом науковості при навчанні основ програмування слід дотримуватися *принципу доступності,* суть якого полягає в тому, що зміст, форми і методи навчання повинні враховувати реальні можливості студентів.

Принцип створення відповідних умов для функціонування процесу навчання. Цей принцип є особливо важливим при навчанні комп'ютерним наукам і, у тому числі, основ програмування, оскільки комп'ютерна техніка, що використовується в процесі навчання, повинна відповідати санітарним нормам. Крім того, у процесі навчання слід дотримуватися норми температурного режиму, освітленості, провітрювання в навчальних приміщеннях і у комп'ютерних класах.

Згідно з цим принципом, на викладача лягає відповідальність забезпечення сприятливої морально-психологічної атмосфери під час проведення занять, передбачених навчальним планом дисципліни, зокрема, і в студентському колективі у цілому.

Принцип систематичності і послідовності навчання. За цим принципом передбачається формування знань, умінь і навичок [17] „в певному порядку, в

системі: кожний елемент навчального матеріалу логічно пов'язується з іншими, наступне спирається на попереднє і готує до засвоєння нового”.

Для навчання інформатики А.П. Єршов запропонував реалізацію принципу послідовності у формі циклічності. Реалізація цього принципу передбачає розподілення навчального матеріалу на окремі модулі, які, у свою чергу, повинні складатися з окремих тем, логічно пов'язаних між собою; визначення теми на лекційних заняттях, побудову логічно обґрунтованої опорної схеми подання змісту навчального матеріалу; формулювання мети заняття на лабораторних роботах, демонстрацію можливості і способів досягнення мети заняття на конкретних прикладах і задачах.

Принцип свідомості, активності і самостійності студентів у навчанні при керівній ролі викладача. Цей принцип спрямований на формування особистості, яка має соціально значущою життєвою позицією. Свідоме здобування знань студентами залежить від таких чинників як організація навчально-виховного процесу; від методів і засобів навчання; від рівня пізнавальної активності студентів, від мотивів навчання. Для реалізації принципу свідомості при навчанні основ програмування викладачу слід пояснювати значення і важливість навчального матеріалу кожного заняття; забезпечувати розуміння студентом кожного поняття, що вводиться в процесі навчання; навчати так, щоб студент розумів, які і у якій послідовності треба виконувати дії для розв’язування поставленої перед ним задачі, запобігаючи механічному, неусвідомленому виконанню завдань; навчати студентів знаходити декілька різних алгоритмів розв’язування задачі; використовувати можливості взаємонавчання студентів; забезпечувати розвиток інтелектуальних здібностей студентів через їх усвідомлене ставлення до здобування знань.

Принцип активності підкреслює функцію студента як суб'єкта навчання, а не пасивного учасника процесу. У методиці навчання інформатики розрізняють інтелектуальну та емоційну активність [130]. Інтелектуальна

активність у навчанні основ програмування забезпечується постановкою проблемних задач. Наприклад:

Задача 1. Обміняти місцями значення цілих змінних *x* та *y*, не використовуючи додаткові змінні.

Задача 2. Записати арифметичний вираз, значенням якого є більше з двох значень цілих змінних *a* та b.

Задача 3. Дані дійсні числа *x* та *y*. Не використовуючи ніякі інші арифметичні операції крім множення, додавання та віднімання, обчислити

$$1-2x+3x^2-4x^3 \text{ і } 1+2x-3x^2+4x^3.$$

Дозволяється використати не більше восьми операцій.

Емоційна активність студента передбачає його зацікавленість предметом вивчення. Для досягнення успіху процесу навчання необхідно сформувати позитивне емоційне відношення студента до предмету, а саме, потрібно чітко позначити студентам перспективу їхньої професійної діяльності, забезпечити їх необхідними для навчання методичними матеріалами, привчати студентів до підтримки в порядку навчальних приміщень (лекційні аудиторії, комп'ютерні класи). Не менш важливим є формування в студентів позитивного емоційного відношення до викладача, що забезпечується уважним, поважним, доброзичливим, строгим і справедливим відношенням викладача до студентів.

Для реалізації принципу активності викладачу слід навчати студентів ставити питання; контролювати чинники, які відволікають увагу студентів від об'єкту навчання, і усувати причини їх виникнення; вивчати і розвивати індивідуальні інтереси студентів до програмування; підтримувати ініціативу студентів, яка спрямована на досягнення навчальних цілей, а саме, уважно відноситься до гіпотез, версій, варіантів розв’язування поставленої перед студентами задачі.

Принцип самостійності виражається в умінні студента управляти своєю навчальною діяльністю. У зв'язку із цим важливі дії викладача, спрямовані на формування у студентів умінь і навичок самостійної навчально-пізнавальної

діяльності, які полягають в умінні знаходити потрібні відомості в навчальній, науковій літературі, у мережі Internet і використовувати їх для розв’язування завдань; результативно працювати в комп'ютерному класі в час, якій відведено для самостійної роботи.

Принцип індивідуалізації і колективності навчання. Принцип індивідуалізації випливає із необхідності врахування індивідуально-психологічних особливостей кожного студента. Викладач повинен ураховувати індивідуальні особливості студентів, їхні здібності до навчання при виборі темпу навчання, підготовці завдань для самостійного виконання. Це забезпечується спостережливістю, чуйним й поважним відношенням викладача до кожного студента. Викладач повинен знати кого зі студентів варто похвалити за найменший успіх; кому потрібно вчасно зробити зауваження; коли зробити вигляд, що не помітив помилки студента; кому дати індивідуальне навчальне завдання.

Принцип колективності враховує необхідність колективної роботи студентів, з одного боку, для зміцнення колективістських якостей студентів і для посилення виховної дії навчання, з іншого боку, для формування професійних якостей спеціаліста в галузі комп’ютерних наук.

Принцип ефективного поєднання методів навчання. В основі цього принципу знаходиться теза про те, що „процес засвоєння знань, умінь і навичок відбуватиметься найбільш ефективно, якщо в міру можливості в ньому братимуть участь всі органи чуття людини – зір, слух, дотик та ін.” [17]. Набуття студентами практичних навичок розв’язування задач за допомогою комп'ютера в курсі з основ програмування є реалізацією принципу поєднання практичних і наочних методів навчання, що забезпечує досягнення освітніх цілей у підготовці майбутніх інженерів-програмістів.

Принцип міцності і дієвості знань, умінь і навичок. Згідно з цим принципом, викладач повинен, з одного боку, обміркувати зміст предмету з погляду виділення в ньому найважливіших, істотних знань, які повинні бути

обов'язково осмислені і засвоєні студентами і, з іншого боку, сформувати у студентів уміння і навички використання цих знань для розв’язування практичних навчальних задач програмування. Для реалізації цього принципу викладачу слід сприяти розвитку мислення; вчити і привчати студентів користуватися довідниками з інформаційних технологій, мов програмування і т.п.; формувати позитивне відношення до науки програмування; стежити за логікою подання навчального матеріалу; застосовувати технічні засоби, навчальні дискусії, яскраве емоційне подання навчального матеріалу; обмірковувати структуру завдань до кожного заняття; за можливістю підбирати завдання, що мають практичне застосування; при повторенні навчального матеріалу активізувати у студентів не тільки пам'ять, але й мислення (див. 2.3.2.1).

Принцип забезпечення оперативного контролю і самоконтролю в навчанні. Контроль рівня знань, умінь і навичок студентів в процесі навчання основ програмування, необхідний для забезпечення зворотного зв'язку в процесі навчання; отримання відомостей про ступінь ефективності процесу навчання; для здійснення функції освіти, навчання, діагностики, стимулювання, виховання, розвитку; формування у студентів навичок самоконтролю; можливого коректування процесом навчання під впливом одержаних відомостей про хід засвоєння навчального матеріалу.

Для реалізації цього принципу слід використовувати різноманітні форми контролю: усне опитування, тестування, тематичні диктанти, самостійні роботи, письмові контрольні роботи, перевірка виконання домашніх завдань, модульний контроль.

У ході контролю викладачу слід забезпечувати сприятливу психологічну атмосферу, виключити можливість виникнення у студентів стресових станів, які згубно відображаються на їх емоційній сфері і можуть викликати заторможенність розумової діяльності.

2.3.2. Сучасні підходи до навчання основ програмування

2.3.2.1. Задачний підхід у навчанні основ програмування. М.М. Скаткін відзначає, що “сучасна психологія і дидактика розкрили значну роль задач у навчанні як одного з найважливіших чинників збудження і підвищення пізнавальної і практичної активності учнів” [184].

У роботах Є.М. Кабанової-Мєллєр [87], Г.С. Костюка [104] та ін., обґрунтовується концепція формування мислення, яку визначають як „задачний підхід”. Розглядаючи розв’язування задачі як цілеспрямовану діяльність, центральне місце автори відводять розумовій діяльності, в ході якої на основі спеціального формування прийомів розумовій діяльності відбувається розвиток мислення учнів. Початковим моментом розумового процесу, зазвичай, є проблемна ситуація. Мислити людина починає, коли в неї з'являється потреба щось зрозуміти [173].

У програмуванні, як у математиці, фізиці, хімії, біології, уміння і навички розв’язування задач формуються безпосередньо розв’язуванням задач із застосуванням знань теоретичних основ навчального предмету. При цьому, як показує наш досвід роботи, одним з найбільш ефективних методів навчання основ програмування є „задачний підхід”.

У навчально-методичній літературі поняття „задача” із погляду дидактики розглядується як форма втілення навчального матеріалу і засіб навчання, з погляду психології – як мета і спонукання до мислення. Існують різні визначення поняття „задача”. При цьому в більшості визначень задача визначається як структура, що складається з декількох компонент. Так, за І.Я. Лернером: „ознаки будь-якої задачі складаються: 1) з наявності мети розв'язування, що диктується вимогою або питанням до задачі; 2) з необхідності врахування умов і чинників, що є передумовою застосування способу розв'язування і правильності самого рішення; 3) з наявності або необхідності виявлення, побудови способу розв'язування„ [118]. Г.О. Бал визначає завдання як „систему, обов'язковими компонентами якої є: а) предмет

завдання, що знаходиться в початковому стані, б) модель необхідного стану предмету завдання„ [19]. Відомий математик Д. Пойа розрізняє „чотири ступені у процесі розв’язування” задачі: розуміння постановки задачі; складання плану розв’язування задачі; здійснення плану; аналіз результатів ([154; 155]). При цьому, як наголошується в роботі М.І. Кобозєва, „ ... задача повинна бути поставлена коректно в тому сенсі, щоб для її розв’язування був потрібен відбір визначених, обмежених у кількості і принципово доступних даних. При цьому неістотно, чи є ці дані в наявному матеріалі, чи вказується, які з них необхідно одержати” [98]. Н. Вірт визначає, що „У процесі розв'язування будь-якої задачи як за допомогою ЕОМ, так і без її потрібно обрати деяку абстракцію дійсності, тобто визначити безліч даних, що описують реальну ситуацію. Цей вибір залежить від задачи, яку потрібно розв'язати, потім потрібно обрати спосіб подання цієї інформації. ... У більшості випадків ці два етапи взаємозалежні” [35].

Відзначимо, що в програмуванні, як і в математиці, під методом розв’язування задач розуміють сукупність прийомів розумової діяльності або логічних дій і операцій, за допомогою яких розв’язується широкий клас задач, а під способом розв’язування задач розуміють сукупність прийомів розумової діяльності або логічних дій і операцій для розв’язування окремої задачі [186]. Також у програмуванні, як і в математиці, практично кожна задача має декілька способів (алгоритмів) розв’язування. При цьому вважається, що задача розв’язана успішно в тому випадку, якщо знайдено найбільш раціональний спосіб (ефективний алгоритм) її розв’язування. Особливість розв’язування задач з програмування полягає в тому, що:

1. „При розробці програми в першу чергу необхідно зосередитися на характеристиках ситуації, що мають безпосереднє відношення до поставленої задачі, і не звертати увагу на фактори, що вважаються в даному випадку несуттєвими” [68].

2. Розв’язування задачі з програмування, як правило, передбачає реалізацію алгоритму конкретною мовою програмування, що вимагає від програміста, як знання відповідних мовних конструкцій, так і вміння працювати в певному середовищі програмування.

Задачний підхід у процесі навчання основ програмування полягає в тому, що в процесі розв’язування навчальної задачі головним є побудова інформаційної моделі задачі, пошук і побудова алгоритму її розв’язування, а використання, і відповідно, вивчення конструкцій мови програмування є другорядним. Тобто мова програмування є інструментом, за допомогою якого розв’язується задача і який можна поміняти, якщо в цьому виникає необхідність. Зміщення акценту з вивчення синтаксису мови в процесі навчання надає можливість зосередити увагу студентів на розв’язуванні задачі і „знімає бар'єр” для вивчення нових мов програмування, що важливо для інженера-програміста.

Як показує досвід, у процесі навчання основ програмування можна виділити чотири стадії вміння і здатностей розв’язування задач студентами:

- репродуктивне програмування;
- осмислене програмування;
- доказове програмування;
- творче програмування.

Репродуктивне програмування. На стадії репродуктивного програмування студент розв’язує задачі „за аналогією”. Тобто знаходить розв'язану раніше задачу, співставляє умову задачі, яку необхідно розв'язати, з умовою вже розв'язаної задачі з метою використання її алгоритму і тексту програми. Це первинний і найнижчий рівень програмування в процесі навчання. У такій діяльності задіяна пам'ять і, у меншій мірі, процес мислення.

Осмислене програмування. На цій стадії програмування студент здатний самостійно визначити типи даних, необхідних для розв’язування задачі, побудувати алгоритм; усвідомлено використовує конструкції мови

програмування для написання тексту програми; може протестувати програму. Проте, на цій стадії навчання студент бачить найчастіше лише один алгоритм розв’язування задачі, який може бути не завжди найкращим;

Доказове програмування. На цій стадії студент здатний не лише розв'язати поставлену задачу, але і довести правильність роботи алгоритму при всіх допустимих вхідних даних; оцінити час виконання алгоритму, обсяг пам'яті, що потрібен для його реалізації; розглянути декілька алгоритмів розв’язування задачі і обрати з них найбільш раціональній за певними критеріями;

Творче програмування. Ця стадія програмування характерна тим, що студент здатний самостійно поставити задачу; визначити метод і побудувати алгоритм для розв’язування задачі за цим методом; опанувати мову і середовище програмування з метою розв’язування поставленої задачі; використовувати навчальну, наукову, періодичну літературу з програмування в пошуках нових знань, які потрібні для розв’язування задачі. При цьому завдання викладача полягає у всебічній підтримці студента в його прагненні здобувати нові знання і здійснювати пошук.

Досвід роботи показує, що на початковому етапі навчання розв’язуванню нескладних задач у курсі з основ програмування завдання педагога полягає в підтримці будь-якого алгоритму розв’язування задачі, запропонованого студентом, якщо в результаті розв’язування задачі буде одержаний правильний результат. На наступному етапі усі запропоновані алгоритми аналізується студентами разом із викладачем на предмет ефективності з погляду використаних конструкцій мови програмування, кількості операцій, кількості проміжних змінних і т.п. У результаті вибирається найкращий із запропонованих алгоритм.

Такий підхід сприяє підвищенню інтересу студентів до навчання, вселяє в них впевненість у свої можливості і знання, що, у свою чергу, активізує їх пізнавальну активність, стимулює навчальну діяльність студентів. Якщо ж

задача розв’язується за готовим алгоритмом, запропонованим викладачем, її розв’язування не вимагає від студента самостійного творчого мислення. У цьому випадку, у кращому разі, студент на деякий час запам’ятовує розв’язок задачі, але не вчиться мислити, оскільки навички вміння розв'язувати задачі відбувається тільки в результаті самостійної праці.

Відзначимо, що побудова циклу лабораторних робіт із вивченням двох мов програмування можлива у двох варіантах:

1. *Послідовне навчання мов програмування Pascal і C.* Наш досвід показує, що така методика навчання (при тій кількості годин у трисеместровому курсі з основ програмування, яка відведена на цикл лабораторних робіт) цілком можлива.

2. *Паралельне навчання двох мов програмування Pascal і C* із використанням порівняльного підходу ([43], [44], [52]). У цьому випадку використовуються знання мови Pascal, які студенти одержали в шкільному курсі інформатики.

Ми погоджуємося з тим, що нова система позначень, додаткові можливості мови С на деякий час можуть поглинути увагу студентів, але, досвід показує, що поступово приходить розуміння, що за наявності навичок алгоритмізації і структуризації даних кодування алгоритмів іншою мовою, у тому числі і мовою С, не викликає у студентів значних труднощів.

Вибір першого або другого варіанту залежить від рівня підготовки студентів зі шкільного курсу інформатики, а саме, розділу основ алгоритмізації та програмування. Рівень підготовки студентів зі шкільного курсу інформатики визначається за допомогою тестового завдання, яке складається з питань, вправ і задач з основ алгоритмізації та програмування відповідного рівня.

2.3.2.2. Навчання стилю програмування, як складова формування професійної культури майбутнього інженера-програміста. У процесі програмування виникає важлива задача – підтримка єдиного стилю програмування розробниками програмного продукту. Під стилем

програмування розуміють набір прийомів або методів програмування, які використовують програмісти, щоб одержати правильні, ефективні, і зрозумілі програми [198]. Н. Вірт у роботі „Програмування мовою Модула-2” [36] зазначає, що „програма некорисна, якщо людина не може в ній розібратися і не упевнена в її правильності”. У свою чергу, Б. Керніган і Р. Пайк у роботі „Практика програмування” [91] відзначають, що „добре і красиво написаний код простіше читати і сприймати; у ньому, без сумніву, менше помилок, і він напевно буде коротший, ніж код, якій бездумно скомпонований і залишений без поліпшення”. Вони також визначають, що текст програми повинен мати такі властивості:

- очевидна логіка;
- природні вирази;
- використання домовленостей, прийнятих у мові розробки;
- осмислені імена;
- акуратне форматування;
- розгорнені коментарі;
- відсутність хитрих трюків і незвичайних конструкцій.

Програму, яка написана з дотриманням принципів хорошого стилю, простіше читати як самому розробнику, так і іншим програмістам. Тому однією із цілей навчання основ програмування майбутніх програмістів є прищеплення студентам навичок хорошого стилю написання текстів програм.

Наведемо рекомендації стилю написання програм, „які є результатом здорового глузду і випливають з досвіду” [140].

Ідентифікатори. Треба навчати студентів використовувати ідентифікатори, які є інформативними, такими, що запам'ятовуються: сенс усіх імен повинен бути зрозумілий при читанні програми:

```
index, max, min
```

Ідентифікатор, що складається з декількох частин, розділяти символом підкреслення:

```
My_Line, Set_of_Char, X_Start, X_End
```

Не прийнято використовувати як ідентифікатор російські (українські) слова, написані латинськими літерами.

Хоча є мови, які не є чутливими до регістра клавіатури в ідентифікаторах (наприклад, Pascal), все-таки не бажано писати один і той самий ідентифікатор, використовуючи довільну зміну регістрів літер. Наприклад:

```
My_Variable, my_variable, MY_variable
```

Студенти повинні знати про те, що в програмуванні існують деякі домовленості, використання яких полегшує розуміння тексту програми. Наприклад, ідентифікатори `i, j` використовують для позначення лічильників циклів; `n, m, l` – для цілих чисел; `s` і `t` – для рядків; `p` і `q` – для покажчиків; `b, f` – для логічних змінних і констант.

Також студенти повинні знати про так звану угорську систему позначень, у якій ідентифікатор починається з написаного рядковими літерами префікса, що визначає тип об'єкту. Наприклад, `i` – для типу `integer`, `r` – для `real`, `s` – для `string`. Індивідуальна частина ідентифікатора обов'язково починається з великої літери: `rSum, sStr`.

Не можна використовувати як ідентифікатори службові слова.

Рядки в тексті програми не повинні бути довше 80 символів. Це пояснюється тим, що стандартно в текстовому режимі на екрані відображається 80 символів у 25 рядках.

Константи і змінні. Числові константи повинні бути задані за допомогою іменованих констант [91]. Наприклад, якщо в програмі використовується число, яке відповідає кількості місяців у році, то слід описати відповідну константу так:

```
const
  Count_Month=12;  {кількість місяців у році}
```

Це надасть можливість відрізнити значення „Кількість місяців у році” від числа, що визначає, наприклад, порядковий номер місяця.

Згідно з рекомендаціями [140], „у кожному оголошенні повинна оголошуватися тільки одна змінна”. Такий підхід зручний при редагуванні великих програм. На наш погляд, при розв’язуванні невеликих навчальних задач можна однотипні змінні описувати в одному рядку, але при цьому в окремих рядках описувати вхідні дані, проміжні змінні і результати:

```
var
  a, b, c,            {вхідні дані}
  tmp,                {проміжна змінна}
  R        : real;    {результат}
```

Оголошення змінної повинне займати один рядок, у якому вказано ім'я змінної, її тип і коментар із коротким описом змінної:

```
var
  count : byte;  {кількість простих чисел на заданому інтервалі}
```

*Вирази з*аписують так, щоб їх зручно було читати. Щоб покращити читабельність виразу, спростити розуміння його структури або, якщо є сумнів у тому, як обчислюється вираз, треба використовувати дужки, навіть якщо вони не є обов'язковими:

```
2*(Ln(z)/Ln(10))+1/(a*x-b)
```

Бажано вираз повністю розміщувати на одному рядку. Якщо частина виразу переноситься на наступний рядок, то знак бінарної операції слід залишити на першому рядку:

```
Sqrt(Exp(Abs((Sqr(Sin(x))))))+
Sin(x) /Cos(x) -Abs(x*x-Ln(x))
```

У мові Pascal можна відокремлювати бінарні оператори від операндів пропуском, а можна цього не робити:

```
c:= Sqrt(a*a + b*b);
```

або

```
c:=Sqrt(a*a+b*b);
```

Унарні оператори не відокремлюють від операнда:

```
t:=-t;
```

Введення початкових значень. При введенні з клавіатури значень початкових даних разом із процедурою Read (ReadLn) прийнято використовувати процедуру виведення Write (WriteLn). У тексті процедури виведення вказують імена змінних, що вводяться, та їхня послідовність при введенні:

```
Write('Введіть значення змінних a i b : ');
Read(a, b);
```

При розв'язуванні деяких задач привласнити значення вхідним даним можна за допомогою функції, що генерує випадкові числа Random:

```
a:=Random(100);
b:=Random(500);
```

Для ініціалізації випадкових значень використовують процедуру Randomize:

```
Randomize;
a:=Random(100);
b:=Random(500);
```

Для деякого типу задач вхідні дані зручно читати з файлу. Повне ім'я файлу із вказівкою диска, шляху каталогів та імені файлу, у якому зберігаються дані, слід уводити з клавіатури в рядкову змінну. Ім'я цієї змінної повинно бути параметром для процедури Assign:

```
var
  Name : string;
* * *
Write('Вкажіть повне ім'я файлу :');
Read(Name);
Assign(f, Name);
```

Форматування. „Основна теорема форматування свідчить, що візуальне форматування показує логічну структуру програми" [120]. Студенти повинні знати, що розташування операторів у тексті програми повинно відображати структуру алгоритму. Форматування програми виконується відступами. Майже завжди відступ означає два пропуски, рідше – один, три або чотири.

Використовувати знак табуляції для відступу не прийнято, бо він у різних середовищах програмування інтерпретується по-різному і це незручно [91].

Заголовок програми прийнято писати, починаючи з першої позиції рядка. Хоча заголовок програми у всіх версіях середовіща розробки мови програмування Pascal не обов'язковий, краще все-таки, щоб програма мала заголовок. Заголовок розділу вказівки модулів (uses), що підключаються, заголовки розділів описів (`label, const, type, var`), опис процедур і функцій (`Procedure, Function`) і розділ операторів (`begin end`) пишуть, починаючи із третьої позиції рядка. Імена модулів, що підключаються, міток, констант, типів змінних розташовують з п'ятої позиції рядка під відповідним заголовком. Якщо який-небудь із розділів відсутній, то про нього в програмі не згадують.

Імена стандартних процедур і функцій прийнято писати так, як пишуть розробники середовища програмування: `ReadLn, Append, Insert` і т.п. Наприклад:

```
program Example_1;
  uses
    Crt;
  begin
    ClrScr;
    Write('Це оформлення програми');
    ReadKey
  end.
```

Операторні дужки. Що стосується операторних дужок, то існують рекомендації ставити `begin` в тому ж рядку, що і конструкція, яка його використовує (наприклад, цикл, умова або `with`), а end вирівнювати по початку цієї конструкції:

```
  while b<>0 do begin
    tmp:= a mod b;
    a:=b;
   b:=tmp
  end;
```

Таке розташування підкреслює структуру програми і не вимагає зайвих рядків і зайвих зсувів.

Деякі програмісти операторні дужки виносять на наступний рядок відносно конструкцій, що їх використовують [36]. Б. Керніган і Р. Пайк [91] пропонують компромісне рішення. А саме, рекомендують вибрати раз і назавжди один стиль і завжди його дотримуватися.

Існують окремі рекомендації з написання умовного оператора. Слід залишати тільки один пропуск перед і після умови. Якщо частина `then` або `else` складається з простого оператора, то операторні дужки можуть бути пропущені:

```
if a>b then
 max:=a
else
  max:=b;
```

Якщо умова не міститься у рядку, то продовження на наступному рядку повинне починатися у тому ж стовпці, що і перший символ виразу першого рядка:

```
if ((ch>='0') and (ch<='9')) or
   ((ch>='a') and (ch<='z')) or
   ((ch>='A') and (ch<='Z'))
```

Стандартною формою запису багатоваріантного розгалуження є послідовність операторів `if` <умова> `then` <оператор> `else` `if` <умова> `then` <оператор> `else` <оператор>. Це, так звані, вкладені оператори розгалуження:

```
if a[l+1]<=b then
 l:=l+1
else
  if a[r]>=b then
    r:=r-1
  else begin
    tmp:=a[l+1];
    a[l+1]:=a[r];
    a[r]:=tmp
  end;
```

В умові не прийнято писати `if b=True`. Натомість пишуть `if b`. Аналогічно, замість `if b=False` пишуть `if not(b)`.

Не менш важливо правильне форматування текстів програм, у яких використовуються конструкції циклу. Як приклад наведемо фрагмент програми, який важко зрозуміти тому, що погано відформатований:

```
R:=1; m:=0; i:=1;
while i<=n do begin
if A[i]<>0 then R:= R*A[i] else m:=m+1;
i:=i+1
end;
```

Зміна форматування суттєво допомагає розумінню програми:

```
R:=1;
m:=0;
i:=1;
while i<=n do begin
  if A[i]<>0 then
    R:= R*A[i]
  else
    m:=m+1;
  i:=i+1
end;
```

Наведемо приклад форматування фрагмента тексту програми, у якому використовуються вкладені конструкції циклів:

```
for i:=1 to n do
  for j:=1 to m do begin
    Write('Введіть значення A[',i,',',j,']=');
    Read(A[I,j])
  end;
```

Логічні блоки програми розділяють порожніми рядками. По-перше, це покращує читабельність програми. По-друге, наявність порожніх рядків можна використовувати для додавання коментарів.

До питання *про кількість операторів у рядку*. Звичайно, у рядку розміщують один оператор. Наприклад:

```
iSum_P:=0;
iSum_N:=0;
iSum_0:=0;
```

Для полегшення розуміння деяких частин програми можна використовувати додаткові (необов'язкові) операторні дужки.

Треба привчати студентів, майбутніх розробників програмного забезпечення, документувати програму за допомогою к*оментарів.*

Наприклад, рекомендується коментувати оголошення констант і змінних: слід указувати одиниці вимірювання числових величин, діапазони допустимих значень числових величин, обмеження на вхідні дані. Ці коментарі розташовують наприкінці рядка, після вказівки імені і типу змінної або константи:

```
Sq_Meters : real; {кількість тканини в метрах}
```

Коментар повинен не відтворювати словами дію, яка виконується:

```
{ Розділити Salary  N_Day і результат присвоїти Answer}
Answer:=Salary/N_Day;
```

Він повинен описувати призначення дії:

```
{ Обчислення середньої заробітної платні за один день}
Answer:=Salary/N_Day;
```

Коментують також управляючі структури. Якщо це умовний оператор або оператор вибору, то в коментарях пояснюють умову і результати. Якщо це оператор циклу, то вказують його мету. Такі коментарі розташовують у попередніх рядках структури:

{Пошук максимального елементу в масиві і його порядкового номера}

```
max:=a[1];
N_max:=1;
for i:=2 to n do
  if a[i]>max then begin
     max:=a[i];
     N_max:=i
  end;
```

Крім того, коментують завершення управляючої структури. Особливо корисно застосовувати коментарі для позначення кінців довгих циклів і для пояснення їхньої вкладеності. Наприклад:

```
end; {for j}
end; {for i}
end; {while}
```

Також у коментарях описують дію, що виконується процедурою або функцією:

```
Function Count_Num(n: word):byte;
{ Підрахунок кількості цифр натурального числа}
  var
    tmp : byte;
  begin
    tmp:=0;
    while n<>0 do begin
      n:=n div 10;
      tmp:=tmp+1
    end;
    Count_Num:=tmp
  end;
```

Практика показує, що лише 25% часу витрачається на створення програми, інші 75% йдуть на її підтримку – зміну і модернізацію [105].

Головний розробник програмного забезпечення компанії Construx Software С. Макконелл зазначає, що „якщо програміст не піклується про те, щоб його програма була легкою для читання, то це означає, що він просто не розуміє свою програму досить добре” [121]. Досвід підтверджує, якщо студент не дотримується стилю оформлення програми, то це в більшості випадків означає, що він не розуміє зміст і умову задачі, і, як наслідок, не розуміє відповідну програму. У додатку Б наведено текст програми, який задовольняє перерахованим вимогам до оформлення програм.

2.3.2.3 Диференційований підхід як засіб підвищення мотивації студентів при навчанні основ програмування. З метою якомога повнішого розкриття студентами свого потенціалу при навчанні основ програмування доцільним є використання диференційованого підходу. Це пов’язано з тим, що „ … люди розрізняються своїми здібностями, типами пам'яті, стилем сприйняття оточуючого, домінуючим характером мислення і ін. Різні умови, різний життєвий досвід, різний ступінь успішності навчання на попередніх етапах навчання, характер викладання раніше і тепер обумовливають відмінності між учнями. От чому, якщо ми хочемо, щоб навчальний процес у максимально можливій мірі розвивав усіх учнів, то треба при виборі *форм, методів і засобів навчання* здійснювати диференційований підхід” [17].

На нашу думку, одним із засобів реалізації диференційованого підходу в навчанні основ програмування є використання системи навчальних завдань різного рівня складності. Тому нами було розроблено методичний посібник „Початки програмування” із завданнями до лабораторних робіт з основ програмування, побудований за принципами рівневої диференціації (див. п. 1.4.4) [58].

При доборі задач ми орієнтувалися на алгоритмічну складову задач. Алгоритми розв'язування задач збірника можуть бути реалізовані за допомогою будь-якої алгоритмічної мови програмування.

Збірник містить систему завдань до 18 лабораторних робіт. До кожної роботи пропонується близько двадцяти завдань. Завдання до перших семи лабораторних робіт є завданнями-вправами і завданнями-задачами на використання управляючих конструкцій послідовністю дій на рівні операторів: композиція (лінійні алгоритми), розгалуження (алгоритми розгалуження), повторення (циклічні алгоритми). Через велику різноманітність типів задач на використання конструкції повторення в збірнику на цю тему запропоновані завдання до п'ятьох лабораторних робіт. Усі задачі в цих завданнях слід розв'язувати з використанням елементарних (стандартних) типів даних. Для розв'язування задач із наступних завдань потрібні знання і навички, отримані при розв'язуванні попередніх завдань, і знання про структуровані структури даних, такі як лінійні масиви, двовимірні масиви, рядки, множини, записи, файли. В окремих завданнях пропонуються задачі на використання допоміжних і рекурсивних алгоритмів.

Завдання до кожної лабораторної роботи розподілено за трьома рівнями складності. Перший рівень складності – *задовільний* – містить близько десяти задач, другий – *достатній* – шість задач, третій – *високий* – чотири задачі. Таке співвідношення обумовлене часом, який потрібен, на нашу думку, для розв'язування задач: для кожного рівня складності він повинен бути однаковим.

Задачі першого рівня найпростіші. Розв'язування цих задач передбачає засвоєння студентом навчального матеріалу із програмування на задовільному рівні.

Приклад завдання першого рівня складності з теми «Одновимірні масиви»: Написати програму, що розв’язує наступну задачу. Дано натуральне число *n* та масив *x : array[1..n] of integer*. Всі елементи масиву з парними номерами, що розміщені раніше першого по порядку елемента зі значенням $max(x_1, \ldots, x_n)$, помножити на $max(x_1, \ldots, x_n)$.

Задачі другого рівня більш трудомісткі і складні. Ці задачі вимагають від студента вдумливості, уміння аналізувати і їх розв’язування передбачає засвоєння навчального матеріалу на достатньому рівні.

Приклад завдання другого рівня складності з теми «Одновимірні масиви»: Написати програму, що розв’язує наступну задачу. Задано текст з цифр, у кінці тексту стоїть крапка. Вивести цифру, яка зустрічається у тексті найбільшу кількість разів (якщо є різні цифри, що зустрічаються однакову максимальну кількість разів, вивести будь-яку одну з них).

Задачі третього рівня – найскладніші. В основному це задачі, рівень складності яких наближений до олімпіадних задач з програмування. Задачі цього рівня вимагають від студента творчого відношення і зацікавленості процесом програмування, а їх розв’язування забезпечує засвоєння навчального матеріалу на високому рівні.

Приклад завдання третього рівня складності з теми «Одновимірні масиви»: Написати програму, що розв’язує наступну задачу. Дано натуральне число *n* та масив *x : array[1..n] of integer*. Знайти кількість різних чисел серед елементів цього масиву, якщо всі елементи масиву – числа від *1* до *k* і кількість ітерацій має бути порядку $n+k$.

На кожній лабораторній роботі кожний студент сам обирає рівень складності задач, які він буде розв’язувати. Студент одержує максимальну кількість балів за лабораторну роботу, якщо при перевірці домашнього

завдання ним були розв’язані задачі 3-го рівня складності. Якщо студент розв’язав задачі 2-го рівня, то він одержує середню кількість балів. І лабораторна робота вважається зарахованою з мінімальною кількістю балів, якщо студент розв’язав задачі лише першого рівня складності.

При такому підході викладач корегує вибір студентом рівня складності задач в тому випадку, якщо той регулярно вибирає складний для нього рівень задач і не справляється з їх розв’язуванням, тобто у студента завищена самооцінка. Або, навпаки, викладач радить студенту розв’язувати задачі складніші за ті, які він обирає, будучі здатним, але не впевненим у своїх силах.

Проведені нами теоретичні й експериментальні дослідження надають можливість зробити висновок про те, що підвищенню ефективності і якості навчання основ програмування сприяє диференціація навчання, яка реалізується через застосування системи завдань різного рівня складності з урахуванням інтелектуальних здібностей студентів.

Досвід показує, що застосування рівневої диференціації сприяє розвитку у студентів творчого підходу до навчання, прагненню досягти поставлених перед ними цілей, активізує пізнавальну діяльність, підтримує інтерес до навчання у всіх студентів, сприяє розвитку відчуття впевненості у своїх силах, знаннях, здібностях, що має величезне значення для майбутнього фахівця, який буде жити і працювати в умовах жорсткої конкуренції на ринку інтелектуальної праці.

Проведене експериментальне дослідження засвідчили доцільність та ефективність такого підходу до навчання основ програмування, застосування якого також забезпечує індивідуалізацію процесу навчання, надає можливість більш об’єктивно визначити рівень знань кожного студента.

2.3.2.4. Індивідуалізація процесу навчання основ програмування. З метою розв’язування освітніх задач в процесі навчання основ програмування, для забезпечення активної позиції кожного студента ми вважаємо за доцільне враховувати інтелектуальні характеристики (особливості мислення, пам'яті,

сприймання та інших пізнавальних процесів), а також індивідуальні особливості *учня (студента)* як особистості (уважність, тривожність, екстраверсія – інтроверсія, рівень самооцінки та ін.) [146].

Однією із проблем навчання курсу з основ програмування є, на наш погляд, відмінність у стартовому рівні знань з інформатики, зокрема з алгоритмізації і програмування, у студентів першого курсу [63]: від переможців обласних олімпіад з інформатики до осіб, у яких практично відсутні знання, вміння і навички з алгоритмізації і програмування. Зроблені нами кроки (комбінований вступний іспит математика + інформатика, додаткова дисципліна „Вступ до спеціальності"), які були спрямовані на зменшення відмінностей в рівнях знань студентів, не забезпечили повне вирішення даної проблеми. Модульно-рейтингова система навчання, що використовується нами в курсі з основ програмування, тільки підкреслює існуюче положення. Ранжування задач за рівнями складності в межах кожної теми (див. п. 2.3.2.5) надає можливість більш об'єктивно визначити рівень знань студента, проте, повністю не вирішує питання якісного підвищення рівня знань і вмінь студентів.

Ми вважаємо, що для вирішення даної проблеми необхідним є застосування індивідуального підходу до кожного студента від добору індивідуальних завдань до індивідуального темпу навчання, тобто, побудови індивідуальної траєкторії навчання кожного студента для забезпечення кінцевої мети навчання основ програмування.

2.3.3. Методи навчання основ програмування

2.3.3.1. Традиційні методи навчання основ програмування. Відомо, що на вибір методів навчання впливають такі чинники як специфіка дисципліни, цілі навчання, зміст курсу, складність навчального матеріалу, кількість відведеного часу (кредитів) на дисципліну, рівень підготовленості студентів, рівень мотивації навчання студентів, матеріально-технічна база навчального закладу.

Тому в процесі навчання основ програмування доцільно використовувати методи навчання, які обумовлюються видами навчально-пізнавальної діяльності студентів ([15-18], [37], [72], [110], [117], [133], [187-188], [192], [210]).

Інформаційно-рецептивний метод навчання основ програмування застосовується при поданні теоретичного матеріалу дисципліни [18]. Наприклад, при вивченні методів побудови алгоритмів, зокрема методу жадібних алгоритмів, доцільно, з методичної точки зору, при поясненні алгоритму даного методу як перший приклад розглянути „задачу про рюкзак”, яка завжди викликає інтерес у студентів. А потім вже охарактеризувати клас задач, до розв’язування яких доцільно застосувати даний метод, навести інші приклади задач (знаходження найбільшої загальної підпослідовності двох послідовностей і т.п.), які можна розв’язати методом, що вивчається. При використанні цього методу діяльність студентів зводиться до усвідомленого сприйняття і запам'ятовування навчального матеріалу. Інформаційно-рецептивний метод вимагає застосування словесних (розповідь, читання навчальної і наукової літератури), наочних (схеми, блок-схеми, графіки), технічних (кодоскоп, мультимедійний проектор, комп'ютер) засобів навчання.

За допомогою *репродуктивного методу навчання* основ програмування відбувається формування у студентів умінь використовувати здобуті знання. Це здійснюється, як правило, через систему доцільно дібраних завдань. Наприклад, завдань на:

- перевірку знань синтаксису мови програмування;
- перевірку знань семантики мови програмування;
- читання і пояснення результатів операцій;
- використання належним чином управляючих конструкцій при розробці алгоритмів і реалізація цих конструкцій у програмах;
- перевірку вмінь працювати в інтегрованому середовищі мови програмування;

– побудову алгоритму і написання програми за зразком;
– побудову таблиць трасувань;
– використання і адаптацію стандартних алгоритмів;
– читання і пояснення фрагментів програм;
– аналіз алгоритмів (часу виконання, необхідного обсягу пам'яті);
– формування навиків тестування програми і т.п. (див. Табл. 2.2, 2.3).

Діяльність викладача при використанні репродуктивного методу навчання полягає в аналізі виконаного студентом завдання, поясненні та допомозі у виправленні допущених помилок.

Інформаційно-рецептивний і репродуктивний методи забезпечують засвоєння студентами передбачених робочою програмою фактів, методів, алгоритмів, тощо і формування у них вмінь і навичок їх застосування, що є передумовою успішного навчання і творчої діяльності студентів, тоді як методи проблемного навчання – метод проблемного викладу, евристичний метод, дослідницький метод – формують риси творчої діяльності студентів.

Проблемний метод навчання основ програмування полягає в тому, що викладач, поясняючи навчальний матеріал, висуває проблему, допомагає студентам у розв’язуванні цієї проблеми, керує процесом систематизації і закріплення здобутих знань.

Цей метод навчання використовується під час лекцій у формі проблемного викладу навчального матеріалу викладачем, навчальної діяльності студентів під час лабораторних занять, самостійного дослідження навчального завдання і в процесі розробки навчального програмного продукту (див. Табл. 2.5, 2.6).

Проблемне подання навчального матеріалу сприяє активізації процесу сприйняття студентами цього матеріалу, підвищенню їхньої уваги і пізнавальної активності, також творчому засвоєнню знань і способів діяльності.

Евристичний метод навчання основ програмування полягає в тому, що викладач наперед готує систему питань, відповідаючи на які студенти самостійно формулюють визначення поняття, знаходять метод розв’язування поставленої задачі (див. Табл. 2.2, 2.3).

Дослідницький метод навчання основ програмування передбачає самостійний пошук студентами розв’язку поставленої задачі. Наприклад, при вивченні навчального матеріалу змістового модуля «Сортування і порядкові статистики» (див. Додаток А) студенти самі можуть запропонувати деякі алгоритми (методи) сортування.

На активізацію особистих здібностей учнів (студентів) орієнтовані *методи активного навчання*, критеріями ефективності яких є: суб'єктивне відчуття учня (студента) зміни в самому собі, розширення рамок розуміння, формування здатності проектувати власну навчальну діяльність, уміння працювати в колективі [192].

В процесі навчання основ програмування методом активного навчання, який використовується в запропонованій методичній системі, є метод проектів, який передбачає розробку студентами власного навчального програмного продукту (див. п. 2.3.3.3.).

При проведенні лабораторних занять у процесі навчання основ програмування доцільно застосовувати *методи інтерактивного навчання* [189]. Інтерактивна діяльність при цьому передбачає організацію і розвиток діалогового спілкування студентів у процесі спільного розв’язування навчальних завдань. У ході діалогового спілкування студенти вчаться критично й оперативно мислити, зважувати альтернативні думки, виробляти і приймати самостійні продумані рішення, коротко і чітко висловлювати свої думки, спілкуватися один з одним, брати участь у дискусіях. Інтерактив виключає домінування як однієї особи над іншими учасниками дискусії, так і однієї думки над іншою.

Наприклад, при вивченні навчального матеріалу з теми «Символьні рядки і рядкові функції» змістового модулю «Основи мови програмування С» студентам пропонується розв’язати наступну задачу: «Напишіть програму, яка зчитує вхідні дані до тих пір, поки не зустрінеться символ EOF, і обраховує кількість прочитаних слів, прописних і рядкових букв кирилиці, знаків пунктуації та цифр».

Для розв’язування цієї задачі студенти спільно, у діалоговому режимі, з’ясовують як саме відокремити слово у вхідному потоці, означують спосіб визначення прописних, рядкових букв та цифр (за порядковим номером або зазначенням інтервалу відповідних символів у кодової таблиці ASCII), будують алгоритм розв’язування задачі, визначають типи даних і керуючи конструкції, яки потрібно використати. Викладач повинен стежити за цім процесом, вчасно коригувати пропозиції студентів щодо розв’язування задачі, надавати можливість висловити думку кожному студенту.

Досвід показує, що інтерактивні методи навчання сприяють особистісному розвиткові студентів і розумінню вчинків і мотивів поведінки оточуючих, розвиває в них комунікативну компетентність, необхідну для фахівця, що працює в колективі розробників програмного забезпечення.

Важливо відзначити, що при навчанні основ програмування "... величезна роль вчителя, який не стільки розповідає про щось, скільки показує, як він робить те-те й те-те" [35].

2.3.3.2. Метод парного програмування. У процесі навчання основ програмування доцільно використовувати метод «парного програмування», суть якого полягає в тому, що під час виконання завдань на лабораторних заняттях, при роботі над завданнями для самостійного виконання два студента одночасно працюють за одним комп'ютером. При цьому один із студентів набирає текст програми, а другий контролює або коригує його роботу. В цьому випадку студенти, що працюють в парі, постійно спілкуються з приводу

навчального завдання, здійснюють взаємонавчання і взаємоконтроль. Як показує практика, при парному програмуванні студенти швидше виконують поставлене перед ними завдання, що пояснюється їх підвищеною пізнавальною активністю. Крім того, парне програмування сприяє формуванню навичок та вмінь працювати в колективі розробників.

При парному програмуванні обов'язковою є умова обміну ролями студентів в процесі роботи, тобто періодично студенти, що працюють в парі, повинні мінятися місцями за робочим місцем (комп'ютером).

Даний метод є також і формою організації діяльності студентів, за якої майбутні програмісти показують більш, ніж у двічі більшу продуктивність, у порівнянні з тим, коли вони працюють поодинці [180].

2.3.3.3. Колективний проект як метод активного навчання основ програмування. Сучасне програмування є колективним, і корисність окремого програміста тісно пов'язана з його корисністю для всієї команди. Науковці, викладачі і практики розуміють необхідність формування і розвитку у майбутніх програмістів таких важливих якостей як уміння працювати в колективі і вміння працювати із замовником ([28], [53], [54], [69], [121], [174]).

Для подолання протиріччя між абстрактним предметом навчально-пізнавальної діяльності студентів і їх майбутньої професійною діяльністю у навчальному процесі необхідно організувати таке навчання, що забезпечує перехід, трансформацію одного типу діяльності (пізнавального) в іншій (професійний) з відповідною зміною потреб і мотивів, цілей дій, засобів, предметів і результатів [34].

Створення програмного продукту тісно пов'язано з поняттями проект і команда. Програмний продукт – „це не тільки програми, але і вся супутня документація, а також конфігураційні дані, необхідні для коректної роботи програм" [191].

Сучасна система вищої освіти повинна надати майбутньому інженеру-програмісту можливість отримати досвід колективної роботи над створенням програмного продукту. На нашу думку, перші кроки до набуття досвіду студентами колективної розробки навчального програмного продукту – проекту – доцільно робити вже на молодших курсах, зокрема, при навчанні основ програмування [55].

Для цього, на початку останнього семестру навчання основ програмування, студентам пропонується об'єднатися в мікрогрупи по 2-3 особи для створення власного навчального програмного продукту і продемонструвати свою розробку після завершення вивчення дисципліни. У зв’язку з тим, що програмний продукт створюється колективом розробників, визначаються функції кожного з розробників проекту. Кожний з учасників повинен виконати свою частину роботи над програмним продуктом, за яку несе відповідальність перед групою. Усі рішення щодо проекту повинні прийматися з відома всіх учасників групи. Завершений програмний продукт усі групи повинні подати і захистити прилюдно на підсумковому занятті курсу з основ програмування.

Змістова частина проектів повинна мати творчу спрямованість. Допускаються і заохочуються завдання, сформульовані самими студентами. Викладач, оцінюючи рівень складності проекту, може тільки коректувати цей вибір, при цьому бажано щоб коректування носило виключно рекомендаційний характер. Зауважимо, що наш досвід роботи показує, що через вікові особливості, майже кожна група, за рідкісним виключенням, як проект реалізовує власну версію якої-небудь гри.

Кожен проект повинен відповідати основним показникам якісного програмного забезпечення:

– 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.

Тексти програм проекту повинні бути написані мовами Pascal або C у відповідних інтегрованих середовищах програмування.

Кожен проект повинен спиратися на методи побудови алгоритмів, що вивчаються, і передбачати розробку власного алгоритму. Треба, щоб розробка інтерфейсу користувача вимагала від студентів самостійного вивчення роботи із графікою, організації взаємодії з периферійними пристроями на низькому рівні.

Зауважимо, що такий вид навчальної діяльності змінює традиційні ролі студента і викладача. При створенні колективного проекту студенти вільні у виборі мовних засобів, у розподілі обов'язків кожного члена групи, у виборі темпу роботи над проектом. Студентам надається можливість самостійно встановлювати послідовність засвоєння нових знань для реалізації своїх ідей. Викладач тільки корегує роботу студентів, спрямовуючи їх зусилля. Причому дії викладача носять рекомендаційний характер. Крім того, для надання консультації і контролю над процесом роботи викладачеві необхідно періодично призначати консультації групам розробників. Така співпраця взаємовигідна: студенти вимушені регулярно працювати над проектом, викладач може відстежувати процес розроблення програмного продукту.

Демонстрація проектів та їхні захисти повинні відбуватися публічно, тобто в присутності всіх студентів групи. Важливо, щоб студенти кожної мікрогрупи мали можливість порівняти власну розробку з розробками інших мікрогруп.

Оцінювання якості проекту і внесок кожного студента (розробника) у проект здійснює комісія. До складу комісії, окрім викладачів, що читають курс, доцільно включати студентів, що користуються авторитетом у своїх колег. Члени комісії повинні оцінити проект у цілому:

– алгоритмічну складність;

– завершеність проекту;

– об'єм виконаної роботи;

– якість призначеного для користувача інтерфейсу;

– супровідну документацію.

Крім того, комісія визначає підсумковий бал кожного з учасників колективної розробки на основі виконаної частини проекту.

Враховуючи важливість і складність такого виду навчальної діяльності, пропонується відвести на цей вид роботи 25% балів за 100-бальною шкалою, у модульно-рейтинговій системі оцінювання навчальної діяльності студентів.

Оцінка розробленого студентами програмного продукту здійснюється у 2 етапи. На першому етапі проект оцінюється у цілому. Тобто з’ясовується, яке завдання було поставлено мікрогрупі, відповідність розробленого проекту завданню і рівень його завершеності. Максимальна кількість балів, яку можуть отримати розробники проекту на першому етапі (у випадку відповідності проекту усім вимогам до нього), дорівнює 15-ти балам. Ця кількість балів нараховується кожному учаснику розробки.

На другому етапі з’ясовується дольова участь кожного учасника у розробки проекту. Як правило, кожний з учасників розробки відповідає за окрему (одну або дві) частину проекту: розробка технічного завдання, складання алгоритму, написання коду програм, відображення графічних об’єктів, забезпечення інтерфейсу користувача, підготовка супровідної документації програмного продукту тощо. Кожен з цих видів роботи оцінюється від 1 до 5 балів. На цьому етапі кожен з учасників розробки може отримати від 1 до 10 балів, в залежності від виконаної частини проекту.

Кожен член комісії має оцінити у балах кожен проект і участь кожного розробника проекту.

Після перегляду усіх розробок підраховуються загальна кількість балів кожного студента за проект.

Підсумкові бали студента складаються з кількості балів, отриманих за проект на першому етапі, плюс середнє арифметичне балів, що булі виставлені персонально учаснику розробки усіма членами комісії на другому етапі. Якщо сума балів, яку отримав студент, перевищує 25 балів, то студенту все одно зараховується 25 балів.

Участь студентів у роботі комісії такого роду, надає можливість формувати в них уміння оцінювати трудовитрати на створення програмного продукту, розвивати критичне мислення, що, у свою чергу, необхідно в професійній діяльності інженера-програміста при плануванні і реалізації широкомасштабної розробки.

Досвід роботи показує, що у студентів, завдяки завданню на розробку програмного продукту, значно зростає інтерес до лекційного курсу, і відповідно, зростає рівень засвоєння теоретичного матеріалу. Студенти на лекціях задають питання, відповіді на які їх цікавлять: про методи побудови ефективних алгоритмів, про вибір найбільш відповідних структур даних для розв’язування задач, про типові алгоритми, про можливості обчислювальної системи, про різні алгоритми розв’язування однієї й тієї ж задачі – швидких або економних із погляду об’єму оперативної пам'яті**.**

Можна стверджувати, що розробка колективного проекту надає можливість підвищити пізнавальну активність студентів, систематизувати і поглибити теоретичні знання, одержані в курсі з основ програмування.

Такий вид навчальної діяльності сприяє розвитку у студентів наступних умінь і навичок:

- уміння формулювати і висловлювати свої думки та ідеї;
- уміння здійснювати презентацію, як у формальній, так і в неформальній обставинах;
- навички роботи з навчальною, науковою, спеціальною літературою, періодичними виданнями;
- розуміння і конструктивне обговорення виступу інших;

- складання технічної документації (опис структури проекту, алгоритмів).

А також виховує такі якості як:

- розуміння відповідальності за виконання своєї частини проекту;
- спілкування з колегами з проблем майбутньої професійної діяльності;
- уміння працювати в колективі.

Однією з найважливіших характеристик методу активного навчання – створення програмного продукту – є самостійна групова діяльність, яку студенти виконують протягом певного періоду.

Крім того, такий вид навчальної діяльності спонукає студентів до самостійної творчої роботи; глибокого і усвідомленого вивчення курсу програмування, який є базовим у підготовці інженерів-програмістів; для досягнення загальної мети в процесі спільної діяльності [186].

Отже, перехід від розв’язування навчальних задач з програмування до створення закінченого, об'ємного, самостійно розробленого програмного продукту з розвиненим інтерфейсом, з одного боку, і усвідомлення своїх можливостей, з іншого боку, є для студентів сильним стимулом до подальшого навчання. Крім того, можливість роботи в колективі над програмним проектом на ранніх етапах навчання формує у студентів уявлення про майбутню професійну діяльність.

У додатку В наведено загальні положення щодо завдання на розробку студентами навчального програмного продукту, а в додатку Г – текст програм та вихідні форми навчального програмного продукту «Автомобільні шляхи України», розробленого студентами.

2.3.3.4. Метод паралельного навчання мов програмування. Проблемам навчання мов програмування присвячені роботи В.М. Бондарьова, В.І. Рубенецького, Є.Г. Качка [27], Г. Буча [32], В. Вейтмана [33], Н. Вірта [36], Р. Грехема [47], Г. Григаса [48], В. Кассера і Ф. Кассера [89], В.М. Заварикіна,

В.Г. Житомирського, М.П. Лапчика [77], М.І. Жалдака, Ю.С. Рамського [75], Б. Кернігана, Д. Рітчі [91], Е.Б. Коффмана [105], Г.В. Лебедєва, Г.В. Кушніренка [114], О.В. Крячкова, І.В. Сухініна, В.К. Томшина [106], О.І. Марченка і Л.О. Марченка [122], О.В. Мілова [129], Н.В. Морзе [135], С.М. Окулова [141], Т.А. Пауелла [150], С. Прата [157, 158], Т. Пратта і М. Зелковіца [159], В.С. Проценка, П.П. Чаленка, А.Б. Ставровського [168, 194], Д.Г. Румянцева, Л.Ф. Монастирського [174], Б. Страуструпа [195], О.Ф. Ускової та ін. [166], А. Фьюера [215], Е. Хювєнена, І. Сеппянена [217, 218], А. Шеня [220], Г. Шилдта [221] та ін.

М. Бен-Арі [21], Т. Пратт та М. Зелковіц [159], Р. Себеста [177] та ін. розглядають у своїх роботах загальні концепції, синтаксис і семантику мов програмування, дають їхні порівняльні характеристики.

У Черкаському національному університеті імені Богдана Хмельницького проводився педагогічний експеримент з впровадження в навчальний процес створеної автором методичної системи навчання основ програмування. Під час експерименту перевірялась, зокрема, наступна гіпотеза: паралельне навчання двох мов програмування на основі їх порівняння створює умови для інтенсифікації процесу навчання, активізації навчально-пізнавальної діяльності студентів, сприяє розвитку основних мислительних операцій у студентів і формуванню в них узагальнених знань з програмування.

Педагогічний експеримент проводився на протязі одного навчального семестру. В цей час студенти були ознайомлені з синтаксисом і семантикою керуючих конструкцій, типами даних, стандартними процедурами та функціями введення-виведення мов програмування Pascal і C. При цьому зверталася увага студентів на подібності та розбіжності в конструкціях цих мов. У процесі розв’язування задач, на етапі написання програми, викладач наводив приклади програм, написаних на обох мовах програмування. Студентам пропонувалося також самостійно виконувати програмну реалізацію алгоритмів розв’язування задач мовами програмування Pascal і C.

За результатами експерименту можна зробити деякі висновки.

1. У порівнянні з традиційним, послідовним, підходом до навчання основ програмування за схемою Pascal → C → C++, більше часу витрачалося на вивчення синтаксису мов програмування, а під час експерименту цей час вдалося скоротити.

2. Для більшості студентів стало зрозумілим, що головним завданням при розв'язуванні задач з програмування є алгоритмічна складова.

3. Існуючий у студентів експериментальної групи розрив у знаннях з основ програмування істотно збільшився.

Цікавими є думки студентів із цього приводу. Після закінчення першого семестру вивчення курсу „Основи програмування та алгоритмічні мови" було проведено анкетування експериментальної групи студентів першого курсу спеціальності „Програмне забезпечення автоматизованих систем" (див. додаток Д).

Після опрацювання відповідей на запитання анкети отримані наступні результати:

- 77% студентів вивчали мову програмування Pascal у школі;
- 50% студентів до вступу в університет додатково вивчали інформатику, і, зокрема, програмування;
- 46% опитаних студентів вважають доцільним вивчення мови Pascal в університеті, з них, 13% вважають так тому, що не вивчали цю мову в школі й 33% тому, що Pascal більш зрозумілий, ніж C;
- 10% студентів вважають, що знання мови C потрібні, як основа для вивчення мов програмування C++, C#, Java;
- 100% студентів висловлюється за вивчення декількох мов програмування;
- жоден студент не сказав, що йому не подобається мова C (0%).

На екзамені з дисципліни „Основи програмування та алгоритмічні мови” після першого семестру навчання для розв’язування запропонованих задач мову програмування Pascal обрали 79%, з них:

- 13% студентів тому, що ця мова відома їм зі шкільного курсу інформатики;
- 23% - тому, що вона більше подобається;
- 43% - за звичкою;
- 21% - відповіли, що для них немає різниці, якою мовою писати програму.

Решта студентів (21%) розв’язували задачі за допомогою мови С, при цьому робили це з наступних причин:

- 6% - більше подобається з мови С;
- 3% - приємно усвідомлювати, що можуть писати програми мовою С;
- 3% - немає різниці, якою мовою писати програму, бо однаково володіють мовою Pascal и С;
- 3% - хотіли продемонструвати свої знання з мови С;
- 6% - хотіли перевірити свої знання з мови С.

Після підведення підсумків анкетування на формуючому етапі педагогічного експерименту з’ясувалось, що 81% студентів вважають доцільним паралельне вивчення мов Pascal і С на основі їх порівняння, при цьому 55% студентів вважають, що систематизуються знання з мови Pascal і легше, на їхню думку, вивчати мову С, а 42% студентів вважають, що це надає можливість робити порівняльний аналіз мов програмування; 13% респондентів вважають недоцільним паралельне вивчення мов Pascal і С. Для цієї категорії студентів послідовне вивчення мов програмування простіше; 6% - не вважають за потрібне вивчення мови Pascal у ВНЗ тому, що вивчали її в школі.

На основі опитувань і тестування студентів за матеріалами лекційного курсу, аналізу їх роботи на лабораторних заняттях, результатів проведеного екзамену, а також анкетування студентів, можна зробити попередній висновок

про те, що за певних умов (використання різноманітних методів навчання, диференційованого підходу у навчанні, рейтингової системи навчання) паралельне вивчення мов Pascal і C на основі їх порівняння може сприяти інтенсифікації процесу навчання, активізації пізнавальної діяльності студентів, розвитку в них основних розумових операцій і формуванню узагальнених знань з програмування. Але істотною умовою успішного навчання основ програмування на першому курсі з використанням методу паралельного навчання мов програмування Pascal і C є, по-перше, наявність у студентів знань основ алгоритмізації і програмування в межах шкільного курсу інформатики, по-друге, рівень цих знань у студентів повинен бути приблизно однаковим.

2.4. Засоби навчання основ програмування

Під засобами навчання розуміють підручники, навчальні посібники, науково-популярну та довідкову літературу, журнали, дидактичні матеріали, методичні вказівки, наочні посібники і технічні засоби навчання, моделі, прилади, інструменти (креслярські і вимірювальні) [187]. Невід'ємною частиною засобів навчання є комп’ютери і педагогічні програмні засоби.

У цілому, засоби навчання повинні складати єдиний комплекс разом з іншими компонентами методичної системи, що забезпечує досягнення освітньої мети.

Розглянемо засоби навчання основ програмування.

2.4.1. Аналіз підручників, посібників і збірників задач з основ програмування. Підручник є одним з основних засобів навчання [34]. К.Д. Ушинський називав підручник „фундаментом гарного навчання”. У методичних рекомендаціях щодо структури, змісту та обсягів підручників і навчальних посібників для вищих навчальних закладів зазначено, що „підручник – навчальне видання, що містить систематизоване викладення

навчальної дисципліни, відповідає програмі дисципліни та офіційно затверджене як такий вид видання” [127].

Ми вважаємо, що зміст підручника, як взагалі, так і з основ програмування, повинен задовольняти деяким вимогам, зокрема: тлумачення понять, термінологія і символіка, що використовується в підручнику, повинні відповідати прийнятим у програмуванні; подання навчального матеріалу повинно бути науковим і доступним; у теоретичному матеріалі підручника повинно розкриватися прикладне призначення програмування; система завдань повинна розкривати внутрішні предметні та міжпредметні зв'язки. Крім того, у підручниках з програмування повинно бути приділено увагу формуванню наукового світогляду студентів, професійних якостей майбутніх програмістів.

Дамо стислий аналіз деяких підручників, посібників і збірників задач з основ програмування за такими критеріями, як зміст, мова програмування, яка пропонується до вивчення, наявності питань, тестів для самоконтролю, завдань на програмування і т.п.

Підручник А.Б. Ставровського [194], написаний за матеріалами початкового курсу програмування, мета якого – „дати основні поняття програмування, прищепити початковий досвід самостійної розробки програм і познайомити з деякими класичними задачами й алгоритмами”. Підручник складається з трьох частин. У першій частині викладаються початкові відомості про комп'ютер, мову Pascal, про побудову програм і про метамову. Друга частина присвячена організації й обробленню структурованих типів даних, методам роботи з масивами, записами, файлами і списками зв'язаного збереження. У третій частині дається поняття складності алгоритму. А також розглянуті задачі сортування масивів і файлів, перебір варіантів, синтаксичний аналіз і обчислення виразів. Кожна частина поділена на глави, що, у свою чергу, розбиті на теми. Для закріплення теоретичного матеріалу з кожної теми автор розглядає задачі і пропонує пояснення до них. Є в підручнику задачі для самостійного виконання, але їх недостатньо і нема розподілу цих задач за

рівнями складності. До деяких з них пропонуються розв’язки або вказівки до розв’язування. Підручник виданий російською та українською мовами.

У підручнику українських авторів В.П. Проценка, П.П. Чаленка, А.Б. Ставровського [168] „викладені основні засоби та методи конструювання ефективних програм сучасною мовою програмування С”. Наведені алгоритми і програми оброблення таких структур як лінійні списки, матриці, багатовимірні масиви, дерева, графи, файли. Цей підручник позиціонується авторами як підручник для студентів ВНЗ, що навчаються за спеціальностями „Прикладна математика” та „Програмне забезпечення автоматизованих систем”. Підручник поділений на 6 розділів, кожний з яких містить кілька параграфів. У кожному параграфі пропонуються завдання для самостійної роботи. Теми, в яких розглядаються абстрактні типи даних, добре ілюстровані, що має важливе значення при поданні навчального матеріалу такого рівня складності. На жаль, у текстах програм цього підручника досить часто зустрічаються помилки.

Слід зазначити, що в даний час з'явилося досить багато гарних підручників з алгоритмізації і програмування, які видані за кордоном і перекладені на російську мову: Т. Пратт, М. Зелковіц [159]; Р. Себеста [177]; Т. Кормен, Ч. Лейзерсон, Р. Рівест [100]; Дж. Макконелл [120]; А.В. Ахо, Д.Е. Хопркрофт, Дж. Ульман [13;14]; С. Гудман, С. Хідетніємі [65]; Елліот Б. Коффман [105]; Н. Вірт [35]; Б. Керніган, Д. Рітчі [90]; А. Фьюер [215]; Б.В. Керніган, Р. Пайк [91]; С. Прата [157] та ін. На жаль, на українську мову ці підручники не перекладені.

Підручники, які перекладені з англійської мови на російську мову, орієнтовані на певні навчальні дисципліни, відмінні від курсу з основ програмування. Зрозуміло, що це пояснюється різними навчальними планами ВНЗ різних країн. Але ця обставина не зменшує значимості цих підручників. Наприклад, деякі розділи підручника викладачів Массачусетського технологічного інституту Т. Кормена, Ч. Лейзерсона, Р. Ривеста [100] можуть

бути частиною курсу з основ програмування. Характерним для цього підручника є те, що фрагменти програм і програми представлені на псевдокоді.

У додатку Е наведено аналіз *підручників* (таблиця Е.1) *і навчальних посібників* (таблиця Е.2) *з основ програмування* з погляду змісту, мови програмування, яка пропонується до вивчення, наявності порад з програмування, питань, тестів для самоконтролю, завдань-вправ та завдань-задач на програмування, розподілу завдань за рівнями складності; кількості завдань із теми (достатньої/недостатньої для засвоєння навчального матеріалу), відповідей на запитання та розв'язків завдань.

На жаль, якісних підручників з основ програмування, зокрема, українською мовою, недостатньо. А з тих, які є, не всі задовольняють визначеним у дослідженні вимогам організації навчального процесу. Аналіз показує, що більша частина підручників з основ програмування російських і українських авторів орієнтована на вивчення мов програмування Pascal або C. При цьому підручників з навчання мови програмування C значно менше. Слід зазначити, що у цих підручниках недостатньо уваги приділяється питанням подання даних у пам'яті комп'ютера, методології програмування, питанням компіляції та інтерпретації програм, формам подання алгоритмів, аналізу алгоритмів, методам побудови алгоритмів.

Укладанням збірників завдань і вправ з програмування з різних тем і рівнем складності займались С.О. Абрамов, Г.Г. Гнезділова, О.М. Капустіна, М.І. Селюн [78], Д.О. Гуденко, Д.В. Петроченко [176], О.П. Зеленяк [80], В.С. Зубов [81], Т.П. Караванова [86], В.М. Пільщіков [153], А.Б. Ставровській [194], А.Г. Юркін [225], Ч. Уезерелл [209], О.М. Бухтіяров, Г.Д. Фролов [31], Г. Уоррен [207] та ін.

Математичним основам алгоритмізації, методам побудови алгоритмів і аналізу складності алгоритмів присвятили свої роботи А.В. Ахо, Д.Е. Хопркрофт, Дж. Ульман [13, 14], Дж. Бентлі [22], С.Б. Гашков, В.М. Чубаріков [45], Д. Грін, Д. Кнут [49, 97-99], Д. Гріс [50], Е. Дейкстра

[69, 70], С. Гудман, С. Хідетніємі [65], Б.М. Іванов [82], Б.В. Керніган, Р. Пайк [91], Т. Кормен, Ч. Лейзерсон, Р. Рівест [100], С.С. Лавров [113], Д. Макконелл [120], Б. Мейєр, К. Бодуєн [125, 126], Г.С. Уоррен [207] та ін.

Збірник задач С.О. Абрамова, Г.Г. Гнєзділової, О.М. Капустіної, М.І. Селюн [78] містить більше тисячі задач. Формулювання задач у цьому збірнику універсальні в тому розумінні, що для написання програм можуть використовуватися різні мови програмування. Діапазон складності задач досить широкий. Збірник складається з двох розділів: „Основні прийоми програмування" і „Задачі з тем".

Цей збірник задач є базовим для багатьох інших задачників з програмування: В.М. Бондарєв, В.І. Рублінецький, О.Г. Качко [27]; Т.П. Караванова [86]; А.В. Крячков, І.В. Сухініна, В.К. Томшин [106]; О.В. Мілов [129]; Д.А. Гуденко, Д.В. Петроченко [176]; А.Г. Юркін [225].

У таблиці Е.3 додатку Е наведено перелік і аналіз деяких *збірників задач і вправ з основ програмування*, з погляду їх змісту, мови програмування, на яку орієнтовано завдання у збірнику, наявності завдань-вправ і завдань-задач, розподілу завдань за рівнями складності, кількості завдань і відповідей до них.

Стрімкий розвиток комп'ютерних наук у цілому, і програмування, зокрема, вимагає від програміста постійного оновлення і поповнення знань. Тому необхідно створити умови, які будуть стимулювати студентів використовувати в навчальній діяльності *періодичну літературу,* електронні джерела в мережі Internet, що, в свою чергу, повинно забезпечити розширення їх світогляду в сфері комп'ютерних наук, комп'ютерної техніки та інформаційно-комунікаційних технологій (див. п. 2.4.2). Сюди можна віднести журнали „Программирование +", „Компьютерное обозрение", „Chip", „Информатика и программирование", сайти в мережі Internet (http://ko-online.com.ua, http://www.maik.ru/cgi-bin та ін.) та ін.

2.4.2. Система завдань з основ програмування як засіб підвищення практичної значущості результатів навчання. Однією з головних практичних цілей навчання майбутніх інженерів-програмістів є формування вміння складати ефективні алгоритми для розв’язування реальних задач, з погляду часу їх виконання, пам'яті, що при цьому використовується, а також реалізація цих алгоритмів однією з мов програмування [60].

У програмуванні під розв’язуванням задачі у загальному випадку розуміють процес створення програмного продукту. Цей процес включає наступні етапи: „визначення проблеми; визначення вимог; створення плану конструювання; розроблення архітектури програмного забезпечення або високорівневе проектування; детальне проектування; кодування і налагодження; блочне тестування; інтеграційне тестування; інтеграція; тестування системи; супровід” [121]. Очевидно, що в процесі навчання майбутній інженер-програміст повинен набути досвіду виконання цих видів робіт. Тому в дисциплінах циклу комп'ютерних наук повинні бути передбачені види і типи завдань, за допомогою яких будуть сформовані відповідні вміння і навички. Це буде сприяти покращенню підготовки майбутніх програмістів через формування цілісної картини щодо етапів розробки і супроводу програмних продуктів.

З огляду на всю розмаїтість завдань – від постановки задачі до інтерпретації результатів її розв’язування – і специфіку програмування, ми вважаємо, що в курсі з основ програмування повинні бути розроблені і запропоновані студентам для розв’язування різні типи навчальних завдань, за допомогою яких будуть формуватися відповідні вміння і навички програмування. При цьому, для забезпечення високої якості результатів навчання основам програмування, доцільно розробити класифікацію навчальних завдань; визначити призначення кожного типу завдань та необхідну кількість завдань у залежності від важливості, складності, часу і трудомісткості їх розв’язування.

Огляд існуючих навчальних посібників і збірників задач з програмування показує, що одна з найважливіших проблем методичного забезпечення навчання програмуванню – формування науково обґрунтованої системи завдань із програмування – у даний час залишається невирішеною, а в педагогічній практиці розподіл завдань на „вправи" і „задачі" відбувається, як правило, інтуїтивно.

Побудові системи вправ з імперативного програмування у фундаментальній підготовці майбутніх вчителів інформатики присвячена дисертаційна робота Ф.В. Толкачова [199]. У цій роботі поняття „задача з інформатики" визначається як сукупність мети, суб'єкта та його діяльності, де суб'єктом є той, кого навчають (студент), а до діяльності суб'єкта включено обчислювальний експеримент – технологію розв'язування задач за допомогою обчислювальної системи, а поняття „вправа з програмування" визначається як задача (або вправа) для того, хто навчається, що написана тим, хто вчить, деякою формальною мовою або природньо-математичною мовою з чітко виділеними передумовою та післяумовою. Також в [199] запропонована класифікація вправ на основі виділення конкретної навчальної діяльності. Як показало наше дослідження, дисертаційні роботи з методики навчання інформатики, зроблені в Україні, в яких би розглядалися питання класифікації і побудови системи навчальних завдань із програмування, у тому числі і для навчання майбутніх інженерів-програмістів, практично відсутні.

У педагогіці існують різні підходи до класифікації навчальних завдань. Існують класифікації, що орієнтуються на діяльність учня ([51], [199]); діяльність учителя [79]; зміст і структуру матеріалу, що вивчається [203]; структурно-компонентний склад завдань [30]. Ми вважаємо, що класифікація навчальних завдань з основ програмування повинна бути орієнтована на вид діяльності того, хто навчається, (вибір структур даних, побудова алгоритму, оцінка ефективності програми, вивчення мови програмування тощо) і на характер його діяльності (репродуктивний, репродуктивно-пошуковий,

творчий). Орієнтація на вид і характер діяльності студента (учня) при побудові класифікації навчальних завдань дає можливість вирішити питання щодо впорядкування завдань у процесі навчання основ програмування.

Навчальним завданням з основ програмування будемо вважати питання, вправи різного характеру, текстові задачі, що містять мету, умову і вимоги до очікуваного результату.

Навчальні завдання з основ програмування, на наш погляд, призначені для забезпечення:

– засвоєння змісту навчального матеріалу курсу;

– загальновизнаних рівнів засвоєння знань і вмінь [72], а саме, а) засвоєння знань і способів діяльності на рівні сприйняття, усвідомлення і запам'ятовування; б) засвоєння їх на рівні застосування за зразком, у знайомій ситуації; в) засвоєння їх на рівні творчого застосування;

– розвитку в студентів таких якостей, як інтелектуальна активність, упевненість у своїх знаннях, виробленню в них уміння оперувати здобутими знаннями;

– оцінювання значущості результатів навчальної діяльності;

– формування позитивної мотивації навчання, потреби до самоосвіти, навичок контролю і самоконтролю, умінь і навичок колективної роботи над одним спільним завданням.

Традиційна система навчання основ програмування передбачає використання в якості навчальних завдань розв’язування задач, згідно заданої умови (С.О. Абрамов і ін. [78], Б. Керніган, Д. Рітчі [90], О.Ф. Ускова [166], А.Г. Юркін [225] і ін.).

Ми вважаємо, що система навчальних завдань з основ програмування повинна містити завдання, які сприятимуть підготовці студентів до створення великих програмних продуктів у колективі розробників. Це завдання на тестування і налагодження програм, читання і розуміння готових програм, документування програм, дослідження предметної галузі і т.п. Ці вміння і

навички, очевидно, необхідно формувати після того, як студентами здобуті певні знання та сформовані початкові вміння та навички програмування. У свою чергу, на початковому етапі навчання основ програмування необхідні завдання, при виконанні яких формуються вміння і навички працювати в середовищі розроблення програмних продуктів, знання синтаксису і семантики мови програмування і т.п. Таким чином, коло завдань з основ програмування повинно бути значно розширено: від невеликих вправ на запис математичних виразів у прийнятій в мові програмування формі до розробки цілісного програмного продукту. Також необхідно визначити послідовність виконання різних типів завдань. При цьому важливо, щоб домінуючими були завдання, що формують алгоритмічний стиль мислення ([73], [97], [99]).

У процесі навчання кожне лабораторне заняття або завдання для самостійного виконання повинно містити різноманітні типи завдань, що якнайповніше розкривають зміст теми, яка вивчається, і відповідають індивідуальним здібностям студентів. Час, якій потрібен на виконання завдань, і рівень складність завдань визначають кількість завдань, які повинен виконати студент на лабораторній роботі та в поза аудиторний час.

Також у процес навчання слід включати і завдання, на виконання яких, крім раніше здобутих знань, сформованих умінь і навичок із програмування, потрібен тривалий час (див. табл. 2.6).

Розподіляємо навчальні завдання з основ програмування на:

– завдання-*вправи*, що спрямовані на забезпечення засвоєння навчального матеріалу і формування способів діяльності на рівні сприйняття, розуміння і запам'ятовування;

– завдання-*задачі*, що спрямовані на забезпечення засвоєння навчального матеріалу і способів діяльності на рівні застосування за зразком, у знайомій ситуації;

– завдання *на складання тестових питань з певної теми навчального матеріалу*, що спрямовані на якісне вивчення навчального матеріалу курсу,

формування таких якостей, як лаконічність і точність висловлювання думок, відповідальність за результати праці;

– завдання-*дослідження*, що спрямовані на формування у студентів навичок роботи з навчальною, науковою літературою і з періодичними виданнями з метою пошуку необхідних відомостей для виконання навчальних завдань, оцінювання значущості знайдених відомостей і т.п.;

– завдання *з розробки програмного продукту*, що спрямовані на формування у студентів навичок і досвіду розробки реальних програмних продуктів із використанням технологій проектування, а також розвиток творчих здібностей студентів, при цьому перевага повинна надаватися колективним розробкам.

Завдання-вправи виконуються студентами після засвоєння нового навчального матеріалу для його закріплення на протязі всього періоду навчання основ програмування при підготовці до чергової лабораторної роботи або після лекції, при виконанні завдань для самостійного виконання. Крім того, за допомогою завдань-вправ викладач може здійснювати контроль знань і визначати рівень засвоєння навчального матеріалу студентом під час лабораторних робіт, контрольних робіт, на колоквіумі і т.п. Також завдання-вправи можуть виконуватися студентами з метою самоконтролю.

У таблиці 2.2 наведено перелік основних типів завдань-вправ і відповідні приклади.

Таблиця 2.2

Типи завдань-вправ

№	Тип завдання-вправи	Приклад
1.	Вправи на перевірку знань теоретичного матеріалу	*Приклад мовою Pascal:* Відповісти на наступні питання: a) Чи може масив містити один елемент? b) Чи правильно, що тип елементів масиву може бути будь-яким? c) Чи може бути типом індексу масиву тип

		integer або real?
		Приклад мовою С: Припустимо, що є оголошення: `int grid[30][50].` Подайте адресу елементу `grid[5][7]` a) одним способом; b) двома способами; c) трьома способами.
2.	Вправи на формування навичок складання послідовності дій природною мовою, що описують хід розв'язування задачі	Дати словесний опис алгоритму розв'язування наступної задачі: задано три числа: a,b і c. Обміняти їх значення так, щоб виконувалась умова $a \le b \le c$.

3.	Вправи на перевірку знань синтаксису мови програмування	*Приклад мовою Pascal:*
		Дана послідовність операторів, за допомогою яких треба обчислити факторіал f натурального числа n:
		`k:=1; f=1; begin`
		`while k<n do f:=f*k`
		`k:=k+1 end`
		Знайдіть синтаксичні помилки. Скільки їх?

		Приклад мовою С:
		Знайдіть синтаксичні помилки в послідовності операторів:
		`if (x>2)`
		`  then x=2;`
		`if x<1`
		`  x=1;`

4.	Вправи на розуміння семантики мови програмування	*Приклад мовою Pascal:*
		1. Чи є правильними такі оператори присвоювання? Відповідь обґрунтувати.
		`k:=k mod 3+k*Cos(0);`
		`x:=x*2 div 6+x/4`
		2. Знайдіть помилки у фрагменті програми:
		`. . .`
		`var`
		`  a,b:array [1..10] of integer;`
		`  c  :array [1..10] of integer;`
		`begin`
		`  . . .`
		`  a:=b;`
		`  b:=c;`
		`end.`

		Приклад мовою С: Є оголошення: `float root[10], things[10][5], *pf, value = 2.2;` Визначити, чи є допустимими або не допустимими наступні оператори: a) `root[2]=value;` b) `scanf("%f" &root);` c) `root=value;` d) `printf("%f", root);` e) `things[4][4]=root[3];` f) `things[5]=root;` g) `pf=value;` h) `pf=root;`
5.	Вправи на формування навичок побудови інформаційної моделі задачі	*Приклад:* У деяких видах спортивних змагань виступ кожного спортсмена оцінюється декількома суддями, потім з усієї сукупності оцінок Яилучається найбільша і найменша оцінка, а для решти оцінок знаходиться середнє арифметичне, яке і зараховується спортсмену. Якщо найбільшу оцінку поставили декілька суддів, то із сукупності оцінок вилучають лише одну таку оцінку; аналогічно і для найменшої оцінки. Згідно з умовою задачі визначити величини, які є вхідними даними, результатами та проміжними даними, встановити співвідношення між ними, відобразити ці співвідношення за допомогою формул (або таблиць, графів).
6.	Вправи на формування умінь і навичок кодування алгоритму конкретною мовою програмування	*Приклад* [96]: Написати програму однією з мов програмування (Pascal, C) згідно алгоритму (сортування елементів масиву методом простих вставок): 1) [Цикл за `j`.] Виконати кроки від `2` до `5` при j=2, 3,..., N і після цього завершити виконання процедури. 2) [Установка `i, K, R`.] Присвоїти `i←j-1`, $K \leftarrow K_j$, $R \leftarrow R_j$ (на наступних кроках буде

		зроблена спроба вставити запис R у потрібне місце, порівнюючи K з K_i при спадних значеннях i). 3) [Порівняння K і K_i.] Якщо $K \geq K_i$, то перейти до кроку 5 (знайдено шукане місце для запису R). 4) [Переміщення R_i, зменшення i.] Присвоїти $R_{i+1} \leftarrow R_i$, $i \leftarrow i-1$. Якщо i>0, то повернутися до кроку 3 (якщо i=0, то K – найменший із розглянутих раніше ключів, а значить, запис R повинен зайняти першу позицію). 5) [Перенесення на місце R_{i+1}.] Присвоити $R_{i+1} \leftarrow R$.

7.	Вправи на формування навичок правильного оформлення програми	*Приклад мовою Pascal:*
		Фрагмент програми оформити відповідно до вимог і з'ясувати яка задача розв'язується:
		`. . .`
		`m:=a; n:=b;`
		`while not ((m=0) or (n=0)) do`
		`begin`
		`if m>=n then begin m:=m-n end else`
		`begin n:=n-m; end`
		`if m=0 then k:=n else k:=m;`
		`. . .`

		Приклад мовою C:
		Фрагмент програми оформити відповідно до вимог і з'ясувати якого значення набудуть змінні x, y, z після її виконання:
		`main() { int x, y = 1, z;`
		`if (y!=0) x=5;`
		`if (y==0) x=3; else x=5;`
		`x=1; if (y<0) if (y>0) x=3;else`
		`x=5;`
		`if (z=y<0) x>3; else if (y==0)`
		`x=5; else x=7;`
		`if (z=(y==0)) x=5; x=3;`
		`if (x=z=y); x=3;`
		`}`

<table>
<tr><td rowspan="2">8.</td><td rowspan="2">Вправи на формування умінь покрокового виконання програми</td><td>Приклад мовою Pascal:
Побудувати таблицю трасування фрагменту програми:
<pre>. . .
  j:=10;
for i:=1 to 5 do begin
  WriteLn(i, j);
  j:=j-2
end;
. . .</pre></td></tr>
<tr><td>Приклад мовою C:
Побудувати таблицю трасування програми:
<pre>#define PR(format, value)
        printf(#value"= %"
        #format"\t", (value))
#define NL putchar('\n')
#define PRINT1(f, x1)
        PR(f, x1), NL
#define PRINT2(f, x1, x2)
        PR(f, x1),
        PRINT1(f, x2)

#include <stdio.h>
main()
{
 int x, y, z;

 for (x=0, y=1000; y>1;
      x++, y/=10)
  PRINT2(d, x, y);
}</pre></td></tr>
<tr><td rowspan="2">9.</td><td rowspan="2">Вправи на формування навичок аналізу програм</td><td>Приклад мовою Pascal:
Скільки разів виконається цикл у фрагменті програми:
<pre>. . .
Read(x);
n:=10;
i:=0;
S:=0;
Sl:=1;
while i<=n do begin
  Sl:=Sl*x;
  S:=S+Sl
end;
. . .</pre></td></tr>
<tr><td>Приклад мовою C:
Якого значення набуде змінна z після виконання операторів:
<pre>. . .
int x, z;</pre></td></tr>
</table>

<table>
<tr><td></td><td></td><td>

```
x = 2;
z+ = -x++ - 1;
z = x / ++x;
. . .
```

</td></tr>
<tr><td rowspan="2">10.</td><td rowspan="2">Вправи на формування умінь і навичок побудови рекурсивних алгоритмів і відповідних програм</td><td>

Приклад мовою Pascal:

Яка задача розв’язується за допомогою наступного фрагменту програми:

```
. . .
min:=a[1];
for i:=1 to n do
  if a[i]<=min then
    min:=a[i];
. . .
```

Побудувати рекурсивний алгоритм і написати програму згідно з алгоритмом для розв’язування цієї задачі.

</td></tr>
<tr><td>

Приклад мовою C:

Яка задача розв’язується за допомогою наступного фрагменту програми:

```
. . .
int highbit (unsigned int x)
  {
   int i; h=0;

   for (i=16; i>=1; i>>=1) {
    if (x>>i) {
     h+=i;
     x>>=i;
    }
   }
  }
. . .
```

Побудувати рекурсивний алгоритм і написати програму згідно з алгоритмом для розв’язування цієї задачі.

</td></tr>
<tr><td>11.</td><td>Вправи на формування уміння і навичок читання і пояснення готових алгоритмів і програм</td><td>

Приклад мовою Pascal:

Яка задача розв’язується за допомогою фрагменту програми:

```
. . .
Read(a,b);
while b<>0 do begin
  tmp:=a mod b;
  a:=b;
  b:=tmp
end;
Write(a);
. . .
```

</td></tr>
</table>

		Приклад мовою С:
		Яка задача розв'язується за допомогою наступної функції?
		`int task (unsigned int x)`
		`{`
		`int i; h=0;`
		`for (i=16; i>=1; i>>=1) {`
		`if (x>>i) {`
		`h+=i;`
		`x>>=i;`
		`}`
		`}`
		`return(h);`
		`}`

Оскільки найважливішим видом навчальної діяльності, у процесі якої формується система знань, умінь і навичок з програмування, є розв'язування задач. Завдання–задачу можна вважати одним з основних завдань з основ програмування. При розв'язуванні невеликих навчальних задач студенти повинні:

– набути навичок структуризації даних і побудови простих обчислювальних алгоритмів, навичок поширення алгоритму розв'язування задачі на коло типових задач;

– оволодіти методами вибору найефективнішого методу розв'язування поставленої задачі;

– ознайомитися з різними методами побудови алгоритмів.

Кожне завдання-задача передбачає написання програми, її налагоджування, документування тексту програми, аналіз результатів її виконання. Тому таке завдання вимагає від студента більше інтелектуальних зусиль і часу, ніж завдання-вправа. Завдання–задачі, як правило, доцільно використовувати як індивідуальні завдання в циклі лабораторних робіт, завдання для самостійного виконання, завдання для контрольних робіт (табл. 2.3).

Таблиця 2.3

Типи завдань-задач

№	Тип завдань-задач	Приклад
1.	Завдання на формування умінь і навичок побудови алгоритму і написання програми з використанням основних алгоритмічних конструкцій для розв'язування навчальної задачі	*Приклад 1:* Скласти програму, за допомогою якої знайти число, яке отримується шляхом запису в зворотному порядку цифр заданого натурального числа n.
		Приклад 2: Дано дійсні числа a,b,x. Скласти програму, за допомогою якої обчислюється значення виразу: $$y=\left(\frac{1}{a*x+b}-\frac{1}{b*x-a}\right)^{\pi}.$$
2.	Завдання на формування умінь і навичок побудови алгоритмів методом структурної декомпозиції	*Приклад:* Дано координати вершин двох трикутників. Скласти програму, за допомогою якої визначається, який із трикутників має більшу площу. Для розв'язування задачі скласти дві підпрограми: підпрограму обчислення площі трикутника за формулою Герона, яка, у свою чергу, викликає підпрограму обчислення відстані між двома точками (вершинами трикутників).
3	Завдання на формування навичок документування текстів програм	*Приклад:* Дано символьний файл f. Скласти програму, за допомогою якої у файл g записуються компоненти файлу f у зворотному порядку, використавши для цього рекурсивну підпрограму. При створенні програми зробити пояснення окремих фрагментів програми у вигляді коментарів.
4.	Завдання на формування навичок вибору структур початкових і	*Приклад:* Дано ціле число n і дійсні числа $x_1,x_2,\ldots,x_n$. Скласти програму, за допомогою якої обчислюється значення виразів:

	проміжних даних і результатів, що найбільш відповідають інформаційній моделі задачі, та встановлення зв'язку між даними	$M=\frac{\sum_{i=1}^{n} x_i}{n}$, $D=\sqrt{\frac{\sum_{i=1}^{n}(x_i-M)^2}{n-1}}$. При складанні програми не використовувати структуровані типи даних.
5.	Завдання на формування навичок аналізу алгоритмів	*Приклад:* Скласти програми сортування елементів масиву наступними методами: a) сортування вибором; b) сортування злиттям; c) швидке сортування; d) сортування купою. Порівняти часову складність виконання програм і необхідний обсяг пам'яті.
6.	Завдання на побудову різних алгоритмів розв'язування поставленої задачі та порівняння їх ефективності	*Приклад:* Дано два неспадні масиви цілих чисел $x[1..k]$ і $y[1..l]$. Скласти програму, за допомогою якої знаходиться число різних елементів серед $x[1], x[2], \ldots, x[k]$ і $y[1], y[2], \ldots, y[l]$. Чи можна розв'язати цю задачу за $k+l$ кроків?
7.	Завдання на формування умінь та навичок проектування подійно-керованих програм	*Приклад:* Написати програму, що, відповідно до вибору користувача, в упорядкований список зв'язаного збереження додає елемент; видаляє елемент списку; здійснює пошук елемента з заданим значенням; виводить на екран список; припиняє свою роботу.
8.	Завдання на формування навичок пошуку ефективного алгоритму розв'язування задачі	*Приклад 1:* Дано натуральне число n, у якому послідовно виписані ступені числа 10: 1101001000100001000001000000… Скласти програму, за допомогою якої визначається k-та цифра цього числа. При розв'язуванні задачі не використовувати оператори циклу, безумовного

		переходу і рекурсивні підпрограми.
		Приклад 2: Дано дійсні числа x, y, z. Скласти програму, за допомогою якої отримується $\max(x, y, z)$ за найменшу кількість порівнянь.
9.	Завдання на формування знань методів побудови алгоритмів і умінь вибору кращого методу для розв'язування конкретної задачі	*Приклад 1:* Дано дві послідовності. Побудувати алгоритм, заснований на динамічному програмуванні, та скласти програму, за допомогою якої знаходиться спільна підпослідовність найбільшої довжини цих послідовностей. *Приклад 2:* Нехай дано n заявок на проведення занять в одній і тій же аудиторії. У кожній заявці зазначені початок і кінець заняття. Різні заявки можуть перетинатися в часі, і тоді можна задовольнити тільки одну з них. Побудувати алгоритм, якій заснований на жадібному алгоритмі, та скласти програму, за допомогою якої буде обрано максимальну кількість спільних одна з однією заявок.
10.	Завдання на формування навичок формулювання постановки задачі	*Приклад:* Написати програму, яка визначає рейтинг студентів групи. Згідно із заданою проблемою розробити детальний план розв'язання завдання: визначити критерії оцінювання успішності кожного студента групи, визначити вимоги до інтерфейсу програми.

Формуванню вміння точно формулювати питання та лаконічно висловлювати думки, вихованню почуття відповідальності за результати праці вже в процесі навчання сприяють завдання *на складання тестових питань з*

певної теми навчального матеріалу. Тому, ми вважаємо, завдання такого типу також потрібно включати в процес навчання майбутніх програмістів (табл. 2.4).

Таблиця 2.4

Завдання на складання тестових питань з певної теми навчального матеріалу

№	Тип завдання	Приклад
1.	Складання тестових питань з певної теми навчального матеріалу	*Приклад 1:* Підготувати тестові питання і варіанти відповідей до них із теми: „Стандартні типи даних мови Pascal: числові типи, булевський тип, символьний тип, покажчик.”
		Приклад 2: Підготувати тестові питання і варіанти відповідей до них із теми: „Класи пам'яті в мові С. ”

У додатку Ж наведено приклади тестових питань, запропонованих студентами.

До переваги завдань–досліджень слід віднести формування у студентів таких якостей, як уміння знаходити матеріал за темою реферату або доповіді, аналіз зібраних відомостей, здатність робити висновки, досвід публічного виступу і досвід оволодіння прийомами публічного захисту роботи, що виконана. Також формуються навички роботи з навчальною, науковою і спеціальною літературою, з періодичними виданнями, Internet- ресурсами і т.п. (табл. 2.5).

Таблиця 2.5

Типи завдань-досліджень

№	Тип завдання-дослідження	Приклад
1.	Підготовка доповіді	*Тема доповіді:* Подібності та відмінності статичних та динамічних масивів.
		Тема доповіді: Особливості внутрішнього та зовнішнього сортування.
2.	Написання реферату	*Тема реферату:* Формальні методи опису синтаксису мови програмування. Форма Бекуса-Наура і контекстно-вільні граматики. Розширена форма БНФ. Синтаксичні графи.
		Тема реферату: Порівняльна характеристика операторів циклу в мовах програмування Pascal і C.

Виконання завдань на підготовку доповіді (виступу), написання реферату та складання тестів з певної теми навчального матеріалу вимагають тривалого відрізку часу, тому таких завдань у студента повинно бути небагато, але достатньо для початку формування зазначених вище якостей. Зрозуміло, що такі завдання слід виносити на самостійну роботу студентів.

Завдання з розробки програмного продукту призначене для вирішення найважливіших задач у навчанні майбутніх інженерів-програмістів, а саме:

- надати студентові можливість здійснити перехід від написання невеликих навчальних програм до створення повноцінного програмного продукту, працювати в колективі вже на ранніх етапах навчання;
- забезпечити розуміння зв'язку теорії і практики, сфери застосування отриманих знань.

Підкреслимо, що діяльність із розробки проекту є для студентів не лише засвоєнням когнітивного досвіду, але й надає їм можливість набути досвіду креативної діяльності. Крім того, цей вид діяльності виконує особистісно-розвиваючу функцію: надає можливість студенту проявити себе як особистість, проявити творчу ініціативу, взяти на себе відповідальність за виконану роботу, уміти погоджувати свої міркування з учасниками розробки, виконувати своєчасно свою частину роботи, бути готовим аргументовано відстоювати свою точку зору. Усі ці якості будуть затребувані в процесі майбутньої професійної діяльності.

Створення програмного продукту передбачає наявність, щонайменше, знань базових конструкцій хоча б однієї мови програмування, стандартних алгоритмів пошуку і сортування, методів побудови алгоритмів; навичок кодування, налагодження, тестування програм; уміння інтегрувати програмні модулі, що створені окремо.

Завдання на створення програмного продукту передбачає розв’язування однієї складної, але посильної для студентів задачі, і повинно виконуватися наприкінці вивчення курсу з основ програмування.

У Таблиці 2.6 наведено приклади завдань з розробки програмного продукту.

Таблиця 2.6

Завдання з розробки програмного продукту

№	Тип завдання	Приклад
	Розробка програмного продукту	*Приклад 1:* Створити програмний продукт, за допомогою якого реалізується графічна модель руху планет навколо Сонця.
		Приклад 2: Створити програмний продукт, за допомогою якого реалізується дошка Гальтона для демонстрації законів теорії ймовірності.

		Приклад 3: Створити програмний продукт, за допомогою якого реалізується найпростіший текстовий редактор.
		Приклад 4: Створити програмний продукт, за допомогою якого реалізується обертання кубу з різнокольоровими гранями.
		Приклад 5: Створити програмний продукт, за допомогою якого реалізується виконавець “Черепашка”.
		Приклад 6: Створити програмний продукт, за допомогою якого реалізується задача «Калькулятор».
		Приклад 7: Створити програмний продукт, за допомогою якого реалізується гра «Реверсі»
		Приклад 8: Створити програмний продукт, за допомогою якого реалізується гра «Тетрис»

Досвід роботи показує, що ретельно продуманий за змістом і послідовністю широкий спектр різноманітних видів і типів навчальних завдань з основ програмування різного рівня складності надає можливість інтенсифікувати процес навчання, сприяє його високій результативності при підготовці майбутніх інженерів-програмістів.

2.4.3. Використання інформаційно-комунікаційних технологій у навчанні основ програмування. Інформаційно-комунікаційні технології – інформаційні технології на базі персональних комп’ютерів, комп’ютерних мереж і засобів зв’язку, для яких характерна наявність дружнього середовища роботи користувача [134]. Будемо дотримуватися класифікації програмного

забезпечення інформаційно-комунікаційних технологій, запропонованої у [1], згідно з якою ПЗ ІКТ розподіляють на:

- інформаційні продукти праці фахівців, що займаються проблемами освіти;
- комп'ютерні навчальні програми, що включають електронні підручники, тренажери, лабораторні практикуми, тестові системи;
- навчальні системи на базі мультимедійних технологий, побудованих з використанням персональних комп'ютерів, відеотехніки, накопичувачів на оптичних дисках;
- розподілені бази даних з галузь знань.

Застосування інформаційно-комунікаційних технологій в навчальному процесі сприяє підвищенню рівня навчально-пізнавальної активності студентів; посиленню їх інтересу до навчання, в тому числі до способів здобування знань; індивідуалізації та диференціації навчання; підвищенню наочності навчання [201].

Нами розроблено педагогічні програмні засоби для підтримки окремих тем курсу основ програмування, вивчення яких викликає у студентів певні труднощі, наприклад ППЗ ”Linears Rolls” з теми ”Лінійні списки”. У програмі реалізовані всі допустимі операції над однозв'язними списками: створення списку, елементи якого впорядковані за зростанням або за спаданням, або не впорядковані; додавання елемента в початок або кінець списку, перед або після елемента із зазначеним значенням; видалення першого або останнього елементу списку, перед або після елементу із зазначеним значенням; пошук елемента із заданим значенням; виведення „сусідів” елемента із заданим значенням (рис. 2.1).

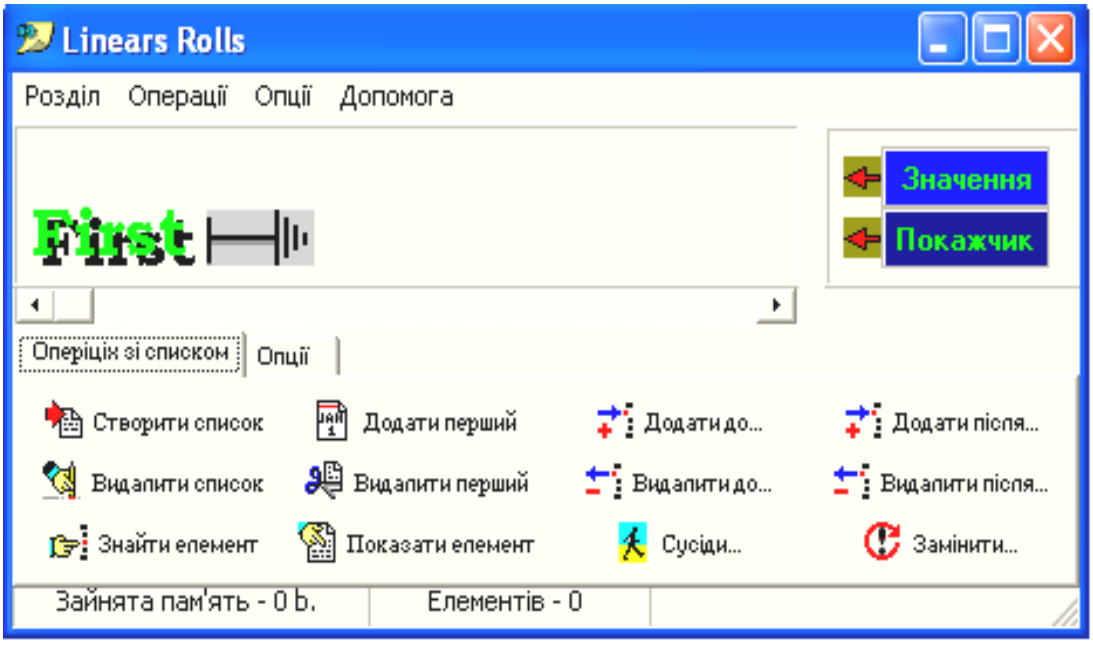


Рис. 2.1. Головне вікно програми «Linears Rolls».

Передбачено можливість роботи зі списком в режимі трасування. У цьому випадку в окремому вікні виводиться текст програми мовою Pascal і кольором виділена команда, що виконується в цей момент (рис. 2.2).

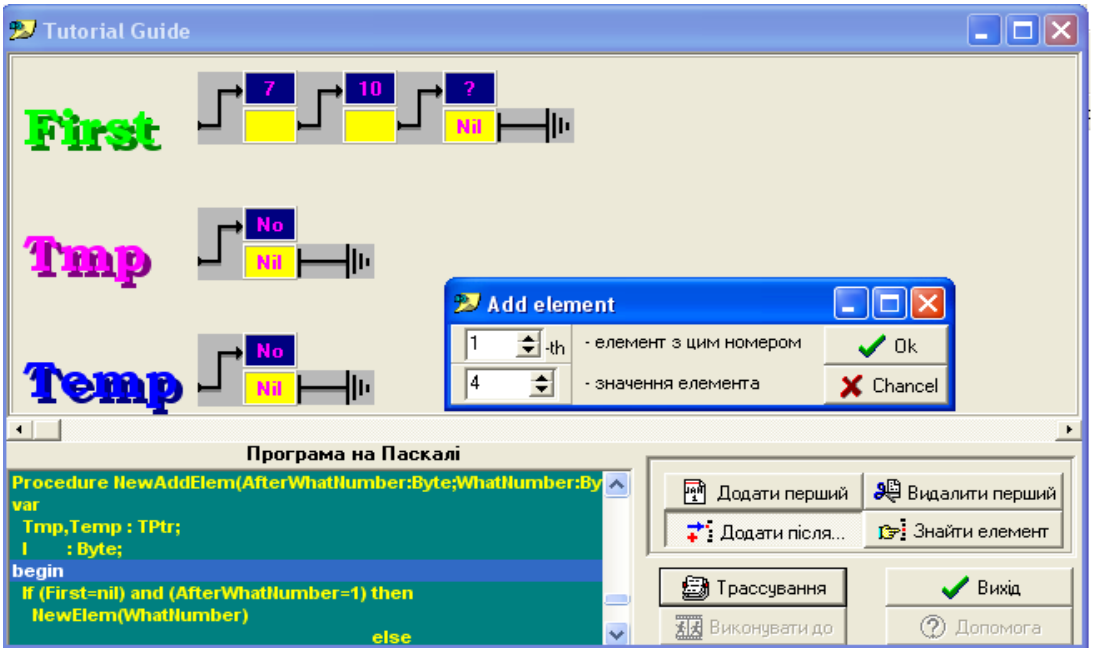


Рис. 2.2. Додавання елементу списку. Режим трасування.

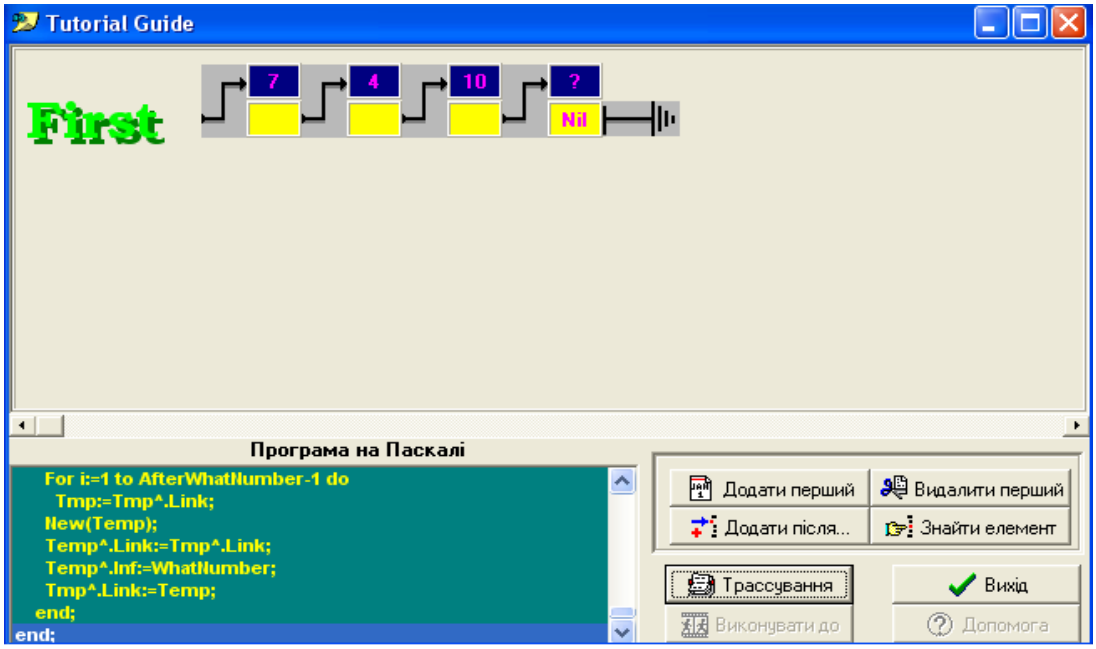


Рис. 2.3. Список після операції додавання.

У програмі розроблений зручний для користувача інтерфейс. Зокрема, виконання операцій додавання, вилучення, обхід списку можна здійснювати за допомогою меню команди «Операції» (рис. 2.4).

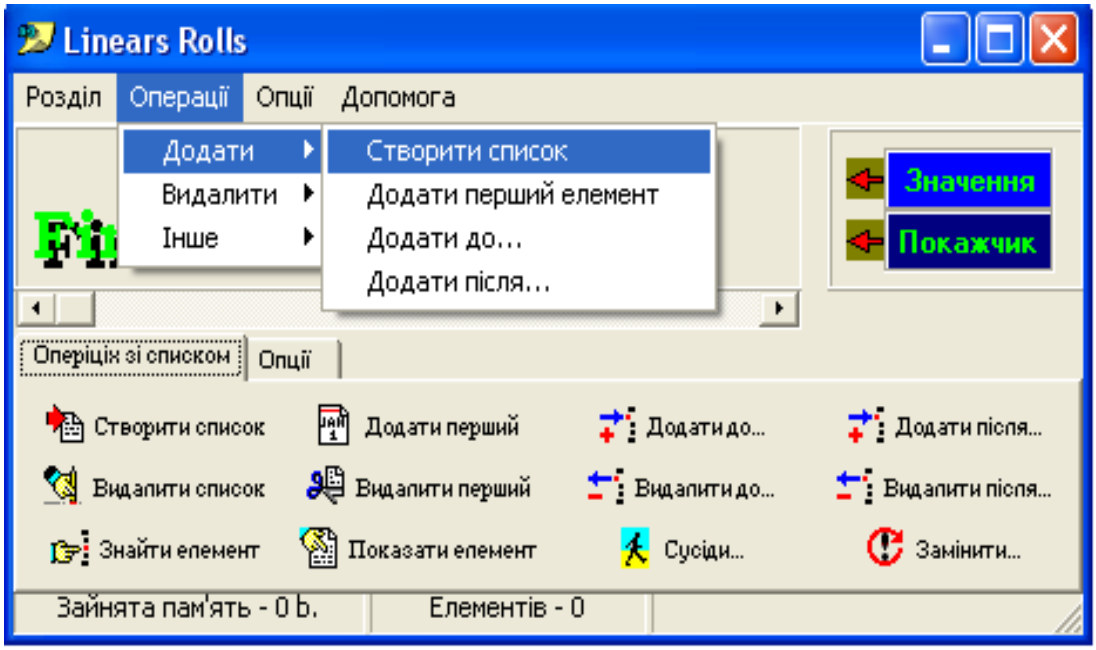


Рис. 2.4. Меню команди «Операції».

Всі повідомлення програми можуть виводиться, залежно від бажання користувача, українською, російською або англійською мовою (рис. 2.5).

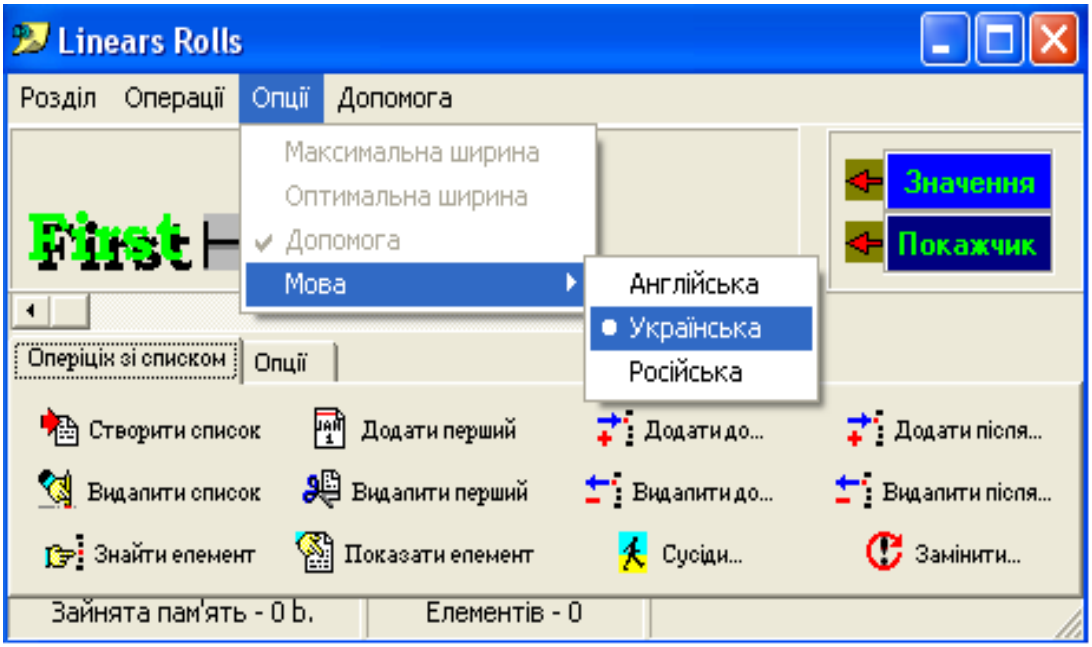


Рис. 2.5. Меню команди «Опції».

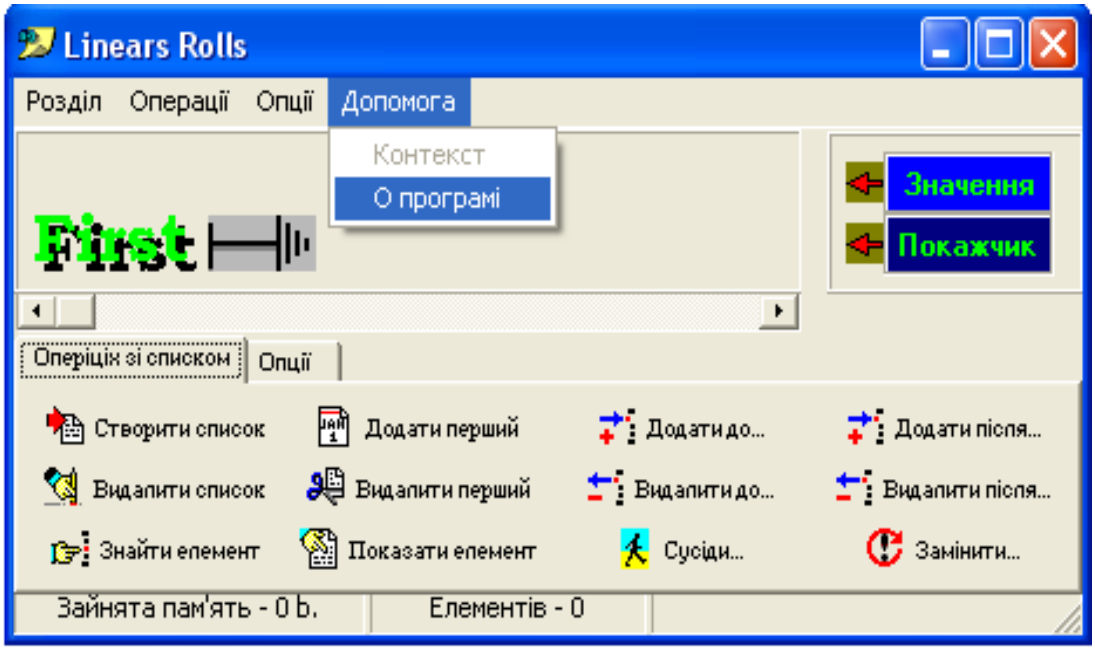


Рис. 2.6. Меню команди «Допомога».

Також у програмі передбачена функція тестування обчислювальної системи користувача, яка доступна з пункту меню команди «Допомога» (рис. 2.7).

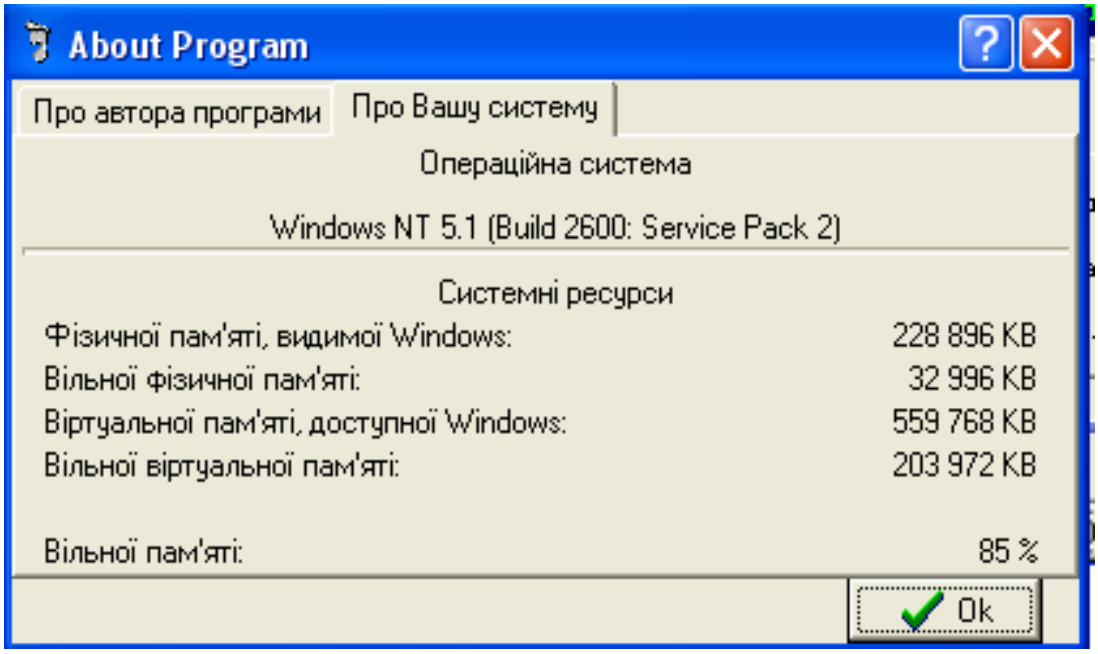


Рис. 2.7. Інформація о системних ресурсах.

ППЗ «Linears Rolls» використовується у навчальному процесі під час самостійної роботи студентів.

Нами розроблено електронний посібник „Програмування мовою С” на основі Internet-технологій, якій містить, зокрема, теоретичний матеріал (рис. 2.8), завдання-вправи (рис. 2.9), завдання-задачі за трьома рівнями складності (рис. 2.10, 2.11).

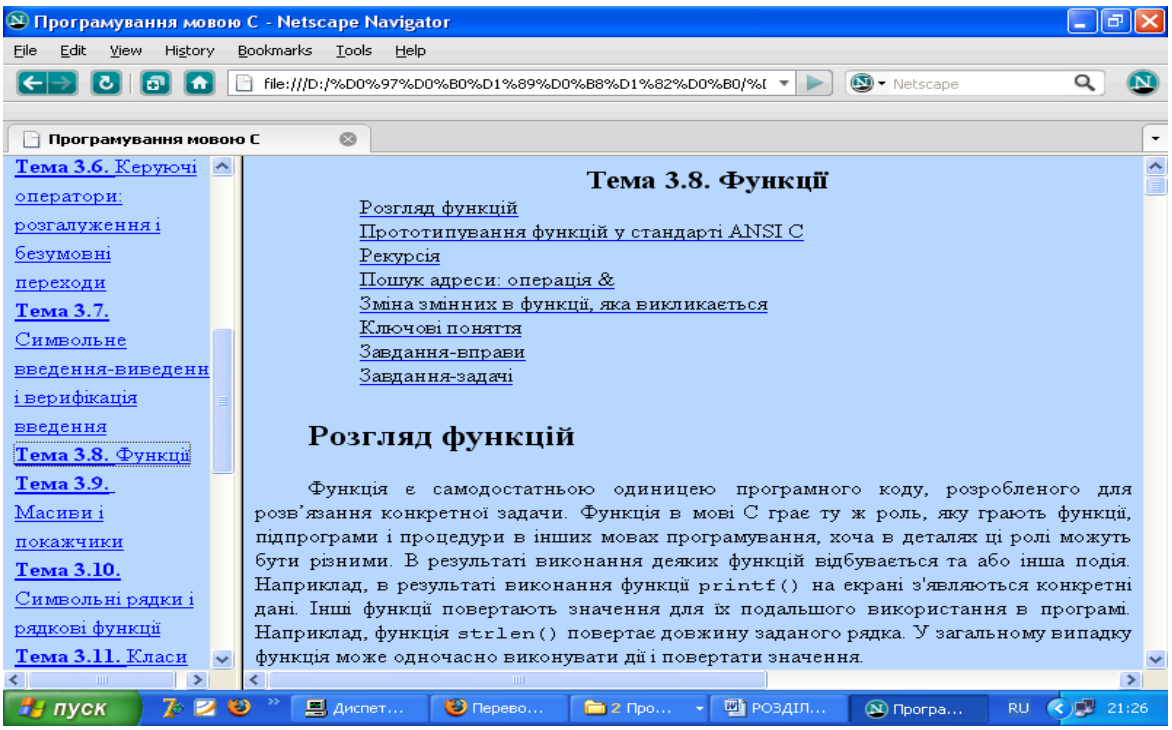


Рис. 2.8. Вікно з теоретичним матеріалом з теми „Функції”.

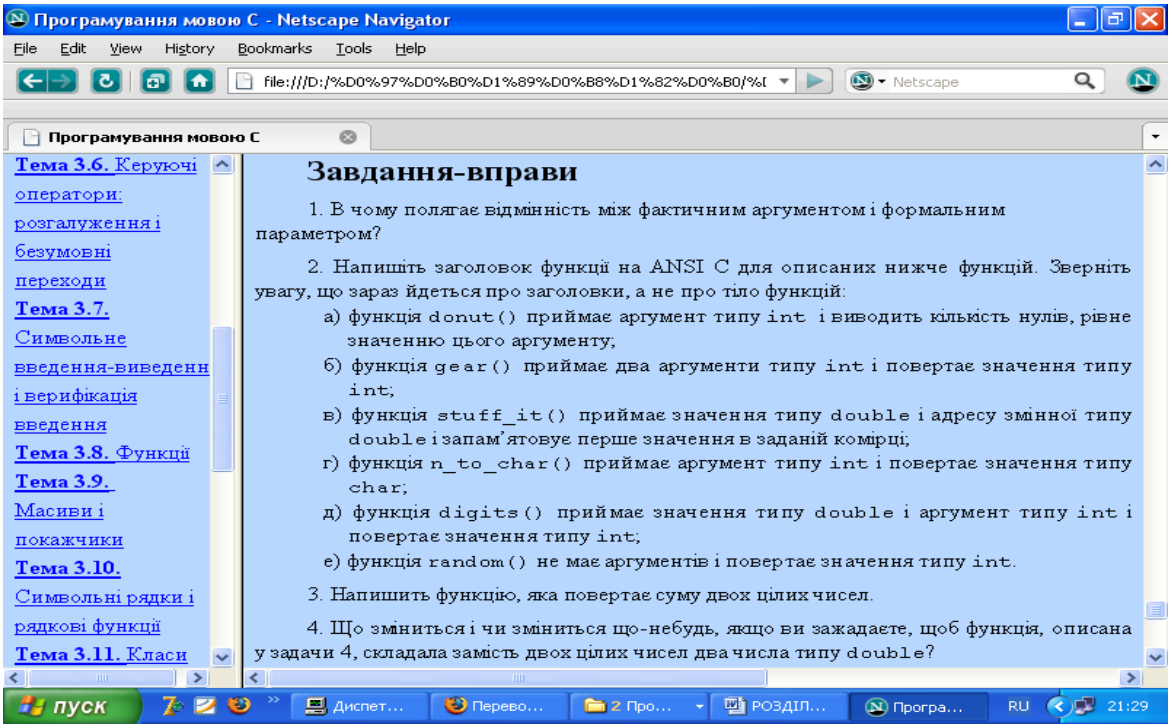


Рис. 2.9. Вікно з завданнями-вправами з теми „Функції”.

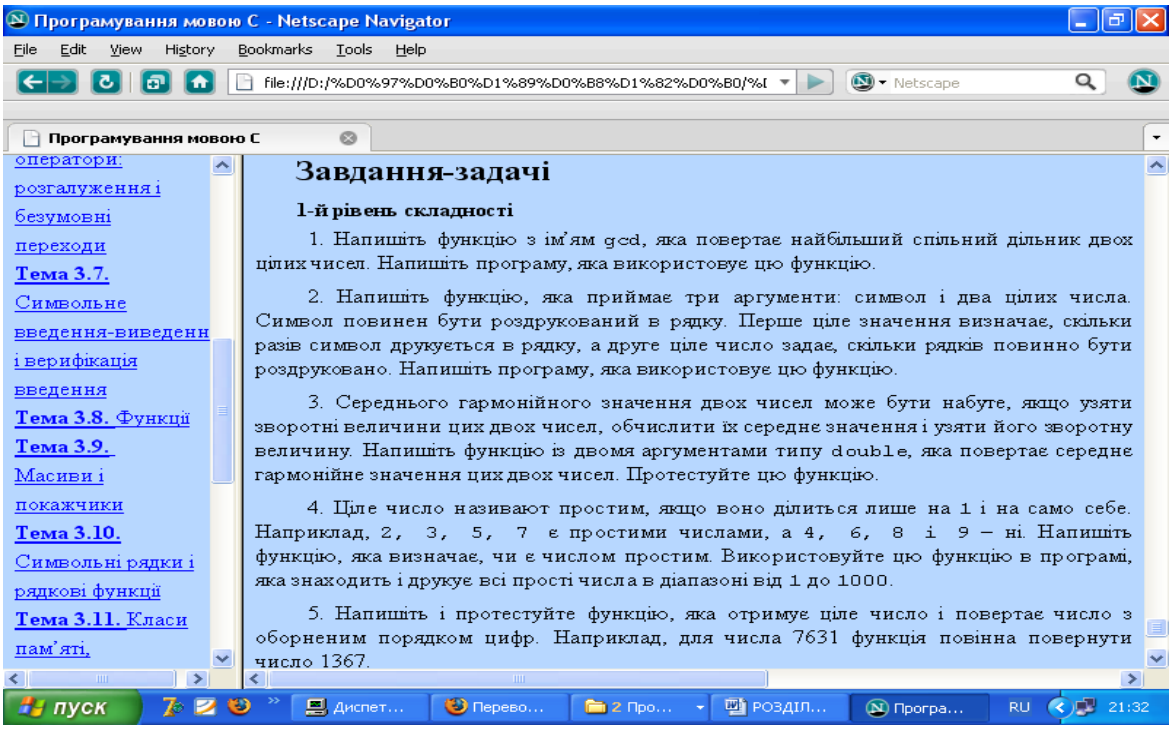


Рис. 2.10. Вікно з завданнями-задачами 1-го рівня складності з теми «Функції».

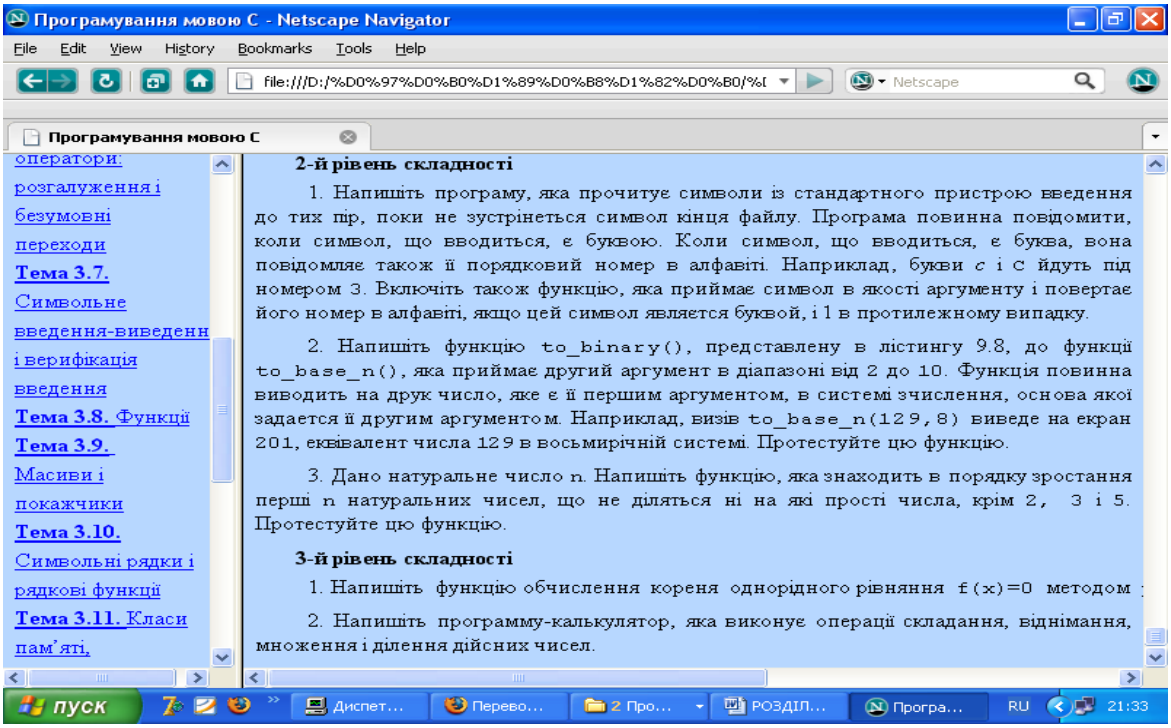


Рис. 2.11. Вікно з завданнями-задачами 2-го і 3-го рівня складності з теми «Функції».

Технічні засоби навчання (кодоскоп, мультимедійний проектор і т.п.) та *нові інформаційно-комунікаційні технології* зручно використовувати на лекціях при викладанні теоретичного матеріалу. Використання цих засобів надає можливість інтенсифікувати процес викладання теоретичного матеріалу завдяки демонстрації завчасно підготовлених малюнків, графіків, формул, блок-схем, текстів програм, їх покрокового виконання і т.п. У додатку З наведено приклад конспекту лекції, підготовлений у вигляді презентації за допомогою програмного продукту PowerPoint.

2.4.4. Системи програмування. Система програмування – частина базового програмного забезпечення, що підтримує процес програмування [140]. До складу системи програмування традиційно входять наступні програмні інструменти і бібліотеки:

– редактор для побудови програм;

– транслятор для перекладу програм з мови програмування на машинну мову;

– налагоджувач для перевірочних запусків програм і виправлення помилок;

– бібліотеки періоду трансляції й періоду виконання;

– засоби керування компіляцією й побудовою програмного проекту;

– монітор, що інтегрує в собі перелічені вище засоби та організовує функціонування системи й програмування в цілому.

Традиційно в нашій країні при вивченні мов програмування Pascal, C і C++ використовують професійні інтегровані середовища розробки фірми Borland, незважаючи на деякі їх недоліки (див. 1.4.3)

Останнім часом у навчальному процесі часто використовують середовище програмування Free Pascal (повна назва Free Pascal Compiler, FPC) – вільно розповсюджуване середовище мови програмування Pascal [180]. Важливою особливістю даного компілятора є орієнтація на поширені комерційні діалекти мови Object Pascal і середовище Delphi. Free Pascal є сумісним з Turbo Pascal, за винятком того, що компілює програми для захищеного режиму процесора, в якому неможливе пряме звертання до пам’яті, з Borland Delphi (включає підтримку класів і інтерфейсів), частково – з GNU Pascal. Це середовище програмування можна безкоштовно отримати у сети Internet, наприклад, за такою адресою: http://www.mnogosofta.com/soft_61602.html.

У зв'язку з усіма перерахованими вище причинами існує необхідність у розробці систем програмування, орієнтованих на використання в навчальному процесі для навчання основ програмування, які надають можливість будувати алгоритми розв’язування завдання у різних видах, наприклад, у вигляді блок-схеми або диаграммой Несси–Шнейдермана, включають перевірку відповідності тексту програми побудованому алгоритму, підтримують кілька мов програмування.

2.5. Форми організації навчальної діяльності студентів при навчанні основ програмування

У вищій школі використовують різні форми організації навчання, які розділяють за функціональними ознаками [191]:

– *форми організації засвоєння знань, формування навичок і умінь, пошуку нових знань*: лекція; лабораторна робота; практичне заняття; консультація; семінар; навчальна конференція; самостійна робота студентів; науково-дослідна робота студентів; навчальна, педагогічна і виробнича практики; написання курсових робіт, дипломної роботи (проекту) та ін.;

– *форми організації контролю знань, умінь і навичок:* колоквіум; поточний, проміжний, рубіжний контролі; залік; екзамен; захист курсових і дипломних робіт (проектів) та ін.

Найбільш поширеними формами організації навчання в сучасних загальноосвітніх ([208], [211]) і вищих навчальних закладах ([165], [212]) є заняття у групі [206], до яких відносять лекції, лабораторні роботи, практичні і семінарські заняття, консультації.

Форми організації навчальної діяльності визначаються особливостями дисципліни, що вивчається. Навчальними планами курсу з основ програмування передбачені наступні форми організації навчальної діяльності: лекції, лабораторні роботи, організація самостійної роботи студентів, консультації.

Розглянемо особливості форм організації навчальної діяльності при навчанні основ програмування.

Лекція є одним з основних видів навчальних занять у ВНЗ. Я.Я. Болюбаш [142] дає таке тлумачення лекції: „Навчальна лекція – це логічно завершений, науково обґрунтований і систематизований виклад певного наукового або науково-методичного питання ілюстрований, при необхідності, засобами наочності та демонстрацією дослідів”.

У вищій школі до лекції пред'являють наступні вимоги ([149], [185]):

– мотивація актуальності навчального процесу;

– науковість та інформативність, що означає ознайомлення студентів із певними науковими відомостями на сучасному науковому рівні, наявність достатньої кількості переконливих прикладів даних теоретичних положень, емоційність подання навчального матеріалу, оптимальний темп подання, активізація розумової діяльності студентів, пояснення нових означень і термінів, використання для наочності дидактичних і технічних засобів навчання (зокрема комп'ютера).

Лекція призначена для послідовного, систематичного подання теоретичного матеріалу, а також визначає напрям, основний зміст і характер усіх інших видів навчальної діяльності.

При традиційному підході читання лекцій „викладач і студент залишаються функціонально відокремленими: функція викладача – викладання, функція студента – фіксування матеріалу в пам'яті та конспекті. З точки зору організації та ведення лекції функція викладача є активною, функція студента – пасивною" [105]. Така ситуація пов'язана з тим, що студентам, як правило, заздалегідь не відомий зміст лекції. Тема і план лекції, які викладач дає на початку лекції, студентами ще не осмислені в зв'язку з тим, що на це не було часу.

Для подолання формального засвоєння лекційного курсу, для забезпечення осмисленого засвоєння теоретичного матеріалу студентами, його розуміння, запам'ятовування основних положень, використання засвоєних знань треба залучити студентів до активної роботи впродовж лекції. Для цього треба заздалегідь знайомити студентів із планом майбутньої лекції, її змістом, з навчально-методичними матеріалами, літературою. Лекція у такому випадку стає більш доступною за змістом теоретичного матеріалу. При модульно-рейтинговій системі навчання можна заохочувати студентів додатковими балами за продумані, коректно поставлені питання і короткі виступи під час

лекції. Для посилення організаційної функції лекції та для забезпечення внутрішніх предметних зв'язків доцільно проведення на початку кожної лекції письмового тематичного диктанту з 4-5 невеликих завдань теоретичного характеру за матеріалами попередніх лекцій.

Приклад письмового тематичного диктанту з теми «Операції, вирази й оператори» змістового модулю «Основи мови програмування С».

ПІБ…….

Вариант №1

1. Припустимо, що всі змінні мають тип int. Знайдіть значення кожної з наступних змінних:

```
x = (2 + 3) * 6;          _____________________
x = (12 + 6)/2*3;         _____________________
y = x = (2 + 3)/4;        _____________________
y = 3 + 2*(x = 7/2);      _____________________
```

2. Знайдіть помилки в наведеній нижче програмі:

```
int main(void)
        {                 _________
  int i = 1,              _________
  float n;                _________
  printf("Будьте обережні!  Далі йде послідовність дробів!\n");
                          ________
  while (i  < 30)         _________
   n = i / 1;             _________
  printf(“%f”, n);        _________
  printf ("От і все!\n");
                          _________
  return;                 _________
  }                       _________
```

3. Що буде надруковано в результаті виконання програми?

```
#include <stdio.h>
#define FORMAT “%s C - это круто!\n”
```

```
int main(void)
 {
  int num = 10;

  printf(FORMAT, FORMAT);
   printf(“%d\n”, ++num);
   printf(“%d\n”, num++);
   printf(“%d\n”, num--);
   printf(“%d\n”, num);
  return 0;
}
```

_____

_____

_____

_____

_____

4. Що буде надруковано в результаті виконання програми?

```
#include <stdio.h>
#define MESG “COMPUTER BYTES DOG”
int main(void)
 {
  int n = 0;

  while (n < 5)
printf(“%s\n”, MESG);
   n++;
    printf(“Молодець!\n”);
    return 0;
 }
```

_____

_____

_____

_____

_____

5. Напишіть оператори, які виконують наступні дії:

а) зменшує значення змінної x на 1;

_____

b) привласнює m залишок від ділення n на k;

_____

c) ділить q на b, віднімає a і привласнює результат p;

_____

d) привласнює змінній x результат ділення суми значень змінних a і b на добуток c і d.

_____

З метою активізації навчально-пізнавальної діяльності студентів, ефективного використання лекційного часу, забезпечення наочності подання навчального матеріалу доцільно проводити *комп'ютерно-орієнтовані лекції* з використанням ІКТ і технічних засобів навчання. Згідно з [201], «комп'ютерно-орієнтована лекція – систематичне, послідовне і логічне подання проблемних ситуацій з розділів конкретної науки із використанням відео і комп'ютерної техніки для демонстрації малюнків, графіків, динамічних зображень і ін.».

Позитивними моментами проведення комп'ютерно-орієнтованих лекцій є:

– зосередженість і готовність студентів до роботи під час лекції;
– гарантована підготовка до лекції з боку студентів;
– творча активність студентів під час лекції;
– осмислене конспектування змісту лекції;
– міцне засвоєння навчального матеріалу;
– стимулювання до подальшого самостійного здобування знань.

Лабораторна робота – це вид навчального заняття, на якому студенти під управлінням викладача проводять експерименти або дослідження в спеціально обладнаних навчальних лабораторіях із використанням обладнання,

пристосованого до умов навчального процесу [142]. Лабораторні роботи сприяють глибшому розумінню теоретичного матеріалу [4].

Якщо в лекційному курсі розкриваються фундаментальні поняття основ програмування, тобто переважає науковий аспект дисципліни, то лабораторні роботи з основ програмування призначені для того, щоб навчити студентів всім етапам програмування: розуміння умови задачі, розробка алгоритму, кодування алгоритму мовою програмування, налагодження програми, її виконання, аналіз результатів. У той же час, на лабораторних роботах з основ програмування прищеплюються навички роботи на комп'ютері, навички роботи в середовищі мови програмування. Виходячи із цього, можна зробити висновок, що в курсі з основ програмування:

– комп'ютер є засобом навчання програмуванню;

– дослідження і розв’язування навчальних і реальних задач за допомогою комп'ютера є метою навчання;

– забезпечуючи постійний, об'єктивний і позбавлений емоцій, зворотний зв'язок зі студентом, комп'ютер є інструментарієм індивідуального і розвиваючого навчання.

Власний досвід автора підтверджує, що структура лабораторної роботи з основ програмування повинна складатися з наступних компонент:

1) актуалізація набутих на попередніх заняттях знань;

2) контроль і перевірка рівнів засвоєння навчального матеріалу студентами;

3) формування нових знань, умінь і навичок студентів;

4) застосування нових знань, умінь і навичок.

Вибір методів навчання, засобів і форм організації навчальної діяльності залежать від конкретної теми заняття.

Під час проведення лабораторних робіт, при виконанні студентами навчальних завдань за допомогою комп'ютера завдання викладача полягає у тому, щоб надати студентам, у разі потреби, допомогу, зокрема: підказати ідею

розв’язання задачі, звернути увагу на помилки в алгоритмах і програмах, проконсультувати щодо налагодження програми та інтерпретації одержаних результатів. У додатку І наведено приклад одного з завдань до лабораторної роботи з теми «Функції» змістового модулю «Основи мови програмування С».

Досвід показує, що при навчанні основ програмування необхідно надати можливість студентам *самостійно* працювати в комп'ютерному класі після аудиторних занять під керівництвом викладача. Для цього доцільно скласти розклад роботи комп'ютерного класу, в якому вказані дні тижня і час роботи для кожної групи студентів, що вивчають курс з основ програмування. Наявність розкладу гарантує можливість роботи на комп'ютері для виконання навчальних завдань, дисциплінує студентів. При цьому у комп'ютерному класі повинен бути присутній викладач, який, у разі потреби, надасть студентові потрібну консультацію.

Самостійність студента проявляється в потребі й умінні самостійно мислити, у здатності орієнтуватися в новій ситуації, самому бачити проблему, розуміти задачу і знайти спосіб її розв’язування. Стрімкий розвиток інформатики змушує фахівця в галузі комп'ютерних наук учитися протягом усього життя. Тому прищеплювання студентам, майбутнім інженерам-програмістам, умінь і навичок самостійної роботи необхідно для забезпечення їхнього професійного зростання протягом усієї трудової діяльності.

Самостійну роботу студента варто розуміти як сукупність всіх видів його навчальної діяльності під час занять [148]: уміння слухати і конспектувати лекцію; уміння працювати над конспектом після лекції; уміння працювати з науковою літературою, довідковим матеріалом, першоджерелами, документальними матеріалами; складання доповідей, написання рефератів, робота в комп'ютерному класі в позаудиторний час. До самостійної роботи можна віднести самостійне вивчення теоретичного матеріалу під час виконання домашнього завдання, розв’язування задач під час лабораторної роботи.

Самостійна робота студента повинна носити систематичний і цілеспрямований характер і мати реальну мету.

На наш погляд, при навчанні основ програмування одним з основних видів самостійної роботи студентів є колективна робота з розробки навчального програмного продукту (див. п. 2.3.2.4).

2.6. Елементи модульно-рейтингової системи навчання основ програмування

Пропонуються наступні принципи побудови модульно-рейтингової системи навчання (МРСН) з основ програмування:

1. МРСН повинна відповідати принципам Болонської декларації.

2. Для забезпечення ефективності МРСН необхідно на початку курсу довести до відома студентів принципи побудови і використання МРСН, а також на початку кожного семестру знайомити студентів із рейтинг-планом поточного семестру (додаток А). Впродовж усього семестру у кожного студента повинна бути можливість дізнатися про кількість набраних їм балів на даний момент.

3. Навчання основ програмування організовується за модульним принципом. Навчальний матеріал розподіляється на модулі, кожний з яких закінчується обов'язковою модульною контрольною роботою (додаток А). Кількість зароблених студентами балів залежить від рівня засвоєння змісту навчального модуля. Одержані бали за весь період навчання дисципліни підсумовуються і складають рейтинг студента з дисципліни.

4. Рейтинг кожного студента з основ програмування складається з наступних видів робіт:

основних:

– виконання обов'язкових завдань лабораторних робіт;

– виконання модульних контрольних робіт;

– створення програмного проекту;

– складання екзамену;

додаткових:

– розв'язування додаткових завдань-вправ і завдань-задач;

– підготовка рефератів, статей, виступів;

– висування ідей, власних пропозицій;

– розробка тестів з певної теми навчального матеріалу;

– результати опитування на лекціях, лабораторних заняттях;

– результати тестування;

– результати тематичних диктантів.

5. Кожен студент повинен виконати завдання лабораторних робіт, виконати передбачені контрольні роботи, розробити міні-проект (брати участь у колективній розробці), складати екзамен.

6. Не дивлячись на набрану кількість балів, на оцінку „відмінно” може претендувати лише студент, який розробив власний міні-проект або брав участь у його створенні в складі групи розробників.

7. Поточний рейтинг студента визначається сумою балів, які йому нараховуються за виконання основних і додаткових видів робіт, зазначених у пункті 4.

8. Якщо в поточному семестрі немає екзамену, а є залік, то залік студент одержує в тому випадку, якщо протягом семестру він набрав не менше 60% від загальної кількості балів, які він міг отримати за передбачені в цьому семестрі види робіт.

9. МРСН передбачає механізми стимулювання навчально-пізнавальної діяльності студентів, їх ініціативи і творчості в самостійній роботі.

На різних спеціальностях кількість балів, які можуть отримати студенти за різні види навчальної діяльності, може бути різною. Наприклад, студенти спеціальності „Програмне забезпечення автоматизованих систем” значну частку балів можуть отримати, виконуючі завдання до лабораторних робіт і за

участь у створенні навчального програмного продукту – міні-проекту. А для студентів спеціальності „Прикладна математика” важливою є не лише практика програмування, але й не менш важливими є знання теорії програмування.

У таблиці 2.7 пропонується співвідношення балів за різні види навчальної діяльності студентів спеціальностей „Програмне забезпечення автоматизованих систем” і „Прикладна математика”.

Таблиця 2.7

Співвідношення балів за різні види навчальної діяльності студентів

Види навчальної діяльності	„Прикладна математика”	„Програмне забезпечення АС”
Лабораторні роботи	30%	40%
Модульні контрольні роботи	15%	10%
Створення міні-проекту	25%	25%
Екзамен	25%	20%
Інші види робіт	5% і більше	5% і більше

2.7. Експериментальна перевірка ефективності розроблених компонентів методичної системи навчання основ програмування

2.7.1. Організація і проведення педагогічного експерименту. Метою педагогічного експерименту було:

– проведення аналізу системи знань, умінь і навичок, які повинні формуватися у студентів спеціальностей „Програмне забезпечення автоматизованих систем” та „Прикладна математика” в курсі з основ програмування;

– виявлення недоліків та переваг традиційної методичної системи навчання основ програмування студентів вищих навчальних закладів, майбутніх інженерів-програмістів;

– визначення шляхів підвищення практичної значущості результатів навчання майбутніх інженерів-програмістів;

– перевірка ефективності компонентів створеної методичної системи навчання основ програмування майбутніх інженерів-програмістів на основі новітніх педагогічних та інформаційно-комунікаційних технологій.

Експеримент проводився в Черкаському національному університеті імені Богдана Хмельницького на математичному факультеті та факультеті інформаційних технологій та біомедичної кібернетики, Інституті Соціального Управління економіки і права (м. Черкаси), Черкаській філії ПВНЗ «Європейський університет», Полтавському державному педагогічному університету імені В.Г. Короленка.

Педагогічний експеримент передбачав констатувальний, пошуковий і формувальний етапи.

Метою *констатувального етапу* педагогічного експерименту (2000-2002 р.р.) було:

– встановити орієнтовний рівень знань, умінь і навичок студентів першого курсу з інформатики, необхідних для успішного засвоєння дисциплін з комп'ютерних наук та використання в майбутній професійній діяльності;

– проаналізувати зміст теоретичного матеріалу курсу з основ програмування, виходячи з цілей навчання у вищих навчальних закладах відповідних напрямів підготовки;

– проаналізувати зміст лабораторного практикуму з основ програмування і його роль в системі підготовки студентів – майбутніх інженерів-програмістів;

– провести аналіз психолого-педагогічної і методичної літератури з метою визначення ступеня розробленості проблеми, що досліджується;

– визначити найбільш раціональні методи, форми та засоби навчання основ програмування в системі підготовки майбутніх інженерів-програмістів.

У ході констатуючого етапу педагогічного експерименту застосовувалися пасивні методи дослідження. Студенти, які брали участь в експерименті, вивчали курс з основ програмування згідно навчальної програми, яка відповідала вимогам тимчасових стандартів вищої освіти. Ніяких додаткових завдань, крім тих, що було заплановано навчальною програмою, студентам не ставилося. Крім того, для отримання деяких даних було обрано метод анкетування, який забезпечив отримання даних, які надали можливість адекватно і обґрунтовано описати стан проблеми, що досліджується [108].

Анкета була розроблена для студентів першого курсу, які навчалися на спеціальності “Програмне забезпечення автоматизованих систем”. У ході експерименту було опрацьовано 82 анкети.

В анкеті серед інших питань були такі питання:

1. *Чи задовольняє Ваш рівень підготовки зі шкільної інформатики вимогам вищої школи?*

так	
ні	

2. *Чи займалися Ви у школі інформатикою додатково:*

так	
ні	

3. *Чи вистачає підручників та методичних посібників з фундаментальних і професійно-орієнтованих дисциплін, виданих українською мовою?*

так	
ні	

Результати анкетування наведені у табл. 2.8.

Таблиця 2.8

Результати анкетування студентів перших курсів спеціальності “Програмне забезпечення автоматизованих систем”

Питання	Кількість респондентів, що підтвердили об’єктивність зазначеної проблеми	Частота відповідей, що підтверджують об’єктивність проблеми
1	71	0,866
2	34	0,415
3	73	0,890

Для статистичної обробки результатів анкетування було використано загальні положення теорії статистичного опрацювання даних в педагогічних дослідженнях ([74], [112]).

Долю спостережень, у яких відхилення експериментальної величини від теоретичної не перевищує погрішності ε, позначають числом β. У педагогічних дослідженнях можна покласти $\beta = 0{,}95$.

Тоді функція Лапласа $\Phi_0 = \frac{\beta}{2} = 0{,}475$. За допомогою таблиць [112] знаходимо $x_\beta = 1{,}96$.

Величина n, кількість респондентів, які брали участь у анкетуванні, дорівнює 82.

Знаходимо значення погрішності ε за формулою $\varepsilon \approx \frac{x_\beta}{2 \cdot \sqrt{n}}$:

$$\varepsilon \approx \frac{1{,}96}{2 \cdot \sqrt{82}} \approx 0{,}109.$$

Звідси, ймовірність ρ, що лежить у межах довірчого інтервалу, дорівнює:

для 1 питання: ρ ≈ 0, 866 ± 0,109, тобто ρ ∈(0,757; 0,975);

для 2 питання: ρ ≈ 0,415 ± 0,109, тобто ρ ∈(0, 306; 0, 524);

для 3 питання: $\rho \approx 0{,}890 \pm 0{,}109$, тобто $\rho \in (0,\ 781;\ 0{,}999)$.

Таким чином, результати відповідей на перше питання можна вважати достовірними. Тобто шкільний рівень знань з інформатики не відповідає вимогам вищої школи. Тому було зроблено висновок про те, що потрібен пропедевтичний курс, метою якого повинно бути підняття загального рівня знань, умінь і навичок першокурсників з інформатики.

Результати відповідей на друге запитання не можна вважати достовірними або недостовірними.

Відповіді на третє питання підтверджують нашу думку про те, що для навчання основ програмування сьогодні недостатньо підручників та навчально-методичної літератури з дисципліни, виданих українською мовою.

Для визначення орієнтовного рівня знань, умінь і навичок студентів зі шкільного курсу інформатики ми скористалися результатами комп'ютерного тестування 146 абітурієнтів під час вступних випробувань на спеціальність "Програмне забезпечення автоматизованих систем". Результати тестування наведені в діаграмі (рис. 2.12).

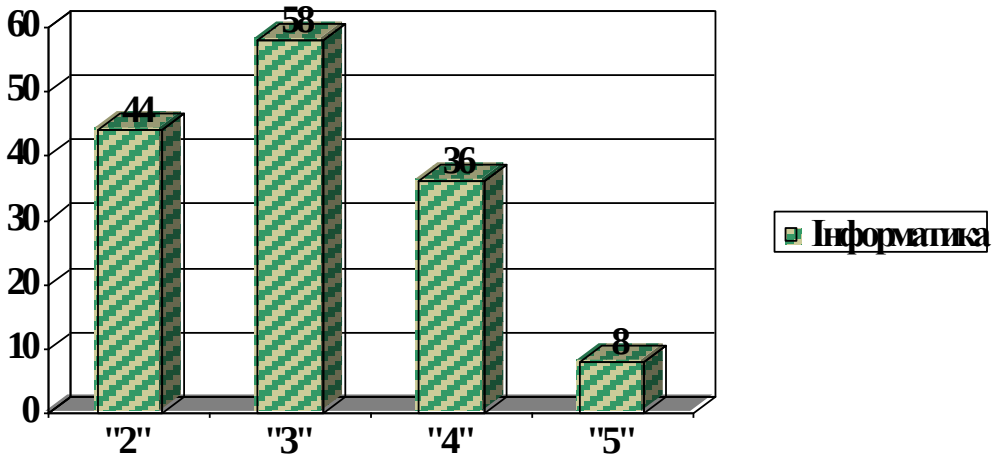


Рис. 2.12. Діаграма з результатами тестування абітурієнтів з шкільного курсу інформатики

Для порівняння наведемо результати тестування тих самих абітурієнтів зі шкільного курсу математики (рис. 2.13).

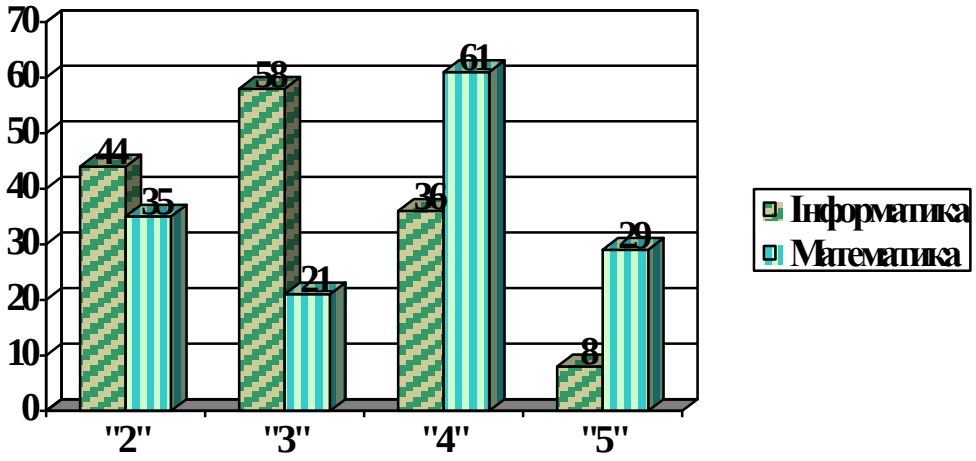


Рис.2.13. Діаграма с результатами тестування зі шкільного курсу інформатики та математики

Отже, рівень знань абітурієнтів з шкільного курсу інформатики значно гірше в порівнянні з рівнем знань з шкільного курсу математики.

На *пошуковому етапі* педагогічного експерименту (2002-2004 р.р.) :

– проводився аналіз проектів стандартів вищої освіти, навчальних планів і навчальних програм з курсу основ програмування;

– проводився аналіз існуючих навчальних посібників з основ програмування для вищої школи;

– визначався зміст теоретичної частини курсу з основ програмування, виходячи із цілей навчання у вищих навчальних закладах відповідних напрямів підготовки;

– визначався зміст лабораторного практикуму з основ програмування в системі підготовки студентів, майбутніх програмістів;

– визначались методи, засоби та форми організації навчання, які було покладено в основу методичної системи, що пропонується;

– проводився пошук і методичний аналіз педагогічного програмного забезпечення для підтримки вивчення теоретичного матеріалу та для підтримки лабораторного практикуму з основ програмування;

– проводився пошук шляхів підвищення рівня навчально-пізнавальної діяльності студентів під час навчання основ інформатики;

– проводився пошук способів організації навчальної діяльності, спрямованих на підвищення практичної значущості результатів навчання з основ програмування.

Результатами пошукового етапу педагогічного експерименту можна вважати наступне:

– була уточнена гіпотеза дисертаційного дослідження;

– створено компоненти методичної системи навчання основ програмування, зокрема розроблено навчально-методичний посібник „Початки програмування” для студентів 1-го та 2-го курсу з основ програмування спеціальності “Програмне забезпечення автоматизованих систем” та спеціальності “Прикладна математика” [57];

– розроблено ППЗ «Linears Rolls», «SCALI», «Tree», електронний посібник „Мова програмування С”;

– розроблено план проведення формуючого етапу педагогічного експерименту.

2.7.2 *Аналіз результатів формувального етапу педагогічного експерименту.* У формувальному етапі педагогічного експерименту (2004-2008 р.р.) взяли участь студенти Черкаського національного університету ім. Богдана Хмельницького, які навчалися на спеціальності “Програмне забезпечення автоматизованих систем” та спеціальності “Прикладна математика”, Черкаського філіалу Європейського університету, які навчалися на спеціальності «Інформаційні управляючи системи і технології»,

Полтавського державного педагогічного університету імені В.Г. Короленка, які навчалися на спеціальності «Інформатика».

Всього у формувальному етапі педагогічного експерименту взяли участь 184 студента, з них у контрольний групі було 90 студентів, у експериментальній групі – 94 студента.

Мета формувального етапу педагогічного експерименту полягала в перевірці ефективності компонентів розробленої методичної системи навчання основ програмування майбутніх інженерів-програмістів.

Формування контрольної та експериментальної груп здійснювалося на основі результатів попереднього тестування з шкільного курсу інформатики таким чином, щоб забезпечити статистичну відповідність рівня знань студентів контрольної та експериментальної груп.

Ефективність компонентів методичної системи навчання в обох групах студентів визначалась за результатами підсумкової контрольної роботи, яка проводилася після вивчення курсу. Контрольна робота складалася з п'яти практичних завдань та одного теоретичного завдання. За кожне виконане завдання можна було одержати до п'яті балів. Отримані бали додавались. Оцінки за контрольну роботу виставлялися відповідно до наступної схеми переведення набраної кількості балів у оцінку за чотирибальною шкалою:

0% - 34% - “незадовільно”;

35% - 59% - “задовільно”;

60% - 85% - “добре”;

86% - 100% - “відмінно”.

З метою досягнення більш об’єктивних результатів статистичне опрацювання експериментальних результатів проводилось за допомогою критерію φ*-кутового перетворення Фішера у поєднанні з критерієм λ Колмогорова-Смірнова ([137], [181]). Опис розрахунків наведено у додатку И.

Таблиця 2.9

Розподіл студентів експериментальної і контрольної груп за рівнями знань основ програмування після підсумкової контрольної роботи

п/п	Успішність	Емпіричні частоти		
		експ. група	конт. група	Суми
1	„5”	18	11	29
2	„4”	36	28	64
3	„3”	27	32	59
4	„2”	13	19	32
		94	90	184

Визначимо точку максимальної розбіжності розподілів результатів підсумкової контрольної роботи (табл. 2.10).

Таблиця 2.10

Розрахунок максимальної різниці накопичених частостей у розподілах результатів підсумковій контрольній роботі студентів експериментальної і контрольної груп

Бали	Емпіричні частоти		Емпіричні відносні частоти		Накопичені емпіричні відносні частоти		Різниця
	експер. гр.	контр. гр.	експер. гр.	контр. гр.	експер. гр.	контр. гр.	
„5”	18	11	0,1915	0,1222	0,1915	0,1222	0,0693
„4”	36	28	0,383	0,3111	0,5745	0,4333	0,1412
„3”	27	32	0,2872	0,3556	0,8617	0,7889	0,0728
„2”	13	19	0,1383	0,2111	1,0000	1,0000	0

Максимальна різниця, яка виявлена між двома накопиченими емпіричними частотами, складає 0,141. Використовуємо верхню границю даної

категорії як критерій для розподілу обох вибірок на групу, де є „ефект” і групу, де „немає ефекту”. Для цього об'єднаємо категорії результатів з оцінками „5” і „4”, з одного боку, і категорії результатів з оцінками „3” і „2” – з іншого. Отриманий розподіл результатів представлений у таблиці 2.11.

Таблиця 2.11

Розподіл результатів в експериментальній групі і контрольній групі

Розподіл на групи		Емпіричні частоти вибору певної категорії результатів		Всього
		експ. гр.	контр. гр.	
1	„5”, „4”	54	39	93
2	„3”, „2”	40	51	91
	Всього:	94	90	184

Задача полягає в тому, щоб з'ясувати чи є закономірним те, що результати підсумкової контрольної роботи у студентів експериментальної групи відрізняються від результатів студентів контрольної групи. Відповідно до алгоритму розрахунку, необхідно повернути таблицю 2.11 на 90^0, обертаючи її за напрямом годинникової стрілки. Одержуємо робочу таблицю для розрахунку критерію φ* - кутового перетворення Фішера (табл. 2.12).

Таблиця 2.12

Виявлення відмінностей в експериментальній групі і контрольній групі

Групи	Є ефект	Немає ефекту	Всього:
експ. гр.	40(42,6%)	54 (57,4%)	94
контр. гр.	51(56,7%)	39(43,3%)	90
Всього:	91	93	184

Сформулюємо гіпотези:

H_0: Рівень знань з основ програмування у студентів експериментальної групи не вище ніж у студентів контрольної групи.

H_1: Рівень знань з основ програмування у студентів експериментальної групи вище, ніж у студентів контрольної групи.

Визначимо величини φ1 і φ2 [181, с.330 (Таблиця XII Додатку 1)]:

φ1(42,6%)=1,422;

φ2(56,7%)=1,705.

Тепер визначимо емпіричне значення критерію φ*:

$$\varphi^* = (1{,}705 - 1{,}422) * \sqrt{\frac{94*90}{184}} = 1{,}919.$$

Ця величина відповідає рівню значущості ρ=0,027 [181, с.332 (Таблиця XIII Додатку 1)].

Критичні значення критерію φ* [181, с.332 (Таблиця XIII Додатку 1)]:

1,64 (p<=0,05);

2,31 (p<=0,01).

З одержаних значень маємо, що гіпотеза H_0 не приймається (ρ=0,027). Підтверджується гіпотеза H_1, тобто емпіричні розподіли в експериментальній групі й контрольній групі розрізняються: рівень знань з основ програмування у студентів експериментальної групи вище, ніж у студентів контрольної групи.

Для наочності побудуємо вісь значущості (рис. 2.14):

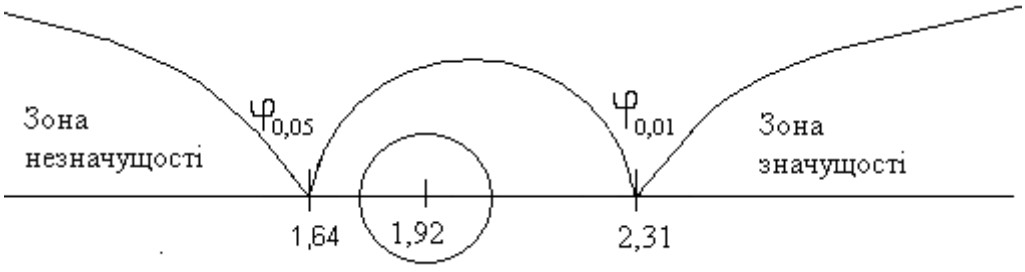


Рис. 2.14. Вісь значущості.

Проаналізуємо також результати сесійного контролю знань студентів експериментальної групи і контрольної групи з дисциплін циклу комп'ютерних

наук: „Об'єктно-орієнтоване програмування”, „Інженерна і комп'ютерна графіка”, „Технологія програмування та створення програмних продуктів”, „Програмування на асемблері”, „Теорія алгоритмів та обчислювальних процесів”. На другому і третьому курсах навчання, відповідно до навчальних планів, ці студенти вивчали вказані дисципліни і складали з них екзамени.

На підставі розрахунків (додаток К), які аналогічні попереднім, узявши за початкові – значення оцінок з екзаменів з вказаних дисциплін, можна зробити висновок про те, що студенти експериментальної групи успішніше засвоїли навчальний матеріал дисциплін з циклу комп'ютерних наук у порівнянні зі студентами контрольної групи.

На підставі отриманих результатів можна зробити висновок про те, що результатом формувального етапу педагогічного експерименту є підтвердження гіпотези про те, що впровадження в навчальний процес розроблених компонентів методичної системи навчання на основі новітніх педагогічних та інформаційно-комунікаційних технологій надає можливість інтенсифікувати процес навчання, активізувати пізнавальну діяльність студентів, підняти межу базового рівня знань студентів з основ програмування, диференціювати й індивідуалізувати процес навчання з огляду на різний рівень здібностей студентів, підвищити рівень підготовленості студентів до вивчення професійно-орієнтованих дисциплін, підвищити рівень їх професійної підготовки.

Вірогідність одержаних висновків визначається тривалістю експерименту; кількістю учасників експерименту; репрезентативністю вибірки; інтерпретацією результатів, що отримані, за допомогою методів математичної статистики.

Висновки до розділу 2

1. Зміст навчання студентів за спеціальностями „Прикладна математика” і „Програмне забезпечення автоматизованих систем”, що передбачають надання випускникам кваліфікації інженер-програміст, доцільно доповнити

пропедевтичним курсом, мета якого – підвищення у студентів стартового рівня знань, умінь і навичок з шкільного курсу алгоритмізації і програмування перед навчанням основ програмування.

2. Зміщення акценту в процесі навчання мов програмування з вивчення синтаксичних та семантичних особливостей на вивчення методів побудови й аналізу алгоритмів надає можливість підвищити рівень підготовки студентів до вивчення дисциплін з циклу комп'ютерних наук на старших курсах.

3. Використання компонентів розробленої автором методичної системи навчання основ програмування сприяє диференціації навчальної діяльності студентів та підвищенню їхньої пізнавальної активності.

4. Зміст курсу з основ програмування повинний складатися з інваріантної та варіативної частин. При цьому інваріантна частина змісту курсу повинна бути однаковою для всіх напрямів підготовки з галузі інформатики та обчислювальної техники, а зміст варіативної частини залежить від конкретної спеціальності.

5. Ретельно продуманий за змістом і послідовністю широкий спектр різноманітних видів і типів навчальних завдань з основ програмування різного рівня складності надає можливість інтенсифікувати процес навчання, сприяє його високій результативності при підготовці майбутніх інженерів-програмістів.

6. Використання задачного підходу до навчання основ програмування надає можливість зосередити увагу студентів на розв’язуванні задач і „знімає бар'єр” для вивчення нових мов програмування, що важливо для інженера-програміста.

7. Застосування такого методу навчання як розробка колективного проекту сприяє формуванню і розвиткові у майбутніх інженерів-програмістів професійних якостей, які властиві фахівцам у галузі комп’ютерних наук.

8. Впровадження методичної системи навчання основ програмування, що ґрунтується на широкому впровадженні в навчальний процес новітніх

педагогічних та інформаційно-комунікаційних технологій створює не лише реальні умови для розширення й поглиблення змісту основ програмування, а й сприяє інтенсифікації процесу навчання, його результативності, інтелектуальному розвитку студентів, формуванню конкурентноспроможних фахівців в галузі інформатики та інформаційних технологій.

9. Проведений педагогічний експеримент показав, що рівень знань, умінь і навичок з основ програмування у студентів, які сформовані в процесі навчання за розробленою методичною системою, вище в порівнянні зі студентами, що навчалися за традиційною системою.

Основні результати другого розділу опубліковано у роботах [52; 53], [55;56], [58; 59], [61; 62], [64].

ВИСНОВКИ

Відповідно до мети та поставлених завдань у процесі дослідження отримані такі основні **результати**:

– обґрунтовано доцільність створення і використання методичної системи навчання основ програмування майбутніх інженерів-програмістів, яка орієнтована на широке використання в навчальному процесі новітніх педагогічних та інформаційно-комунікаційних технологій;

– визначено роль і місце курсу основ програмування в системі підготовки фахівців у галузі комп'ютерних наук;

– розкрито механізми, зокрема і психологічні, формування професійних умінь майбутніх програмістів;

– визначено вимоги до професійних якостей майбутніх інженерів-програмістів;

– визначено сучасні принципи і підходи до навчання основ програмування;

– уточнено мету, завдання і функції навчання курсу з основ програмування;

– розроблено основні компоненти (мету, зміст, методи, засоби, організаційні форми навчання) методичної системи навчання основ програмування майбутніх інженерів-програмістів;

– розроблено методику організації і проведення самостійної роботи студентів із використанням сучасних інформаційно-комунікаційних технологій навчання;

– визначено умови та можливі шляхи реалізації диференційованого підходу при навчанні основ програмування;

– розроблено методичне забезпечення навчального курсу з основ програмування;

– розроблено та обґрунтовано методику використання методів активного навчання, зокрема, розробка навчального програмного продукту колективом студентів з 2-3 осіб;

– розроблено педагогічні програмні засоби для підтримки курсу з основ програмування, зокрема, розроблено мультимедійні педагогічні програмні засоби для підтримки курсу основ програмування;

– розроблено класифікацію типів завдань з основ програмування;

– проведено аналіз існуючої навчальної літератури з курсу основ програмування;

–створено ефективну систему контролю знань, умінь і навичок студентів з основ програмування на основі модульно-рейтингової системи навчання.

Результати проведеного дослідження дають підстави зробити такі **висновки:**

1. Існуючі тенденції розвитку інформаційного суспільства потребують на ринку інтелектуальної праці фахівця у галузі комп’ютерних наук, підготовка якого повинна здійснюватися з врахуванням міжнародних стандартів і сучасних вимог до фахівців у галузі інформаційно-комунікаційних технологій і комп’ютерних наук.

2. Знання, вміння і навички, здобуті студентами у навчальному курсі з основ програмування, потрібні їм при вивченні дисциплін циклу фундаментальних, професіонально-орієнтованих дисциплін і дисциплін спеціалізації, що визначає його роль у цьому процесі.

3. Для формування у студентів знань, умінь і навичок, які відповідають сучасному рівню розвитку комп'ютерних наук і комп'ютерної техніки, в навчальному процесі важливо дотримуватися принципів систематичності і послідовності навчання, свідомості, активності і самостійності студентів у навчанні при керівній ролі викладача.

4. З метою поліпшення якості навчання зміст курсу з основ програмування потрібно узгодити з рекомендаціями, що запропоновані у документах „Computing Curricula 2001” та „Software Engineering 2004” з врахуванням освітньо-професійних програм державних стандартів з відповідних напрямів підготовки.

5. Доцільно здійснювати розподіл змісту курсу з основ програмування на дві частини – інваріантну, однакову для всіх напрямів підготовки в галузі інформатики та обчислювальної техніки, та варіативну, зміст якої залежить від конкретної спеціальності.

6. З метою забезпечення у студентів необхідного стартового рівня знань, умінь і навичок з алгоритмізації і програмування у ВНЗ необхідно впроваджувати пропедевтичний курс, основою якого є навчальна мова програмування Pascal.

7. З міркувань перспективності застосування програмування у професійній діяльності в курсі з основ програмування треба вчити студентів програмувати мовою програмування С.

8. У процесі навчання основ програмування необхідно змістити акцент з вивчення синтаксичних та семантичних особливостей мов програмування на:

– вивчення методів побудови й аналізу алгоритмів;

– використання задачного підходу до навчання основ програмування на базі широкого спектру різноманітних видів і типів навчальних завдань з основ програмування різного рівня складності.

9. Для формування професійних якостей фахівця в галузі комп'ютерних наук треба удосконалювати традиційні методи навчання, а також запроваджувати методи інтерактивного і активного навчання, зокрема, створення студентами навчального програмного продукту.

10. Засоби навчання (посібники, підручники, методичні розробки, ППЗ, технічні засоби навчання, інформаційно-комунікаційні технології) повинні разом з іншими компонентами методичної системи складати єдиний комплекс, що забезпечує вирішення задачі навчання основ програмування.

11. Комп'ютерно-орієнтовані лекційні та лабораторні заняття, індивідуальна і диференційована форми організації навчання, цілеспрямована організація самостійної роботи студентів у позааудиторний час у комп'ютерному класі або з використанням технологій дистанційного навчання,

самостійна колективна робота студентів з 2-3 осіб над єдиною спільною задачею (програмним продуктом) протягом семестру під керівництвом викладача, модульно-рейтингова система контролю і оцінювання знань студентів надають можливість суттєво підвищити результати навчання основ програмування.

12.Проведений педагогічний експеримент показав на практиці ефективність компонентів розробленої методичної системи навчання основ програмування майбутніх інженерів-програмістів.

13.Впровадження в навчальний процес розробленої методичної системи навчання на основі сучасних педагогічних та інформаційно-комунікаційних технологій надає можливість інтенсифікувати процес навчання, суттєво активізувати пізнавальну діяльність студентів, підвищити рівень базових знань студентів з основ програмування, диференціювати та індивідуалізувати процес навчання з огляду на різний рівень здібностей студентів, підвищити рівень підготовленості студентів до вивчення професійно-орієнтованих дисциплін, підвищити рівень їх професійної підготовки.

Запропоновані у дисертації принципи і підходи щодо створення методичної системи навчання основ програмування планується використати при розробці методичних систем навчання дисциплін циклу комп’ютерних наук, зокрема, з використанням технологій дистанційного і мобільного навчання.

Подальших досліджень вимагають питання індивідуалізації процесу навчання основ програмування.

ДОДАТОК А

НАВЧАЛЬНА ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ „ПРОГРАМУВАННЯ”

Таблиця А.1

ОПИС ПРЕДМЕТА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Курс: 1-2 **Підготовка бакалаврів**	**Напрям, спеціальність, освітньо-кваліфікаційний рівень**	**Характеристика навчального курсу**
Кількість кредитів - національних – **8** - ECTS – **12** Модулів: **5** Змістових модулів: **8** Загальна кількість годин: **432** Тижневих годин: 4-4-4	Шифр та назва напряму: **0403 Системні науки та кібернетика** Шифр та назва спеціальності: **6.040301 Прикладна математика** Освітньо-кваліфікаційний рівень: **бакалавр**	*Обов’язковий* Рік підготовки: **1**, семестр **1, 2**; **2**, семестр **1** Лекції: **108** годин Лабораторні заняття: **108** годин Самостійна робота: **216** годин Форма підсумкового контролю: 1, 3 семестри **– екзамен** 2 семестр **– залік**

Мета курсу „Програмування” – формування особистості студента; розвиток його інтелекту, здатності до алгоритмічного мислення; формування розуміння зв'язку теорії програмування із практикою програмування, розуміння галузі застосування здобутих знань з основ програмування в інших професійно-орієнтованих і спеціальних дисциплінах та в сфері майбутньої професійної діяльності.

Завдання вивчення основ програмування полягає у тому, щоб у систематизованій формі навчити студентів основам алгоритмізації та програмування, методам побудови алгоритмів, методам аналізу алгоритмів;

сформувати у студентів практичні навички зі створення правильних, ефективних за часом виконання та розміру коду, програм мовами програмування Pascal та C із використанням структурної методології; ознайомити з парадигмами програмування, з етапами проектування та створення програмних комплексів.

Комплекс основних знань, що формуються в процесі навчання курсу

Студент має знати:

−історію розвитку мов програмування;

−галузі застосування програмування;

−системи числення;

−основні принципи організації комп'ютерів;

−введення в комп’ютерну логіку;

−методологію побудови інформаційних моделей;

−критерії оцінювання мов програмування;

−спрощену модель компілятора;

−правила подання даних у пам'яті комп’ютера;

−властивості алгоритмів;

−способи подання алгоритмів;

−структурну методологію створення програм;

−модульний принцип створення програм;

−етапи розроблення програм на ПК: редагування, трансляція, компонування;

−методи структурного низхідного (спадного) програмування;

−структуру професійних середовищ створення програм мовами Pascal та C;

−алфавіт, лексику і структуру програми на мовах Pascal та C;

−синтаксис та семантику мов програмування Pascal та C;

−класифікацію структур даних;

−скалярні типи даних;

−структуровані типи даних (рядки, масиви, асоціативні масиви, записи, структури, об'єднання, множини, файли, списки, хеш-таблиці, двійкові дерева пошуку) та алгоритми їх обробки;

−синтаксичні діаграми опису типів;

−типові алгоритмічні конструкції: слідування, розгалуження, повторення;

−класифікацію операцій (арифметичні операції, операції відношення, логічні операції, бітові операції та операції зсуву, операції над рядками, операції над множинами);

−стандартні алгоритми обробки даних;

−типи підпрограм мови програмування Pascal;

−бібліотечні підпрограми мови Pascal;

−графічні засоби мови Pascal;

−механізми використання функцій у мові C;

–бібліотечні функції мови С;
–механізм передавання параметрів-значень та параметрів-змінних;
–форми рекурсивних алгоритмів;
–модульний принцип розробки програм;
–методи динамічного розподілу пам’яті;
–класи пам'яті, компонування і керування пам'яттю у мові С;
–файлове введення-виведення у мові С;
–поразрядні операції у мові С;
–засоби препроцесорної обробки програми у мові С;
–методи побудови алгоритмів;
–методи оцінювання швидкодії алгоритмів;
–методи аналізу алгоритмів;
–парадигми програмування;
–основи технологій проектування програмних комплексів.

Комплекс основних умінь, що формуються в процесі навчання курсу:

Студент повинен уміти:
–формалізувати поставлену задачу та будувати інформаційну модель цієї задачі;
–визначати структури даних для розв'язування даної задачі;
–розробляти алгоритми методом покрокового уточнення;
–будувати декілька різних алгоритмів для розв'язування однієї задачі;
–подавати алгоритми в різних формах;
–складати лінійні програми, програми з розгалуженням, циклічні програми мовами Pascal та С;
–використовувати скалярні типи даних: цілі, дійсні, символьні, логічні;
–налагоджувати програму в середовищі програмування мов Pascal та С;
–користуватися довідковою системою середовища програмування;
–підключати процедури та функції стандартної бібліотеки мови Pascal;
–підключати функції стандартної бібліотеки мови С;
–використовувати структуровані типи даних при складанні програм;
–ділити складну задачу на підзадачі і задати їхнє комбінування;
–використовувати власні процедури та функції при складанні програм мовою Pascal;
–використовувати власні функції при складанні програм мовою С;
–оцінювати швидкодію алгоритмів;
–будувати алгоритми з використанням мінімального обсягу пам'яті компютера;
–застосовувати і комбінувати відомі прийоми програмування, типові і стандартні алгоритми;
–застосовувати методи побудови алгоритмів;
–реалізувати багатомодульні програми;

– складати і використовувати макрозасоби та засоби препроцесорної обробки програм у мові С;

а також:

– бачити задачу одночасно на різних рівнях деталізації;

– писати програми, які правильно розв'язують задачу, і добре документовані;

– аналізувати і виправляти помилки у самостійно розроблених алгоритмах і програмах;

– правильно і однозначно висловлювати думку в зрозумілій співбесіднику формі;

– розуміти готові програми;

– співпрацювати з членами колективу при розробці спільного програмного продукту.

Модуль1. ***Вступ до програмування***

Змістовий модуль 1. **Програмування як наука**

Тема 1.1. Галузі застосування програмування

1. Наукові дослідження.
2. Комерційна діяльність.
3. Штучний інтелект.
4. Системне програмування.
5. Мови підготовки сценаріїв.
6. Спеціалізовані мови програмування.

Тема 1.2. Роль мов програмування

1. Парадигми мови.
2. Стандартизація мови.
3. Інтернаціоналізація.

Тема 1.3. Методи реалізації мови програмування

1. Компіляція.
2. Інтерпретація.
3. Змішана реалізація.

Тема 1.4. Середовище мови програмування

1. Вплив на розробку мови програмування.
2. Середовище розробки.
3. Мови керування завданнями і мови створення процесів.

Тема 1.5. Алгоритм і його властивості

1. Різні підходи до поняття "алгоритм".
2. Поняття виконавця алгоритму.
3. Подання алгоритму.
4. Властивості алгоритмів.
5. Поняття алгоритмічної мови.

Змістовий модуль 2. **Основи алгоритмізації і програмування. Навчальна мова програмування Pascal**

Тема 2.1. Алфавіт, лексика, структура програм

1. Текст програми. Алфавіт мови.
2. Лексична структура мови.
3. Загальна структура Pascal-програми.

Тема 2.2. Система типів даних мови. Скалярні типи

1. Ієрархія типів.
2. Група цілочисельних типів.
3. Група дійсних типів.
4. Символьний тип.
5. Булевський тип.
6. Обмежені типи.
7. Перелічені типи.

Тема 2.3. Опис змінних, констант і типів

1. Опис змінних.
2. Опис констант.
3. Опис типів даних.
4. Типізовані константи.

Тема 2.4. Перетворення типів даних. Сумісність типів

1. Неявні перетворення.
2. Використання стандартних функцій.
3. Явні перетворення.
4. Еквівалентність типів.
5. Сумісність типів.

Тема 2.5. Оператори

1. Оператор присвоювання.
2. Оператор процедури.
3. Оператор переходу.
4. Складений оператор.
5. Умовний оператор.
6. Оператор варіанту.
7. Оператор циклу із передумовою.
8. Оператор циклу з післяумовою.
9. Оператор циклу з параметром.
10. Порожній оператор.

Тема 2.6. Структуровані типи даних

1. Числові масиви.
2. Символьні масиви.
3. Типові помилки в роботі з масивами.
4. Записи і селектори.
5. Записи з варіантами.
6. Рядкові типи.
7. Стандартні процедури і функції роботи з рядками.

8. Множини.

Тема 2.7. Процедури і функції

1. Загальна структура підпрограм.
2. Тіло підпрограми.
3. Області дії імен.
4. Механізм передавання параметрів.
5. Параметри-значення.
6. Параметри-змінні.
7. Безтипові параметри.
8. Обчислення значення функції. Завершення підпрограм.
9. Попередні і зовнішні описи підпрограм.
10. Рекурсія та її побічний ефект.
11. Розподіл пам'яті для локальних змінних.

Тема 2.8. Покажчики. Динамічні змінні

1. Посилальні типи і покажчики.
2. Доступ до змінної через покажчик.
3. Статичні і динамічні змінні.
4. Створення і знищення динамічних змінних.
5. Проблема загублених посилань.
6. Установка розмірів динамічної пам'яті.
7. Сумісність і перетворення посилальних типів.

Тема 2.9. Файлові типи

1. Файлові змінні і типи.
2. Операції над файлами.
3. Операції введення-виведення.
4. Переміщення по файлу.
5. Обробка помилок введення-виведення.
6. Текстові файли.
7. Файли без типу.

Тема 2.10. Модулі

1. Загальна структура модуля.
2. Підпрограми в модулях.
3. Компіляція і використання модулів.
4. Стандартні модулі.

Модуль 2. *Мова програмування С*

Змістовий модуль 3. **Основи мови програмування С**

Тема 3.1. Вступ до мови С. Подання даних у мові С

1. Приклад програми на С. Документування. .
2. Попередні відомості про налагодження. Синтаксичні помилки. Семантичні помилки. Стан програми.
3. Ключові слова і зарезервовані ідентифікатори
4. Базові типи мови С. Цілочисельні типи. Символьний тип. Тип `_Bool`. Дані типу `float, double` і `longdouble`. Комплексні і мнимі типи.

5. Використання типів даних.
6. Аргументи і помилки при їхньому використанні.

Тема 3.2. Символьні рядки і форматоване введення-виведення

1. Рядки символів: введення. Масив значень типу `char` і нульовий символ. Використання рядків. Функція `strlen()`.
2. Константи і препроцесор С. Модифікатор `const`. Робота із символічними константами.
3. Використання функцій `printf()` і `scanf()`. Модифікатори специфікації перетворення для функції `printf()`. Модифікатор * і його використання у функціях `printf()` і `scanf()`.

Тема 3.3. Операції, вирази і оператори

1. Фундаментальні операції: присвоювання, арифметичні операції. Пріоритети операцій.
2. Деякі додаткові операції: операція `sizeof` і тип `size_t`, операція ділення за модулем %, операції інкременту і декременту. Пріоритети операцій.
3. Вирази і оператори. Складені оператори.
4. Перетворення типів.
5. Операція приведення.
6. Функції з аргументами.

Тема 3.4. Оператори організації циклу

1. Оператор `while`.
2. Цикл `for`. Операція коми.
3. Додаткові операції присвоювання: `+=, -=, *=, %=`.
4. Цикл `do while`.
5. Вкладені цикли.

Тема 3.5. Керуючі оператори: розгалуження і безумовні переходи

1. Оператор `if`. Конструкція `else` в операторі `if`. Кількість вкладень операторів `if`.
2. Умовна операція `?:`
3. Додаткові засоби організації циклу: `continue і break`.
4. Оператори `switch` і `break`. Використання оператора `switch`.
5. Оператор `goto`.

Тема 3.6. Символьне введення-виведення і верифікація введення

1. Односимвольні функції введення-виведення: `getchar()` і `putchar()`.
2. Буфер клавіатури.
3. Завершення введення з клавіатури. Файли, потоки і введення даних із клавіатури. Кінець файлу.
4. Перенаправлення і файли.
5. Створення дружнього користувальницького інтерфейсу. Робота з буферизованим введенням. Змішування чисельного і символьного введення.
6. Перевірка допустимості введення. Аналіз програми. Потік введення і числа.
7. Перегляд меню. Забезпечення стійкого виконання програм. Змішування символьного і чисельного введення.

Тема 3.7. Функції

1. Огляд функцій. Створення і використання простої функції. Аргументи функції. Визначення функції з аргументами: формальні параметри. Створення прототипу функції з аргументами. Виклик функції з аргументом: фактичні параметри. Повернення значення функцією за допомогою оператора `return`.
2. Прототипування функцій у стандарті ANSI C. Відсутність аргументів і невизначені аргументи.
3. Рекурсія. Основи рекурсії. Хвостова рекурсія. Рекурсія і зворотний порядок. Переваги і недоліки рекурсії.
4. Компіляція програм із двох або більшого числа вихідних файлів. Використання заголовних файлів.
5. Пошук адрес: операція &.
6. Зміна змінних у функції, що викликає іншу функцію.
7. Покажчики. Операція розіменування: *. Оголошення покажчиків. Використання покажчиків для обміну даними між функціями.

Тема 3.8. Масиви і покажчики

1. Масиви. Ініціалізація. Присвоювання значень масивам. Границі масиву. Визначення розміру масиву.
2. Багатомірні масиви. Ініціалізація двовимірного масиву. Масиви з розмірністю більше двох.
3. Покажчики і масиви.
4. Функції, масиви і покажчики. Використання параметрів типу покажчик.
5. Операції з покажчиками.
6. Захист змісту масиву. Використання ключового слова `const` з формальними параметрами. Додаткові відомості про ключове слово `const.`
7. Покажчики і багатовимірні масиви. Сумісність покажчиків. Функції і багатовимірні масиви.
8. Масиви змінної довжини.

Тема 3.9. Символьні рядки і рядкові функції

1. Визначення рядків у програмі. Покажчики і рядки.
2. Введення рядків. Виділення простору пам'яті під рядки. Функції `gets(), fgets(), scanf().`
3. Виведення рядків. Функції `puts(), fputs(), printf().`
4. Можливість створення власних функцій.
5. Рядкові функції `strlen(), strcat(), strncat(), strcmp().` Повернення значення функції `strcmp().` Варіанти функції `strncmp().` Функції `strcpy()` і `strncpu().` Функція `sprintf().` Інші рядкові функції.
6. Приклад обробки рядків. Сортування покажчиків замість рядків. Вибір алгоритму сортування.
7. Символьні функції заготовочного файлу `ctype.h` і рядки.
8. Аргументи командного рядка.
9. Перетворення рядків у числа.

Тема 3.10. Класи пам’яті, компоновка і керування пам'яттю

1. Класи пам'яті. Область видимості. Зв'язування. Тривалість зберігання. Автоматичні змінні. Регістрові змінні. Статичні змінні з областю видимості у межах блоку. Статичні змінні із зовнішнім зв’язуванням.
2. Специфікатори класів пам'яті.
3. Класи пам'яті і функції.
4. Функція генерації випадкових величин і статична змінна.
5. Розподіл пам'яті: функції `malloc()` і `free()`. Розподіл динамічної пам'яті і масиви змінної довжини. Класи пам'яті і динамічний розподіл пам'яті.
6. Класифікатори типів у стандарті ANSI C: `const`, `volatile`, `restrict`.

Тема 3.11. Файлове введення-виведення

1. Обмін даними з файлами. Текстове і двійкове подання файлів. Рівні введення-виведення. Стандартні файли.
2. Стандартне введення-виведення. Перевірка наявності аргументів командного рядка. Функції `fopen()`, `getc()`, `putc()`. Ознака кінця файлу. Функція `fclose()`. Покажчики на стандартні файли. Приклад програми стискання файлів.
3. Функції введення-виведення: `fprintf()`, `fscanf()`, `fgets()`, `fputs()`.
4. Довільний доступ: функції `fseek()`, `ftell()`. Порівняння двійкового і текстового режимів. Переносимість. Функції `fgetpos()` і `fsetpos()`.

Тема 3.12. Структури та інші форми даних

1. Структури. Оголошення змінної типу структури. Ініціалізація структур. Доступ до елементів структури. Виділені ініціалізатори структур.
2. Масиви структур. Оголошення масиву структур. Ідентифікація елементів масиву структур.
3. Вкладені структури.
4. Покажчики на структури. Оголошення та ініціалізація покажчика на структуру. Доступ до елементів структури через покажчики.
5. Взаємодія функцій і структур. Передавання елементів структури. Використання адреси структури. Передача структури в якості аргументу. Аналіз властивостей структур. Символьні масиви або покажчики на символи у структурах. Структура, покажчики і функція `mallc()`. Функції, що використовують масив структур.
6. Збереження вмісту структур у файлі.
7. Об'єднання: коротке знайомство.
8. Перелічені типи. Константи типу `enum`. Значення за замовчуванням. Значення, що привласнюють. Спільно використовувані простори імен.
9. Оператор `typedef`.
10. Фіктивні оголошення.
11. Функції і покажчики.

Тема 3.13. Операції з розрядами

1. Двійкові числа, біти, байти. Двійковий запис цілочисельних значень. Двійкове подання чисел із плаваючою крапкою.
2. Інші основи систем числення. Восьмерична система числення. Шістнадцатирична система числення.
3. Порозрядні операції. Порозрядні логічні операції. Область застосування: маски, включення розрядів, відключення розрядів, перемикання розрядів, перевірка значення розряду. Порозрядні операції зсуву.
4. Розрядні поля. Розрядні поля і порозрядні операції.

Тема 3.14. Препроцесор і бібліотека мови С

1. Іменовані константи: `#define`. Лексеми. Перевизначення констант.
2. Використання аргументів у директиві `#define`. Створення рядків з аргументів макросу: операція #. Засіб об'єднання препроцесора: операція `##.`
3. Включення файлів: директива `#include`. Приклад заголовного файлу. Область застосування заголовних файлів.
4. Інші директиви. Директива `#undef.` Визначеність із погляду препроцесора. Умовна компіляція. Визначені макроси. Директиви `#line, #error, #pragma`.
5. Вбудовані функції.
6. Бібліотека С. Доступ до бібліотеки С. Використання описів бібліотеки.
7. Бібліотека математичних функцій.
8. Бібліотека утиліт загального призначення. Функції `exit(), atexit, qsort().`
9. Бібліотека `assert.`
10. Функції `memcpy()` і `memmove()` з бібліотеки `string.h`
11. Змінні аргументи: файл `stdarg.h`

Модуль 3. Алгоритми: побудова і аналіз

Змістовий модуль 4. **Сортування і порядкові статистики**

Тема 4.1. Сортування за час $\theta(n^2)$.

1. Оцінка часу алгоритмів сортування.
2. «Бульбашкове» сортування.
3. Оптимізоване «бульбашкове» сортування.
4. Сортування вставками.
5. Сортування вибором.

Тема 4.2. Сортування за час $\theta(n*\log_2 n)$.

1. Швидке сортування.
2. Сортування злиттям.
3. Сортування за допомогою купи.
4. Сортування Шелла.

Тема 4.3. Сортування за час $\theta(n)$.

1. Нижні оцінки для сортування.
2. Сортування підрахунком.

3. Цифрове сортування.
4. Сортування вичерпуванням.

Змістовий модуль 5. **Абстрактні типи даних**

Тема 5.1. Елементарні структури даних.

1. Стеки, черги, декі.
2. Зв'язані списки.
3. Реалізація покажчиків і записів з декількома полями.

Тема 5.2. Хеш-таблиці.

1. Пряма адресація.
2. Хеш-таблиці.
3. Хеш-функції.
4. Відкрита адресація.

Тема 5.3. Двійкові дерева пошуку.

1. Властивості двійкового дерева пошуку.
2. Пошук у двійковому дереві.
3. Додавання і вилучення елемента двійкового дерева пошуку.

Змістовий модуль 6. **Методи побудови і аналізу алгоритмів**

Тема 6.1. Швидкість росту функцій

1. Асимптотичні позначення.
2. Стандартні функції і позначення.

Тема 6.2. Алгоритми „Розподіляй і володарюй”

1. Множення довгих цілочислових значень.
2. Баланс підзадач.

Тема 6.3. Динамічне програмування

1. Задача перемножування декількох матриць.
2. Коли застосовується динамічне програмування.
3. Задача про найбільшу загальну підпослідовність послідовності.

Тема 6.4. Жадібні алгоритми

1. Задача про вибір заявок.
2. Коли застосовується жадібний алгоритм.
3. Коди Хаффмена.
4. Теоретичні основи жадібних алгоритмів.
5. Задача про розклад.

Тема 6.5. Пошук з поверненням

1. Функції виграшу.
2. Реалізація пошуку з поверненням.
3. Метод гілок і границь.

Модуль 4. Основи розробки програмного забезпечення

Змістовий модуль 7. **Основи розробки програмного забезпечення**

Тема 7.1. Постановка задачі

1. Характеристика задачі.

2. Вхідні дані.
3. Вихідні дані.
4. Опис алгоритмів розв’язування задачі.

Тема 7.2. Технічне завдання

1. Назва проекту.
2. Мета створення.
3. Характеристика області застосування.
4. Вимоги до проекту.
5. Інформаційна база.
6. Програмне забезпечення.
7. Вимоги до технічного забезпечення.
8. Графік робіт.

Тема 7.3. Створення програмного продукту, впровадження і супровід

1. Концептуальна цілісність системи. Архітектори і розробники.
2. Створення системи. Специфікації.
3. Інсталяція. Ручна установка. Процедура інсталяції. Інсталяція копіюванням. Інсталяція зборкою. Назва програм інсталяції. Висновок попередньої інформації. Інсталяція. Специфікація інсталяції.
4. Впровадження і супровід.

Модуль 5. Парадигми програмування

Змістовий модуль 8. **Огляд парадигм програмування**

Тема 8.1. Мови програмування низького рівня. Мови асемблера

1. Огляд мови програмування `Assembler`.

Тема 8.2. Вступ до функціонального програмування

1. Огляд мови програмування `Lisp`.

Тема 8.3. Вступ до логічного програмування

1. Огляд мови програмування `Prolog`.

Тема 8.4. Вступ до об’єктно-орієнтованого програмування

1. Огляд мови програмування `C++`.

Тема 8.5. Паралельне програмування. Огляд.

1. Паралельне виконання. Синхронізація задач.

Тема 8.6. Internet-програмування. Огляд.

1. Огляд мов програмування під Internet.

Таблиця А.2

СТРУКТУРА ЗАЛІКОВОГО КРЕДИТУ НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

	Кількість годин, Відведених на:				
	Лекції	**Практично – семінарські заняття**	**Лабораторні заняття**	**Самостійну роботу**	**Індивідуальну роботу**
Модуль1. ***Вступ до програмування*** *(1 семестр)*					
Змістовий модуль 1**.** ***Програмування як наука***					
Тема 1.1. *Галузі застосування програмування*	6			4	
Тема 1.2. *Роль мов програмування*					
Тема 1.3. *Методи реалізації мови програмування*					
Тема 1.4. *Середовище мови програмування*					
Тема 1.5. *Алгоритм і його властивості*					
Змістовий модуль 2. ***Основи алгоритмізації і програмування. Навчальна мова програмування Pascal***					
Тема 2.1. *Алфавіт, лексика, структура програм*	4		4	8	
Тема 2.2. *Система типів даних мови. Скалярні типи*					
Тема 2.3. *Опис змінних, констант і типів*					
Тема 2.4. *Перетворення типів даних. Сумісність типів*					
Тема 2.5. *Оператори*	6		10	18	
Тема 2.6. *Структуровані типи даних*	8		1	2	
Тема 2.7. *Процедури і функції.*	6		6	1	
Тема 2.8. *Покажчики. Динамічні змінні*	2		2	4	
Тема 2.9. *Файлові типи*	2		2	4	
Тема 2.10. *Модулі*	2		2	4	
Всього годин за 1-й семестр:	**36**		**36**	**72**	
Модуль 2. ***Мова програмування С*** *(2 семестр)*					
Змістовий модуль 3. ***Основи мови програмування С***					
Тема 3.1. *Вступ до мови С. Подання даних у мові С*	2		2	4	

Тема 3.2. *Символьні рядки і форматоване введення-виведення*	2		2	4	
Тема 3.3. *Операції, вирази і оператори*	2		2	4	
Тема 3.4. *Оператори організації циклу*	4		4	8	
Тема 3.5. *Керуючі оператори: розгалуження і безумовні переходи*	2		2	4	
Тема 3.6. *Символьне введення-виведення і верифікація введення*	2		2	4	
Тема 3.7. *Функції*	2		2	4	
Тема 3.8. *Масиви і покажчики*	6		6	12	
Тема 3.9. *Символьні рядки і рядкові функції*	4		4	8	
Тема 3.10. *Класи пам'яті, компоновка і керування пам'яттю*	2		2	4	
Тема 3.11. *Файлове введення-виведення*	2		2	4	
Тема 3.12. *Структури і інші форми даних*	2		2	4	
Тема 3.13. *Операції з розрядами*	2		2	4	
Тема 3.14. *Препроцесор і бібліотека мови С*	2		2	4	
Всього годин за 2-й семестр:	**36**	**–**	**36**	**72**	**–**
Модуль 3. ***Алгоритми: побудова і аналіз*** *(3 семестр)*					
Змістовий модуль 4. ***Сортування і порядкові статистики***					
Тема 4.1. *Сортування за час* $\theta(n^2)$	6		8	4	
Тема 4.2. *Сортування за час* $\theta(n*\log_2 n)$				4	
Тема 4.3. *Сортування за час* $\theta(n)$				4	
Змістовий модуль 5. ***Абстрактні типи даних***					
Тема 5.1. *Елементарні структури даних*	2		10	4	
Тема 5.2. *Хеш- таблиці*	2			4	
Тема 5.3. *Двійкові дерева пошуку*	4			4	
Змістовий модуль 6. ***Методи побудови і аналізу алгоритмів***					
Тема 6.1. *Швидкість росту функцій*	2		14	2	
Тема 6.2. *Алгоритми „Розподіляй і володарюй”*	2			2	
Тема 6.3. *Динамічне програмування*	2			2	
Тема 6.4. *Жадібні алгоритми*	2			2	
Тема 6.5. *Пошук з поверненням*	2			2	

Модуль 4. ***Основи розробки програмного забезпечення***					
Змістовий модуль 7**.** ***Основи розробки програмного забезпечення***					
Тема 7.1. *Постановка задачі*	2		4	26	
Тема 7.2. *Технічне завдання*					
Тема 7.3. *Створення програмного продукту, впровадження і супровід*					
Модуль 5. ***Парадигми програмування***					
Змістовий модуль 8. ***Огляд парадигм програмування***					
Тема 8.1. *Мови програмування низького рівня. Мови асемблера*	2			2	
Тема 8.2. *Вступ до функціонального програмування*	2			2	
Тема 8.3. *Вступ до логічного програмування*				2	
Тема 8.4. *Вступ до об'єктно-орієнтованого програмування*	2			2	
Тема 8.5. *Паралельне програмування. Огляд.*	2			2	
Тема 8.6. *Internet-програмування. Огляд.*	2			2	
Всього годин за 3-й семестр:	**36**	**–**	**36**	**134**	**–**
В с ь о г о г о д и н:	**108**	**–**	**108**	**216**	**–**

ПРИКЛАДИ ЗАВДАНЬ ДЛЯ САМОСТІЙНОЇ РОБОТИ

1. Створити програмний продукт, за допомогою якого реалізується графічна модель руху планет навколо Сонця.
2. Створити програмний продукт, за допомогою якого реалізується дошка Гальтона для демонстрації законів теорії ймовірності.
3. Створити програмний продукт, за допомогою якого реалізується найпростіший текстовий редактор.
4. Створити програмний продукт, за допомогою якого реалізується обертання кубу з різнокольоровими гранями.
5. Створити програмний продукт, за допомогою якого реалізується виконавець "Черепашка".
6. Створити програмний продукт, за допомогою якого реалізується програма «Калькулятор».

МЕТОДИ НАВЧАННЯ: інформаційно-рецептивний, репродуктивний, проблемний, евристичний, дослідницький, інтерактивні методи, активні методи.

МЕТОДИ ОЦІНЮВАННЯ: поточне оцінювання, модульний контроль, залік, екзамен, оцінювання розробленого програмного продукту і його захист.

РОЗПОДІЛ БАЛІВ, ЯКІ ПРИСВОЮЮТЬСЯ СТУДЕНТАМ

Таблиця А.3

РІК НАВЧАННЯ 1, СЕМЕСТР 1

<table>
<tr><td colspan="10">Модуль 1</td><td>Вид підсумкового контролю</td><td>Загальна сума балів</td></tr>
<tr><td colspan="10">Змістовий модуль 1</td><td>Тестування</td><td rowspan="4"></td></tr>
<tr><td colspan="10">5</td><td rowspan="3">3</td></tr>
<tr><td colspan="2">Т1</td><td colspan="2">Т2</td><td colspan="2">Т3</td><td colspan="2">Т4</td><td colspan="2">Т5</td></tr>
<tr><td colspan="2">1</td><td colspan="2">1</td><td colspan="2">1</td><td colspan="2">1</td><td colspan="2">1</td></tr>
<tr><td colspan="10">Змістовий модуль 2</td><td>Модульний контроль</td><td>Екзамен</td><td rowspan="4">100</td></tr>
<tr><td colspan="10">49</td><td>8</td><td rowspan="3">5</td></tr>
<tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td rowspan="2">7</td></tr>
<tr><td>3</td><td>3</td><td>3</td><td>3</td><td>11</td><td>14</td><td>3</td><td>3</td><td>3</td><td>3</td></tr>
</table>

РІК НАВЧАННЯ 1, СЕМЕСТР 2

<table>
<tr><td colspan="14">Модуль 2
(поточний контроль, залік)</td><td>Вид підсумковог о контролю</td><td>Загальна сума балів</td></tr>
<tr><td colspan="14">Змістовий модуль 3</td><td>Мо дульний контроль</td><td rowspan="4">80</td></tr>
<tr><td colspan="14">54</td><td rowspan="3">26</td></tr>
<tr><td>Т 1</td><td>Т 2</td><td>Т 3</td><td>Т 4</td><td>Т 5</td><td>Т 6</td><td>Т 7</td><td>Т 8</td><td>Т 9</td><td>Т 10</td><td>Т 11</td><td>Т 12</td><td>Т 13</td><td>Т 14</td></tr>
<tr><td>3</td><td>3</td><td>3</td><td>6</td><td>3</td><td>3</td><td>3</td><td>9</td><td>6</td><td>3</td><td>3</td><td>3</td><td>3</td><td>3</td></tr>
</table>

РІК НАВЧАННЯ 2, СЕМЕСТР 3

<table>
<tr><td colspan="5">Модуль 3
(поточний контроль, екзамен)</td><td>Вид підсумкового контролю</td><td>Загальна сума балів</td></tr>
<tr><td colspan="5">Змістовий модуль 4</td><td rowspan="6">Тестування
Модульний контроль
7</td><td rowspan="6"></td></tr>
<tr><td colspan="5">12</td></tr>
<tr><td colspan="2">Т1</td><td>Т2</td><td colspan="2">Т3</td></tr>
<tr><td colspan="2">3</td><td>6</td><td colspan="2">3</td></tr>
<tr><td colspan="5">Змістовий модуль 5</td></tr>
<tr><td colspan="5">16</td></tr>
<tr><td colspan="2">Т1</td><td>Т2</td><td colspan="2">Т3</td><td rowspan="2"></td><td rowspan="2"></td></tr>
<tr><td colspan="2">6</td><td>4</td><td colspan="2">6</td></tr>
</table>

<table>
<tr><td colspan="6">Змістовий модуль 6</td><td></td><td>55</td></tr>
<tr><td colspan="6">13</td><td rowspan="3">14</td><td rowspan="3"></td></tr>
<tr><td>Т1</td><td>Т2</td><td>Т3</td><td>Т4</td><td>Т5</td><td></td></tr>
<tr><td>1</td><td>3</td><td>3</td><td>3</td><td>3</td><td></td></tr>
<tr><td colspan="6">Модуль 4
(поточний контроль, захист програмного продукту*)</td><td>Вид підсумкового контролю</td><td>Загаль на сума балів</td></tr>
<tr><td colspan="6">Змістовий модуль 7</td><td>Захист програмного продукту</td><td rowspan="4">30</td></tr>
<tr><td colspan="6">5</td><td rowspan="3">25</td></tr>
<tr><td colspan="2">Т1</td><td colspan="2">Т2</td><td colspan="2">Т3</td></tr>
<tr><td colspan="2">1</td><td colspan="2">2</td><td colspan="2">2</td></tr>
<tr><td colspan="6">Модуль 5
(поточний контроль, екзамен)</td><td>Вид підсумковог о контролю</td><td>Загальна сума балів</td></tr>
<tr><td colspan="6">Змістовий модуль 8</td><td rowspan="4">10</td><td>Екзамен</td><td rowspan="4">200 (80 балів 2 семестр + 120 балів 3 семестр)</td></tr>
<tr><td colspan="6">10</td><td rowspan="3">25</td></tr>
<tr><td>Т1</td><td>Т2</td><td>Т3</td><td>Т4</td><td>Т5</td><td>Т6</td></tr>
<tr><td>2</td><td colspan="2">2</td><td>2</td><td>2</td><td>2</td></tr>
</table>

*Оцінку "відмінно" студент може одержати тільки у випадку, якщо він брав участь у розробці програмного продукту.

Шкала оцінювання

90% -100% балів – відмінно (A)

75% - 89% балів – добре (B, C)

60% - 74% балів – задовільно (D, E)

35% - 59% балів – незадовільно з можливістю повторного складання (FX)

1%-34% балів – незадовільно з обов'язковим повторним курсом (F)

МЕТОДИЧНЕ ЗАБЕЗПЕЧЕННЯ: опорні конспекти лекцій, презентації лекцій, дидактичні матеріали, педагогічні програмні засоби.

ЛІТЕРАТУРА

Основна

1. Абрамов С.А., Гнездилова Г.Г., Капустина Е.Н., Селюн М.И. Задачи по программированию - М.: Наука. Гл. ред. физ.-мат. лит., 1988. - 224 с.
2. Ахо А.В., Хопркрофт Д.Э., Ульман Д.Д. Построение и анализ вычислительных алгоритмов.: Пер. с англ. - М.: Издательство "Мир", 1978. - 535 с.
3. Ахо А.В., Хопркрофт Д.Э., Ульман Д.Д. Структуры данных и алгоритмы.: Пер. с англ. - М.: Издательский дом "Вильямс", 2001. - 384 с.
4. Брукс Ф.П. Как проектируются и создаются программные комплексы. Пер. с англ. - М.: "Наука", 1979. - 151 с.
5. Вирт Н. Алгоритмы + структуры данных = программы: Пер. с англ. – М.: Мир, 1985. – 406 с.
6. Зуев Е.А. Язык программирования Turbo Pascal 6.0, 7.0. – М.: Веста, Радио и связь, 1993. – 384 с.
7. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ. М.: МЦНМО, 1999. – 960 с.
8. Макконел С. Совершенный код. Мастер-класс /Пер. с англ. – М.: Издательский торговый дом «Русская редакция»; СПб.: Питер, 2005. – 896 с.
9. Пильщиков В.Н.Сборник упражнений по языку Паскаль: Учеб. пособие для вузов.- М.: Наука. Гл. ред. физ-мат. лит., 1980. - 160 с.
10. Прата С. Язык программирования С. Лекции и упражнения, 5-е издание. : Пер. с англ. – М.: Издательство "Вильямс", 2006. – 960 с.
11. Румянцев Д., Монастырский Л. Путь программиста: Опыт созидания личности программиста. - М.: "Издательский дом ИНФРА-М", 2000. - 864 с.
12. Себеста Р. Основные концепции языков программирования. 5-е издание.: Пер. с англ. - М.: Издательский дом "Вильямс", 2001. - 672 с.
13. Фьюер А. Задачи по языку Си. – М.: Финансы и статистика, 1985.
14. Шень А. Программирование: теоремы и задачи. М.: МЦНМО, 1995. – 264 с.

Додаткова література

1. Бондарев В.М., Рубенецкий В.И., Качко Е.Г. Основы программирования. – Харьков: Фолио, 1998. – 368 с.

2. Брукшир Дж. Введение в компьютерные науки. Общий обзор, 6-е издание.: Пер. с англ. - М.: Издательский дом “Вильямс”, 2001. - 688 с.
3. Грин Д., Кнут Д. Математические методы анализа алгоритмов. - М.: Мир, 1987. – 120 с.
4. Грэхем Р., Кнут Д., Паташник О. Конкретная математика. Основание информатики: Пер. с англ. – М.: Мир, 1998. – 703 с.
5. Дал У., Дейкстра Э., Хоор К. Структурное программирование Пер. с англ. - М.: Издательство “Мир”, 1975. - 245 с.
6. Дейкстра Э. Дисциплина программирования Пер. с англ. - М.: Издательство “Мир”, 1978. - 274 с.
7. Зубов В.С. Справочник программиста. Базовые методы решения графовых задач и сортировки. - М.: Информационно-издательский Дом “Филинъ”, 1999. – 256 с.
8. Керниган Б.В., Пайк Р. Практика программирования.: Пер. с англ. - СПб.: “Невский диалект”, 2001. - 381 с.
9. Керниган Б., Ритчи Д. Язык программирования Си.: Пер. с англ. 3-е изд., испр. – СПб.: “Невский диалект”, 2001. - 352 с.
10. Кнут Д.Э. Искусство программирования, том 1. Основные алгоритмы, 3-е изд.: Пер. с англ.: Уч. Пос. – М.: Издательский дом «Вильямс», 2000. – 720 с.
11. Кнут Д.Э. Искусство программирования, том 3. Сортировка и поиск, 3-е изд.: Пер. с англ.: Уч. Пос. – М.: Издательский дом “Вильямс”, 2000. – 832 с.
12. Лавров С.С. Программирование. Математические основы, средства, теория. - СПб.: БХВ-Петербург, 2001. - 320 с.
13. Макконел Дж. Анализ алгоритмов, Вводный курс. – Москва: Техносфера, 2002. – 304 с.
14. Марченко А.И., Марченко Л.А. Программирование в среде Turbo Pascal 7.0. – М.: Бином Универсал, 1977.
15. Милов А.В. Основы программирования в задачах и примерах, Харьков: ФОЛИО, 2002.- 397 с.
16. Могилев А.В. и др. Информатика: Учеб. пособие для пед. вузов / А.В. Могилев, Н.И. Пак, Е.К. Хённер; Под ред. У.К. Хённера. - М.:, 1999. – 816 с.
17. Могилев А.В. и др. Практикум по информатике: Учеб. пособие для студ. высш. учеб. заведений / А.В. Могилев, Н.И. Пак, Е.К. Хённер; Под ред. У.К. Хённера. - М.: Издательский центр “Академия”, 2001. – 608 с.
18. Ноден П., Китте К. Алгебраическая алгоритмика (с упражнениями и решениями): Пер. с франц. – М.: Мир, 1999. – 720 с.
19. Пратт Т., Зелковиц М. Языки программирования: разработка и реализация. - СПб.: Питер, 2002. - 688 с.
20. Ставровский А.Б. Турбо Паскаль 7.0. Учебник. – К.: Издательская группа BHV, 2000. – 400 с.
21. Юркин А.Г. Задачник по программированию. - СПб.: Питер, 2002. - 192 с.

ДОДАТОК Б

Приклад тексту програми, який задовольняє вимогам до оформлення програм.

Умова задачі: Дано символ `ch` і два цілих додатних числа `rows` і `cols`. Написати програму, яка виводить на екран символ `ch` у `rows` рядках і `cols` стовпчиках.

```
/* showchar2.c – друкує символи в рядках та стовпцях */
  #include <stdio.h>

  /* прототип функції display */
  void display(char cr, int lines, int width);

  int main(void)
   {
    int ch;                          /* символ, що виводиться на друк */
     int rows, cols;              /* кількість рядків та столпців     */

    printf("Введіть символ і два циліх числа:\n");
    while ((ch = getchar()) != '\n')
     {
      if (scanf("%d %d", &rows, &cols) != 2)
       break;
      display(ch, rows, cols); /* виклик функції display */
      while (getchar() != '\n')
       continue;
      printf ("Введіть ще один символ і два циліх числа;\n");
      printf ("або введіть символ нового рядку для завершення
                        програми.\n");
      }
     printf("Програма завершена.\n");

    return 0;
  }

  void display(char cr, int lines, int width)
   {
    int row, col;                /*локальні змінні*/

     for (row = 1; row <= lines; row++)
      {
       for (col = 1; col <= width; col++)
        putchar(cr);
       putchar ('\n');        /*перейти на новий рядок*/
       }
   }
```

ДОДАТОК Б

Загальні вимоги до завдання на створення студентами навчального програмного продукту

1. Проект повинен відповідати основним показникам якісного програмного продукту:
   – ефективність, тобто швидкість виконання, мінімальний обсяг необхідної пам'яті;
   – зручність у використанні, а саме, наявність дружнього інтерфейсу і необхідної документації;
   – надійність, мається на увазі, безвідмовність, захищеність і безпека;
   – зручність супроводу – можливість удосконалення і модифікації продукту при внесенні змін до проекту.
2. Тексти програм проекту повинні бути написані мовами Pascal або C у відповідних інтегрованих середовищах програмування.
3. Окремі фрагменти програм або модулі, якщо буде потреба, наприклад, організація взаємодії з периферійними пристроями на низькому рівні, можуть бути написані мовою асемблера.
4. Результатами роботи над навчальним програмним продуктом є:
   a) створений програмний продукт (вихідні тексти програм), збережений на зовнішньому носії (дискета, CD-R або CD-RW);
   b) тверда копія документації програмного продукту з описом роботи модулів (файлів) і їхньої взаємодії;
   c) короткий опис роботи, виконаної кожним учасником розробки;
   d) інструкція користувачеві.

Література:

1. Брукс Ф.П. Как проектируются и создаются программные комплексы: Пер. с англ. – М.: “Наука”, 1979. – 151 с.
2. Макконнелл С. Совершенный код. Мастер-класс: Пер. с анг. – М.: Издательско-торговый дом "Русская редакция"; СПб.: Питер, 2005. – 896 с.
3. Румянцев Д., Монастырский Л. Путь программиста: Опыт созидания личности программиста. – М.: “Издательский дом ИНФРА-М”, 2000. – 864 с.

ДОДАТОК В

Приклад тексту програми, який задовольняє вимогам до оформлення програм.

Умова задачі: Дано символ `ch` і два цілих додатних числа `rows` і `cols`. Написати програму, яка виводить на екран символ `ch` у `rows` рядках і `cols` стовпчиках.

```
/* showchar2.c – друкує символи в рядках та стовпцях */
  #include <stdio.h>
  /* прототип функції display */
  void display(char cr, int lines, int width);

  int main(void)
   {
    int ch;                          /* символ, що виводиться на друк */
    int rows, cols;                  /* кількість рядків та столпців */

    printf("Введіть символ і два циліх числа:\n");
    while ((ch = getchar()) != '\n')
     {
      if (scanf("%d %d", &rows, &cols) != 2)
       break;
      display(ch, rows, cols); /* виклик функції display */
      while (getchar() != '\n')
       continue;
      printf ("Введіть ще один символ і два циліх числа;\n");
      printf ("або введіть символ нового рядку для завершення
                        програми.\n");
     }
    printf("Програма завершена.\n");

    return 0;
  }

  void display(char cr, int lines, int width)
   {
    int row, col;                    /*локальні змінні*/

    for (row = 1; row <= lines; row++)
     {
      for (col = 1; col <= width; col++)
       putchar(cr);
      putchar ('\n');          /*перейти на новий рядок*/
     }
   }
```

ДОДАТОК Г

Тексти програм та вихідні форми навчального програмного продукту «Мапа автомобільних шляхів України», розробленого студентами

```
//***********************************************
//                        GMap
//                      ProGraM
//***********************************************
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include<roadkull.h>
#include<kordonua.h>
#include<dnipro.h>
#include<oblasti.h>
#include<oblasti1.h>
#include<mouse.h>
#include<goroda3.h>

int Menu(void);
void UA(void);
void dnipro(void);
void oblast(void);
void OblastiColor(void);
int exit_program(void);
int HELP(void);
int Abt(void);
int spisok(void);
void save_image(int t);
int intimage(int t);
void WriteStrXY(char *str ,int x,int y,int col,int m, int z,int fon);
void action_map(void);
void Data_out(const float kilometr,int kol,const int *nam_gorod);
void menuoll(void);
char*  Prin_File_to_save(void);
int roadwent(int u1,int u2,float *km,int *vert,int *gor);
void Ukraine(void);
void stroka(float km,char *c);

int main(void)
 {
  /* request auto detection */
  int gdriver = DETECT, gmode, errorcode;

  /* initialize graphics and local variables */
  initgraph(&gdriver, &gmode, "");

  /* read result of initialization */
  errorcode = graphresult();
  if (errorcode != grOk)  /* an error occurred */
   {
    printf("Graphics error: %s\n", grapherrormsg(errorcode));
    printf("Press any key to halt:");
    getch();
    exit(1); /* terminate with an error code */
   }

  ResetMouse();
  intimage(0);
  ShowMouseCursor();
  action_map();
  closegraph();
  return 0;
 }
//***********************************
```

```
void action_map(void)
 {
  int   *nam_gorod;
  unsigned char ch='p';
  int x,y,button;
  int G;
  int u1=0;
  float kilometr;
  int   kol;
  ShowMouseCursor();
  while(ch!=27) {
   if(kbhit())
    ch=getch();

  ReadMouse(&x,&y,&button);
  if(ch==27||(button==1&&(x>=620&& x<=639) && (y>=460 && y<=479)))
   {
    int ep=0;
    unsigned int volume;
    void *arrow;
    HideMouseCursor();
    volume=imagesize(199,89,460,260);
    arrow=malloc(volume);
    getimage(199,89,460,260,arrow);
    ep=exit_program();
    HideMouseCursor();
    if(ep==1)
     {
      closegraph();
      exit(EXIT_SUCCESS);
      free(arrow);
     }
    else
     {
      putimage(199,89,arrow,0);
      HideMouseCursor();
      ch='p';
     }
    free(arrow);
   }

  ShowMouseCursor();
  if(ch=='H'||ch=='h'||(button==1&&(x>=314&& x<=470) && (y>=0 && y<=12)))
   {
    int ep=0;
    unsigned int volume;
    void *arrow;
    HideMouseCursor();
    volume=imagesize(199,89,460,260);
    arrow=malloc(volume);
    getimage(199,89,460,260,arrow);
    ep=HELP();
    HideMouseCursor();
    if(ep==1)
     {
      putimage(199,89,arrow,0);
      ch='p';
     }
    free(arrow);
   }

  if(ch=='a'||ch=='A'||(button==1&&(x>=471&& x<=639) && (y>=0 && y<=12)))
   {
    int ep=0;
    unsigned int volume;
    void *arrow;
    HideMouseCursor();
    volume=imagesize(199,89,460,260);
    arrow=malloc(volume);
    getimage(199,89,460,260,arrow);
```

```
              ep=Abt();
              HideMouseCursor();
              if(ep==1)
               {
                putimage(199,89,arrow,0);
                ch='p';
               }
              free(arrow);
             }
           if(ch=='s'||ch=='S'||  ((button==1)&&((x>=0&&  x<=156)  &&  (y>=0  &&  y<=12))))
//Knopka save
            {
             save_image(1);
//Knopka save
             ch='p';
            }
  if(ch=='o'||ch=='O'|| ((button==1)&&((x>=157&& x<=313) && (y>=0 && y<=12))))  //Knopka
        zagruzki kartinki
            {
             intimage(2);                                                          //Knopka
zagruzki kartinki
             ch='p';
            }
           if(ch=='g'||ch=='G'|| ((button==1)&&((x>=0&& x<=15) && (y>=40 && y<=90))) )
            {
             int  ep;
             if(u1==0)
              {
               G=spisok();
               if(G!=0)
                {
                 u1=gorod[G].nam;
                 intimage(0);
                 setcolor(0);
                 circle(aa[u1].x,aa[u1].y,2);
                 WriteStrXY(aa[u1].gor,aa[u1].x,aa[u1].y,15,0,0,0);
                }
               else
                {
                 intimage(0);
                 ep=0;
                 u1=0;
                }
               ch='p';
              }
             else
              {
               int u2;
               G=spisok();
               if(G!=0)
                {
                 u2=gorod[G].nam;
                 ep=roadwent(u1,u2,&kilometr,&kol,nam_gorod);
                }
               else
                ep=0;
               u1=0;
               intimage(0);
              }
            if(u1==0&&ep!=0)
             {
              button=0;
              Data_out(kilometr,kol,nam_gorod);
              while(ch!=27&&ch!='n'&&button!=1&&ch!='N'&&ch!='G')
               {
              if(kbhit())
               ch=getch();

               ReadMouse(&x,&y,&button);
```

```
            if((button==1)&&(((x>=20&&x<=200)&&(y>=442&&y<=460))))
             ch='G';
            if((button==1)&&((x>=0&& x<=156) && (y>=0 && y<=12)))
             {
              save_image(1);     //Knopka save
              button=0;
             }
            if(ch==27||ch=='n'||ch=='N')
             {
              HideMouseCursor();        //ESC abo 'n,N'

            setfillstyle(0,0);
            bar(10,300,200,460);        //zakrutie okna s kilometrami

            ShowMouseCursor();          //ch obratno 'p'
            }
           }
          if(ch!='G')
           ch='p';
         }
        }
       }
      }
     //***********************************************
     void Data_out(const float kilometr,int kol,const int *nam_gorod)
      {
       int i,j;
       int x,y,button;
       int k=1;
       char *cth;
       HideMouseCursor();
       for(i=10;i<200;i++)
        for(j=300;j<460;j++)
         putpixel(i,j,8);
       for(i=10;i<200;i++){
        putpixel(i,300,2);
        putpixel(i,460,2);
       }
       for(i=300;i<460;i++){
        putpixel(10,i,2);
        putpixel(200,i,2);
       }
       WriteStrXY("Kilometrov:",25,350,0,0,0,0);
       stroka(kilometr,cth);
       WriteStrXY(cth,25,380,4,0,0,0);
       WriteStrXY("ﾉ^^^^^^^^^^^^^^^^^^^^ｻ",20,442,0,0,0,0);
       WriteStrXY("ｺ                    ｺ",20,448,0,0,0,0);
       WriteStrXY("  Close end new->",20,448,0,0,0,0);
       WriteStrXY("ﾈ^^^^^^^^^^^^^^^^^^^^ｼ",20,454,0,0,0,0);
       setcolor(0);
       circle(aa[nam_gorod[k]].x,aa[nam_gorod[k]].y,2);

WriteStrXY(aa[nam_gorod[k]].gor,aa[nam_gorod[k]].x,aa[nam_gorod[k]].y,15,0,0,0);
       moverel(aa[nam_gorod[k]].x-getx(),aa[nam_gorod[k]].y-gety());
       while(k++<kol)
        {
         setcolor(0);
         lineto(aa[nam_gorod[k]].x,aa[nam_gorod[k]].y);
        }
        circle(aa[nam_gorod[k-1]].x,aa[nam_gorod[k-1]].y,2);
        WriteStrXY(aa[nam_gorod[k-1]].gor,aa[nam_gorod[k-1]].x,aa[nam_gorod[k-
1]].y,15,0,0,0);
        ShowMouseCursor();
       }
     //*****************************************
     int exit_program(void)
      {
       char ch='1';
```

```
  int i,j;
  int x,y,button;

  for(i=200;i<460;i++)
   for(j=90;j<260;j++)
    putpixel(i,j,DARKGRAY);
  for(i=200;i<460;i++){
   putpixel(i,90,2);
   putpixel(i,260,2);
  }
  for(i=90;i<260;i++){
   putpixel(200,i,2);
   putpixel(460,i,2);
  }
  WriteStrXY("   Exit program?",260,140,15,0,0,0);
  WriteStrXY("ﾉ^^^^^^^ｻ",260,184,0,0,0,0);
  WriteStrXY("ｺ       ｺ",260,190,0,0,0,0);
  WriteStrXY("  Y-yes",260,190,15,0,0,0);
  WriteStrXY("ﾈ^^^^^^^ｼ",260,196,0,0,0,0);
  WriteStrXY("ﾉ^^^^^^^ｻ",350,184,0,0,0,0);
  WriteStrXY("ｺ       ｺ",350,190,0,0,0,0);
  WriteStrXY("  N-no",350,190,15,0,0,0);
  WriteStrXY("ﾈ^^^^^^^ｼ",350,196,0,0,0,0);
  WriteStrXY("******************************",210,250,15,0,0,0);
  ShowMouseCursor();
  while(ch!=27)
   {
    ReadMouse(&x,&y,&button);
    if((button==1)&&(x>=250 && x<=290)&& (y>=183 && y<=200))
     return 1;
    if((button==1)&&(x>=340 && x<=430)&& (y>=185 && y<=200))
     return 0;

    if(kbhit())
     {
      ch=getch();
      if(ch=='Y'||ch=='y')
       return 1;
      else
       return 0;
     }
   }
   return 0;
 }
//***********************************
int HELP(void)
 {
  char ch='l';
  int i,j;
  int x,y,button;

  for(i=200;i<460;i++)
   for(j=90;j<260;j++)
    putpixel(i,j,DARKGRAY);
  for(i=200;i<460;i++){
   putpixel(i,90,2);
   putpixel(i,260,2);
  }
  for(i=90;i<260;i++){
   putpixel(200,i,2);
   putpixel(460,i,2);
  }
  WriteStrXY("************HELP************",210,100,15,0,0,0);
  WriteStrXY("S- sohranit put",210,110,15,0,0,0);
  WriteStrXY("O- otkrit fail",210,120,15,0,0,0);
  WriteStrXY("!!Sohranat i otkrivat fail s!!",210,130,4,0,0,0);
  WriteStrXY("!!rozhireniem .pic!!",210,140,4,0,0,0);
  WriteStrXY("G- spisok gorodov",210,150,15,0,0,0);
  WriteStrXY("H- pomosh po upravlrniy",210,160,15,0,0,0);
```

```
  WriteStrXY("Vse eti deistvia mozno vipolnat",210,170,4,0,0,0);
  WriteStrXY("s pomoshy mihki najimaya na ",210,180,4,0,0,0);
  WriteStrXY("sootvecvuchi knopki s podpisami",210,190,4,0,0,0);
  WriteStrXY("/^^^
```

```
     ReadMouse(&x,&y,&button);
     if(flag!=nam &&(x>=5 && x<=630) && (y>=5 && y<=470))
      {
       flag=0;
     HideMouseCursor();
       putimage(newx-2,newy-2,arrow,0);
       free(arrow);
      }
     nx=x/126+1;
     ny=y/10+1;
     nam=(ny-1)*5+nx;
     if(flag==0&&((x>=5 && x<=630) && (y>=5 && y<=470)))
      {
     nx=x/126+1;
     ny=y/10+1;
     nam=(ny-1)*5+nx;
     newx=((x/126)*126)+5;
     newy=((y/10)*10)+5;
     HideMouseCursor();
     volume=imagesize(newx-2,newy-2,newx+126,newy+10);
     arrow=malloc(volume);
     getimage(newx-2,newy-2,newx+126,newy+10,arrow);
     WriteStrXY(gorod[nam].ck,newx,newy,0,3,0,0);
     flag=nam;
      }

     ShowMouseCursor();
     if(kbhit())
      {
       ch=getch();
       if(ch==27)
      {
      free(arrow);
      return 0;
      }
     }
    if(button==1)
     {
      free(arrow);
      return nam;
     }
   }
   return 0;
}
//*********************************
    void save_image(int t)
{
  int i,j;
  FILE *fpic;
  HideMouseCursor();
  if(t==0)
  {
   if ((fpic = fopen("dat\\skech.tmp", "w+"))== NULL)
    {
     closegraph();
     fprintf(stderr, "Cannot open output file.\n");
    }
  }
    else
     {
      char *Pstr;
      Pstr =Prin_File_to_save();

      if ((fpic = fopen(Pstr, "w+"))== NULL)
       {
        fprintf(stderr, "Cannot open output file.\n");
       }
     }
    for(i=0;i<640;i++)
     for(j=0;j<480;j++)
```

```
      fprintf(fpic,"%c",(getpixel(i,j)+100));
     fclose(fpic);
     ShowMouseCursor();
    }
//*************************************
int intimage(int t)
{
  int i,j;
  char ch;
  FILE *fpic;
  HideMouseCursor();

  if(t==0)
   {
    if ((fpic = fopen("dat\\skech.tmp", "r+"))== NULL)
     {
      clrscr();
      WriteStrXY("Nenayden fail biblioteki.",5,400,15,15,0,0);
      WriteStrXY("(ENTER)",30,410,15,15,0,0);
      getch();
      closegraph();
     }
    }
   else
    {
   char *Pstr;
   Pstr =Prin_File_to_save();
   if ((fpic = fopen(Pstr, "r+"))== NULL)
    {
     WriteStrXY("Nenayden fail ili nekorekten vvod.",5,400,15,15,0,0);
     WriteStrXY("sohranenie i zagruzka s rozhireniem (*.pic)",6,410,15,15,0,0);
     return 2;
    }
   }
  for(i=0;i<640;i++)
   for(j=0;j<480;j++){
    fscanf(fpic,"%c",&ch);
    putpixel(i,j,ch-100);
   }
   ShowMouseCursor();
        fclose(fpic);
  return 0;
}
//*****************************************
void WriteStrXY(char *str ,int x,int y,int col,int m, int z,int fon)
{
   setcolor(col);
   settextstyle(m,z,1);
   outtextxy(x,y,str);
   textbackground(fon);
}
//*************************************
  char*  Prin_File_to_save(void){
  static  char StrFile[30];

  int x,y;
  unsigned int volume;
  void *arrow;
  HideMouseCursor();
  volume=imagesize(1,13,310,33);
  arrow=malloc(volume);
  getimage(1,13,310,33,arrow);
  setcolor(7);
  setfillstyle(1,7);
  rectangle(1,13,300,33);
  floodfill(3,15,7);
  WriteStrXY(" name:",2,15,10,0,0,10);
  gotoxy(10,2);
  gets(StrFile);
  putimage(1,13,arrow,0);
```

```
  free(arrow);
  return StrFile;
 }
//************************************
 void stroka(float km,char *c)
 {
  int tmp;
  int s,del=1000;
  int i=0;
  tmp=km;
  while(del!=0)
   {
    s=tmp/del;
    if(s!=0)
     c[i++]=48+s;
    if(tmp-(tmp-s*del)<=(del/10))
     c[i++]=48;
    tmp=tmp-s*del;
    del=del/10;
  }
  c[i]='\0';
}
//*******************************************
 int Abt(void)
 {
  char ch='l';
  int i,j;
  int x,y,button;

  for(i=200;i<460;i++)
   for(j=90;j<260;j++)
    putpixel(i,j,DARKGRAY);
   for(i=200;i<460;i++){
    putpixel(i,90,2);
    putpixel(i,260,2);
   }
   for(i=90;i<260;i++){
    putpixel(200,i,2);
    putpixel(460,i,2);
   }
   WriteStrXY("*************WWW**************",210,100,15,0,0,0);
   WriteStrXY("*************MMM**************",210,250,15,0,0,0);
   WriteStrXY("       GMap v1.01.08          ",210,150,4,3,0,0);
   WriteStrXY("Author:   Andrey N aka Skvaer ",210,190,4,3,0,0);
   WriteStrXY("mail:     Skvaer.haos@mail.ru ",210,200,4,3,0,0);
   WriteStrXY("   Free Software Foundation   ",210,230,5,0,0,0);
   ShowMouseCursor();
   while(ch!=27)
    {
     ReadMouse(&x,&y,&button);
     if((button==1)&&(x>=0 && x<=639)&& (y>=0 && y<=479))
      return 1;
     if(kbhit())
      {ch=getch();
       if(ch==27)
      return 1;
      }
   }
  return 0;
}

 void Ukraine(void)
 {
  OblastiColor();
  dnipro();
  oblast();
  UA();
  Menu();
 }
 int Menu(void){
```

```
 WriteStrXY("--Ukraine--",30,25,3,3,0,0);
 return 0;
}
void UA(void)
 {
  int i=0;
  setcolor(6);
  setfillstyle(1,6);
  setlinestyle(2, 5, 10);
  moverel(gran[1].x,gran[1].y);
  while(i++<239)
   lineto(gran[i].x,gran[i].y);
}
/*void dnipro(void)
{
  int j=1;
  setcolor(1);
  setfillstyle(1,1);
  moverel(rive[j].x-getx(),rive[j].y-gety());
  while(j++<84)
   lineto(rive[j].x,rive[j].y);
  floodfill(295,100,1);
  floodfill(307,148,1);
  floodfill(419,296,1);
  floodfill(417,220,1);
}
void oblast(void)
 {
  int j=1;
  setcolor(4);
  setfillstyle(1,4);
  moverel(obl[j].x-getx(),obl[j].y-gety());
  while(j++<722)
   {
    if(obl[j].x==0)
     {
      j++;
      moverel(obl[j].x-getx(),obl[j].y-gety());
      j++;
     }
    lineto(obl[j].x,obl[j].y);
  }
 }
 void OblastiColor(void)
 {
  int i=-1;
  while(i++<=1640)
   {
    if(i>2 && obls[i].x==0)
     floodfill(obls[i].f,obls[i].k,getcolor());
    if(obls[i].x==0)
     {
      setcolor(obls[i].y);
      setfillstyle(1,obls[i].y);
      moverel(obls[i+1].x-getx(),obls[i+1].y-gety());
      i++;
     }
    lineto(obls[i].x,obls[i].y);
   }
}

 void menuoll(void)
{
  //rectangle
  setcolor(7);
  setfillstyle(1,7);
  rectangle(0,40,15,90);     //knopka gorodov
  rectangle(0,0,640,12);     //  verh bar
  floodfill(2,2,7);
```

```
          floodfill(2,41,7);
          setcolor(12);
          rectangle(0,40,15,90);                              WriteStrXY("G",12,46,4,0,1,0);
WriteStrXY("orod",12,46,0,0,1,0);//knopka gorodov
          setcolor(12);
          rectangle(0,0,156,12);       WriteStrXY("S",2,3,4,0,0,0);       WriteStrXY("AVE
as.",2,3,0,0,0,0);//verh bar
          setcolor(12);
          rectangle(157,0,313,12);                            WriteStrXY("O",157,3,4,0,0,0);
WriteStrXY("PEN",157,3,0,0,0,0);
          setcolor(12);
          rectangle(314,0,470,12);                            WriteStrXY("H",314,3,4,0,0,0);
WriteStrXY("ELP",314,3,0,0,0,0);
          setcolor(12);
          rectangle(471,0,639,12);                            WriteStrXY("A",471,3,4,0,0,0);
WriteStrXY("BOUT",471,3,0,0,0,0);

          setcolor(7);
          setfillstyle(1,7);
          rectangle(620,460,639,479);
          floodfill(621,461,7);

          setcolor(4);
          rectangle(620,460,639,479);
          WriteStrXY("X",627,467,4,0,0,0);
         } */

        //
        #include<stdio.h>
         struct kord{
          int x;
          int y;
         };
        struct                                                                    kord
kordon[240]={{0,0},{10,224},{14,214},{29,188},{40,193},{38,185},{40,168},{36,164},{55,138
},{76,119},
{85,119},{88,108},{86,97},{90,95},{85,85},{82,78},{77,70},{79,49},{86,46},{91,52},{101,44
},{102,37},{131,36},{139,39},{153,39},{164,44},{170,44},{176,48},{181,48},{184,52},{194,5
2},{196,60},{209,63},{208,70},{214,69},{214,63},{220,65},{225,61},{228,66},{239,65},{239,
72},{243,72},{246,65},{254,61},{260,76},{265,76},{272,69},{275,75},{279,75},{282,71},{291
,72},{295,82},{300,83},{300,74},{298,62},{302,52},{312,40},{321,39},{327,35},{342,37},{34
6,41},{352,38},{355,31},{354,23},{364,27},{375,27},{379,21},{389,22},{392,26},{396,21},{4
07,29},{409,39},{416,49},{420,56},{409,61},{417,69},{415,75},{419,79},{416,84},{425,83},{
430,88},{442,85},{445,93},{452,93},{449,98},{455,107},{454,119},{462,131},{474,126},{484,
133},{492,133},{496,137},{504,130},{520,123},{527,135},{543,148},{547,140},{552,140},{557
,144},{574,151},{580,148},{587,154},{595,154},{599,161},{606,162},{608,158},{614,158},{61
1,164},{618,177},{609,189},{603,193},{606,199},{615,200},{615,205},{606,206},{605,217},{6
12,218},{616,226},{612,231},{617,237},{612,247},{612,259},{580,262},{578,273},{565,281},{
565,290},{562,294},{567,297},{561,309},{547,310},{540,312},{536,315},{533,321},{524,322},
{517,332},{480,340},{456,360},{447,372},{442,383},{442,390},{458,405},{456,412},{470,415}
,{484,408},{493,410},{497,403},{514,403},{514,412},{506,418},{509,427},{502,429},{484,430
},{478,427},{472,433},{462,443},{456,448},{440,455},{429,468},{413,472},{399,463},{405,44
0},{396,430},{382,421},{370,420},{366,416},{367,410},{377,402},{388,393},{406,387},{406,3
80},{396,374},{383,372},{357,373},{356,368},{340,363},{361,354},{345,343},{316,349},{306,
357},{304,363},{302,370},{289,384},{276,394},{266,393},{263,404},{268,421},{258,416},{250
,415},{240,421},{230,425},{218,421},{213,408},{221,400},{223,395},{234,388},{241,379},{24
1,371},{246,366},{240,354},{251,351},{251,358},{258,353},{264,360},{268,354},{271,361},{2
76,360},{276,360},{280,357},{274,350},{274,338},{263,328},{263,307},{256,307},{248,298},{
248,288},{251,276},{248,269},{243,272},{234,262},{228,262},{224,265},{217,257},{202,244},
{186,244},{175,245},{166,252},{145,266},{117,268},{112,277},{95,262},{73,262},{53,255},{5
0,248},{45,247},{37,254},{36,246},{27,245},{24,237},{17,234},{10,224}}, *gran=kordon;

        #include<stdio.h>
        struct OblastInfo{
         int x;
         int y;
         int f;
         int k;
        };
```

```
        /*zakarpatska */
        struct                                                          OblastInfo
OblastMas[1640]={{0,7},{41,193},{43,199},{46,204},{50,206},{52,209},{56,210},{60,214},{64
,214},{67,219},{71,220},{71,224},{75,225},{75,231},{79,228},{82,227},{82,236},{85,239},{8
7,236},{92,243},{94,248},{93,253},{96,257},{95,262},{73,262},{53,255},{50,248},{45,247},{
37,254},{36,246},{27,245},{24,237},{17,234},{10,224},{14,214},{29,188},{41,193},{0,2,51,2
27},
        /*lvov                                                          */
{41,193},{43,199},{46,204},{50,206},{52,209},{56,210},{60,214},{64,214},{64,207},{66,205}
,{65,200},{69,197},{69,193},{73,192},{77,194},{79,192},{82,193},{85,191},{87,193},{90,190
},{95,190},{95,187},{91,183},{95,180},{94,175},{95,171},{98,169},{101,172},{103,174},{106
,172},{106,168},{110,168},{114,165},{118,160},{119,155},{125,155},{128,143},{130,150},{13
0,146},{128,143},{124,138},{124,131},{122,126},{115,126},{110,127},{109,124},{105,122},{1
04,116},{103,112},{99,110},{96,112},{93,108},{88,108},{85,119},{76,119},{55,138},{36,164}
,{40,168},{38,185},{41,193},{0,14,85,164},
        /*ivan-fran                                                     */
{106,172},{106,177},{109,182},{107,186},{111,188},{108,192},{112,196},{110,200},{114,202}
,{117,204},{117,210},{121,208},{121,212},{127,211},{133,219},{134,224},{131,227},{131,241
},{122,240},{106,259},{107,267},{107,273},{112,277},{95,262},{96,257},{93,253},{94,248},{
92,243},{87,236},{85,239},{82,236},{82,227},{79,228},{75,231},{75,225},{71,224},{71,220},
{67,219},{64,214},{64,207},{66,205},{65,200},{69,197},{69,193},{73,192},{77,194},{79,192}
,{82,193},{85,191},{87,193},{90,190},{95,190},{95,187},{91,183},{95,180},{94,175},{95,171
},{98,169},{101,172},{103,174},{106,172},{0,4,99,211},
        /*chernivci                                                     */
{134,224},{139,225},{140,229},{147,226},{149,231},{153,233},{159,235},{164,239},{169,236}
,{170,231},{173,229},{176,233},{181,232},{191,230},{195,234},{198,239},{202,244},{202,244
},{186,244},{175,245},{166,252},{145,266},{117,268},{112,277},{107,273},{107,267},{106,25
9},{122,240},{131,241},{131,227},{134,224},{0,6,150,240},
        /*Hmelll                                                        */
{159,235},{153,225},{153,216},{152,210},{153,202},{153,193},{155,190},{155,182},{152,177}
,{157,168},{155,163},{157,160},{154,154},{154,148},{158,141},{158,137},{163,132},{167,132
},{176,120},{190,117},{193,124},{190,127},{193,131},{202,137},{205,141},{202,145},{205,14
8},{200,151},{200,156},{207,168},{207,174},{206,179},{208,185},{209,195},{211,198},{206,1
98},{205,201},{202,201},{199,199},{193,206},{193,225},{191,230},{181,232},{176,233},{173,
229},{170,231},{169,236},{164,239},{159,235},{0,5,172,176},
        /*ternopol                                                      */
{159,235},{153,225},{153,216},{152,210},{153,202},{153,193},{155,190},{155,182},{152,177}
,{157,168},{155,163},{157,160},{154,154},{154,148},{158,141},{158,137},{157,131},{154,132
},{150,134},{148,131},{146,135},{141,136},{134,137},{132,141},{128,142},{128,143},{130,14
6},{130,150},{128,153},{125,155},{119,155},{118,160},{114,165},{110,168},{106,168},{106,1
72},{106,177},{109,182},{107,186},{111,188},{108,192},{112,196},{110,200},{114,202},{117,
204},{117,210},{121,208},{121,212},{127,211},{133,219},{134,224},{139,225},{140,229},{147
,226},{149,231},{153,233},{159,235},{0,2,127,185},
        /*vinnnn                                                        */
{207,168},{211,167},{214,165},{217,168},{221,166},{226,168},{228,165},{233,164},{235,167}
,{240,166},{244,161},{247,164},{247,168},{250,171},{247,174},{249,179},{258,179},{263,176
},{266,180},{266,185},{263,189},{265,193},{269,196},{271,199},{270,204},{267,206},{266,20
9},{270,211},{267,215},{271,219},{271,224},{275,226},{275,230},{279,232},{278,236},{280,2
39},{278,243},{272,248},{271,257},{267,259},{267,263},{258,263},{254,264},{253,260},{250,
261},{248,258},{244,261},{239,264},{234,262},{224,265},{217,257},{202,244},{198,239},{195
,234},{191,230},{193,225},{193,206},{199,199},{202,201},{205,201},{206,198},{211,198},{20
9,195},{208,185},{206,179},{207,174},{207,168},{0,3,237,197},
        /*rivne                                                         */
{139,39},{153,39},{164,44},{170,44},{176,48},{181,48},{184,52},{194,52},{196,60},{209,63}
,{208,70},{205,72},{202,69},{201,72},{203,76},{197,83},{195,89},{191,89},{192,93},{191,10
4},{194,105},{194,109},{191,111},{190,117},{176,120},{167,132},{163,132},{158,137},{157,1
31},{154,132},{150,134},{148,131},{146,135},{141,136},{134,137},{132,141},{128,143},{124,
138},{124,131},{122,126},{124,124},{123,118},{129,118},{128,113},{132,114},{132,110},{135
,112},{141,108},{146,114},{149,103},{154,103},{152,99},{152,95},{156,94},{156,89},{151,88
},{151,85},{156,85},{156,82},{151,75},{148,74},{148,70},{144,70},{142,64},{142,55},{146,4
9},{146,41},{139,39},{0,4,178,102},
        /*luckk                                                         */
{88,108},{93,108},{96,112},{99,110},{103,112},{104,116},{105,122},{109,124},{110,127},{11
5,126},{122,126},{124,124},{123,118},{129,118},{128,113},{132,114},{132,110},{135,112},{1
41,108},{146,114},{149,103},{154,103},{152,99},{152,95},{156,94},{156,89},{151,88},{151,8
5},{156,85},{156,82},{151,75},{148,74},{148,70},{144,70},{142,64},{142,55},{146,49},{146,
41},{139,39},{131,36},{102,37},{101,44},{91,52},{86,46},{79,49},{77,70},{82,78},{85,85},{
90,95},{86,97},{88,108},{0,4,118,92},
```

```
        /*odesssa                                                                 */
{271,257},{275,258},{278,256},{281,262},{290,262},{286,266},{287,273},{289,280},{293,283}
,{292,288},{294,293},{299,292},{302,294},{305,291},{307,297},{307,303},{310,305},{309,310
},{314,315},{319,314},{317,320},{321,319},{323,326},{321,330},{313,331},{318,339},{319,34
5},{316,349},{306,357},{304,363},{302,370},{289,384},{276,394},{266,393},{263,404},{268,4
21},{258,416},{250,415},{240,421},{230,425},{218,421},{213,408},{221,400},{223,395},{234,
388},{241,379},{241,371},{246,366},{240,354},{251,351},{251,358},{258,353},{264,360},{268
,354},{271,361},{276,360},{276,360},{280,357},{274,350},{274,338},{263,328},{263,307},{25
6,307},{248,298},{248,288},{251,276},{248,269},{243,272},{234,262},{239,264},{244,261},{2
48,258},{250,261},{253,260},{254,264},{258,263},{267,263},{267,259},{271,257},{0,2,269,37
0},
        /*mikolay                                                                 */
{290,262},{296,259},{300,262},{303,258},{309,261},{312,257},{317,257},{322,262},{328,262}
,{331,266},{334,262},{339,264},{340,267},{336,268},{337,273},{342,273},{342,278},{340,281
},{343,284},{346,281},{350,284},{355,284},{358,282},{368,281},{368,274},{372,274},{371,27
1},{376,271},{376,267},{381,268},{381,274},{384,282},{380,286},{381,292},{387,295},{386,3
00},{384,304},{386,312},{379,320},{384,326},{383,329},{379,329},{381,333},{377,334},{376,
337},{366,335},{364,339},{356,338},{353,342},{345,343},{316,349},{319,345},{318,339},{313
,331},{321,330},{323,326},{321,319},{317,320},{319,314},{314,315},{309,310},{310,305},{30
7,303},{307,297},{305,291},{302,294},{299,292},{294,293},{292,288},{293,283},{289,280},{2
87,273},{286,266},{290,262},{0,6,335,323},
        /*kiev                                                                    */
{263,176},{266,180},{266,185},{263,189},{265,193},{269,196},{271,199},{274,201},{277,198}
,{279,193},{283,195},{286,193},{289,197},{292,196},{294,191},{297,192},{300,190},{303,192
},{306,190},{310,191},{310,187},{314,184},{317,179},{319,174},{320,169},{318,164},{322,16
2},{326,157},{328,162},{332,162},{337,164},{340,159},{343,160},{343,156},{343,152},{344,1
48},{350,144},{351,135},{346,130},{348,128},{348,125},{340,119},{335,127},{330,125},{325,
128},{319,124},{322,122},{321,117},{317,111},{305,111},{306,107},{303,103},{302,97},{297,
96},{298,90},{300,83},{295,82},{291,72},{282,71},{279,75},{275,75},{272,69},{265,76},{260
,76},{258,82},{259,89},{265,93},{265,98},{262,100},{266,106},{266,109},{263,111},{268,113
},{264,117},{263,125},{264,131},{267,132},{271,138},{271,143},{268,146},{271,149},{269,15
2},{271,156},{268,158},{269,161},{261,165},{261,169},{264,171},{263,176},{0,13,290,140},
        /*Gitomir                                                                 */
{263,176},{258,179},{249,179},{247,174},{250,171},{247,168},{247,164},{244,161},{240,166}
,{235,167},{233,164},{228,165},{226,168},{221,166},{217,168},{214,165},{211,167},{207,168
},{200,156},{200,151},{205,148},{202,145},{205,141},{202,137},{193,131},{190,127},{190,11
7},{191,111},{194,109},{194,105},{191,104},{192,93},{191,89},{195,89},{197,83},{203,76},{
201,72},{202,69},{205,72},{208,70},{214,69},{214,63},{220,65},{225,61},{228,66},{239,65},
{239,72},{243,72},{246,65},{254,61},{260,76},{258,82},{259,89},{265,93},{265,98},{262,100
},{266,106},{266,109},{263,111},{268,113},{264,117},{263,125},{264,131},{267,132},{271,13
8},{271,143},{268,146},{271,149},{269,152},{271,156},{268,158},{269,161},{261,165},{261,1
69},{264,171},{263,176},{0,3,230,130},
        /*kirovog                                                                 */
{278,243},{283,243},{285,240},{288,242},{292,239},{292,235},{297,237},{299,235},{297,233}
,{300,225},{312,226},{316,228},{320,225},{325,228},{332,223},{332,217},{339,215},{344,219
},{347,216},{350,219},{351,214},{354,207},{358,206},{359,210},{363,211},{363,215},{367,21
4},{369,211},{372,214},{375,211},{373,204},{371,201},{372,198},{378,198},{386,206},{390,2
11},{398,216},{402,212},{407,212},{409,217},{408,223},{400,223},{406,229},{401,235},{395,
236},{396,246},{398,255},{393,260},{388,259},{389,263},{385,263},{386,267},{381,268},{376
,267},{376,271},{371,271},{372,274},{368,274},{368,281},{358,282},{355,284},{350,284},{34
6,281},{343,284},{340,281},{342,278},{342,273},{337,273},{336,268},{340,267},{339,264},{3
34,262},{331,266},{328,262},{322,262},{317,257},{312,257},{309,261},{303,258},{300,262},{
296,259},{290,262},{281,262},{278,256},{275,258},{271,257},{272,248},{278,243},{0,13,350,
250},
        /*Xerson                                                                  */
{419,300},{425,305},{426,310},{429,310},{429,306},{433,307},{434,310},{433,313},{433,318}
,{436,325},{442,328},{442,332},{445,332},{445,337},{442,337},{442,340},{438,342},{443,344
},{444,347},{440,348},{441,351},{447,351},{449,354},{454,354},{456,360},{447,372},{438,36
8},{428,369},{422,363},{423,369},{418,372},{415,367},{404,366},{402,371},{396,374},{383,3
72},{357,373},{356,368},{340,363},{361,354},{345,343},{353,342},{356,338},{364,339},{366,
335},{376,337},{377,334},{381,333},{379,329},{383,329},{384,326},{379,320},{386,312},{384
,304},{386,300},{387,295},{392,299},{400,293},{404,299},{413,297},{419,300},{0,3,390,350}
,
        /*Krim                                                                    */
{447,372},{442,383},{442,390},{458,405},{456,412},{470,415},{484,408},{493,410},{497,403}
,{514,403},{514,412},{506,418},{509,427},{502,429},{484,430},{478,427},{472,433},{462,443
},{456,448},{440,455},{429,468},{413,472},{399,463},{405,440},{396,430},{382,421},{370,42
0},{366,416},{367,410},{377,402},{388,393},{406,387},{406,380},{396,374},{402,371},{404,3
66},{415,367},{418,372},{423,369},{422,363},{428,369},{438,368},{447,372},{0,6,420,410},
```

```
        /*ZZZapor                                                                 */
{419,300},{433,292},{446,291},{447,280},{442,273},{446,269},{442,264},{443,260},{452,260}
,{460,258},{467,260},{473,257},{476,260},{481,258},{484,260},{485,267},{488,269},{486,273
},{491,273},{497,273},{502,272},{503,275},{506,275},{507,280},{511,285},{515,288},{517,28
6},{524,298},{521,302},{517,302},{518,308},{514,309},{525,317},{524,322},{517,332},{480,3
41},{456,360},{454,354},{449,354},{447,351},{441,351},{440,348},{444,347},{443,344},{438,
342},{442,340},{442,337},{445,337},{445,332},{442,332},{442,328},{436,325},{433,318},{433
,313},{434,310},{433,307},{429,306},{429,310},{426,310},{425,305},{419,300},{0,2,470,310}
,
        /*Harkov                                                                  */
{445,197},{448,199},{452,198},{453,202},{461,207},{473,210},{477,206},{481,216},{485,222}
,{489,225},{491,228},{489,232},{492,233},{496,224},{500,227},{504,226},{504,220},{502,217
},{504,214},{508,216},{513,215},{515,211},{519,212},{519,207},{523,206},{522,202},{531,19
4},{528,191},{532,187},{537,189},{541,187},{542,183},{539,181},{540,176},{537,172},{540,1
66},{543,161},{544,158},{539,158},{543,154},{541,151},{543,148},{527,135},{520,123},{504,
130},{496,137},{492,133},{484,133},{474,126},{462,131},{457,133},{453,134},{452,138},{447
,138},{441,143},{439,148},{439,153},{444,158},{450,159},{450,163},{451,167},{458,168},{45
9,172},{459,178},{454,179},{457,182},{456,186},{453,191},{446,190},{443,193},{445,197},{0
,3,500,180},
        /*Doneck                                                                  */
{541,187},{543,192},{547,191},{547,196},{549,199},{546,203},{554,202},{555,208},{558,209}
,{558,216},{555,219},{558,225},{560,230},{564,230},{566,239},{569,242},{570,245},{578,250
},{578,253},{585,253},{587,261},{580,262},{578,273},{565,281},{565,290},{562,294},{567,29
7},{561,309},{547,310},{540,312},{536,315},{533,321},{524,322},{525,317},{514,309},{518,3
08},{517,302},{521,302},{524,298},{517,286},{515,288},{511,285},{507,280},{506,275},{503,
275},{502,272},{501,268},{506,267},{502,263},{504,256},{512,257},{514,251},{513,247},{508
,243},{509,235},{510,229},{504,226},{504,220},{502,217},{504,214},{508,216},{513,215},{51
5,211},{519,212},{519,207},{523,206},{522,202},{531,194},{528,191},{532,187},{537,189},{5
41,187},{0,4,540,260},
        /*Lugansk                                                                 */
{541,187},{542,183},{539,181},{540,176},{537,172},{540,166},{543,161},{544,158},{539,158}
,{543,154},{541,151},{543,148},{547,140},{552,140},{557,144},{574,151},{580,148},{587,154
},{595,154},{599,161},{606,162},{608,158},{614,158},{611,164},{618,177},{609,189},{603,19
3},{606,199},{615,200},{615,205},{606,206},{605,217},{612,218},{616,226},{612,231},{617,2
37},{612,247},{612,259},{587,261},{585,253},{578,253},{578,250},{570,245},{569,242},{566,
239},{564,230},{560,230},{558,225},{555,219},{558,216},{558,209},{555,208},{554,202},{546
,203},{549,199},{547,196},{547,191},{543,192},{541,187},{0,8,590,212},
        /*Chernigiv                                                               */
{351,135},{355,135},{358,131},{363,135},{373,134},{381,125},{381,118},{386,111},{382,106}
,{385,103},{384,99},{379,98},{380,94},{376,92},{375,87},{379,83},{380,73},{384,67},{381,6
1},{383,57},{380,53},{382,50},{382,45},{389,44},{389,40},{386,37},{392,36},{392,33},{387,
29},{389,22},{379,21},{375,27},{364,27},{354,23},{355,31},{352,38},{346,41},{342,37},{327
,35},{321,39},{312,40},{302,52},{298,62},{300,74},{300,83},{298,90},{297,96},{302,97},{30
3,103},{306,107},{305,111},{317,111},{321,117},{322,122},{319,124},{325,128},{330,125},{3
35,127},{340,119},{348,125},{348,128},{346,130},{351,135},{0,9,340,80},
        /*Chrkasy                                                                 */
{271,199},{274,201},{277,198},{279,193},{283,195},{286,193},{289,197},{292,196},{294,191}
,{297,192},{300,190},{303,192},{306,190},{310,191},{310,187},{314,184},{317,179},{319,174
},{320,169},{318,164},{322,162},{326,157},{328,162},{332,162},{337,164},{340,159},{343,16
0},{343,156},{343,152},{344,148},{350,144},{355,148},{354,151},{361,159},{360,164},{365,1
72},{369,175},{369,184},{368,189},{361,189},{361,192},{367,197},{372,198},{371,201},{373,
204},{375,211},{372,214},{369,211},{367,214},{363,215},{363,211},{359,210},{358,206},{354
,207},{351,214},{350,219},{347,216},{344,219},{339,215},{332,217},{332,223},{325,228},{32
0,225},{316,228},{312,226},{300,225},{297,233},{299,235},{297,237},{292,235},{292,239},{2
88,242},{285,240},{283,243},{278,243},{280,239},{278,236},{279,232},{275,230},{275,226},{
271,224},{271,219},{267,215},{270,211},{266,209},{267,206},{270,204},{271,199},{0,10,340,
190},
        /*Dniproper                                                               */
{387,295},{392,299},{400,293},{404,299},{413,297},{419,300},{433,292},{446,291},{447,280}
,{442,273},{446,269},{442,264},{443,260},{452,260},{460,258},{467,260},{473,257},{476,260
},{481,258},{484,260},{485,267},{488,269},{486,273},{491,273},{497,273},{502,272},{501,26
8},{506,267},{502,263},{504,256},{512,257},{514,251},{513,247},{508,243},{509,235},{510,2
29},{504,226},{500,227},{496,224},{492,233},{489,232},{491,228},{489,225},{485,222},{481,
216},{477,206},{473,210},{461,207},{453,202},{452,198},{448,199},{445,197},{442,199},{438
,199},{432,202},{430,206},{425,204},{423,208},{423,216},{424,224},{415,219},{409,217},{40
8,223},{400,223},{406,229},{401,235},{395,236},{396,246},{398,255},{393,260},{388,259},{3
89,263},{385,263},{386,267},{381,268},{381,274},{384,282},{380,286},{381,292},{387,295},{
0,4,440,230},
```

```
        /*Poltava                                                                 */
{409,217},{415,219},{424,224},{423,216},{423,208},{425,204},{430,206},{432,202},{438,199}
,{442,199},{445,197},{443,193},{446,190},{453,191},{456,186},{457,182},{454,179},{459,178
},{459,172},{458,168},{451,167},{450,163},{450,159},{444,158},{439,153},{439,148},{441,14
3},{433,144},{429,143},{428,136},{423,136},{423,131},{416,123},{413,126},{408,122},{405,1
24},{403,128},{392,126},{387,123},{385,126},{381,125},{373,134},{363,135},{358,131},{355,
135},{351,135},{350,144},{355,148},{354,151},{361,159},{360,164},{365,172},{369,175},{369
,184},{368,189},{361,189},{361,192},{367,197},{372,198},{378,198},{386,206},{390,211},{39
8,216},{402,212},{407,212},{409,217},{0,12,390,160},
        /*Sumi                                                                    */
{381,125},{381,118},{386,111},{382,106},{385,103},{384,99},{379,98},{380,94},{376,92},{37
5,87},{379,83},{380,73},{384,67},{381,61},{383,57},{380,53},{382,50},{382,45},{389,44},{3
89,40},{386,37},{392,36},{392,33},{387,29},{389,22},{392,26},{396,21},{407,29},{409,39},{
416,49},{420,56},{409,61},{417,69},{415,75},{419,79},{416,84},{425,83},{430,88},{442,85},
{445,93},{452,93},{449,98},{455,107},{454,119},{462,131},{457,133},{453,134},{452,138},{4
47,138},{441,143},{433,144},{429,143},{428,136},{423,136},{423,131},{416,123},{413,126},{
408,122},{405,124},{403,128},{392,126},{387,123},{385,126},{381,125},{0,0,400,95}},
*obls=OblastMas;

        //
        #include<stdio.h>
        struct  Dnipe{
         int x;
         int y;
        };
        struct                                                                  Dnipe
DniproPoi[85]={{0,0},{362,354},{371,350},{383,345},{389,340},{398,333},{403,327},{406,322
},{413,308},{418,304},{435,295},{445,296},{457,293},{454,281},{448,289},{436,289},{426,29
2},{415,293},{413,308},{418,304},{435,295},{445,296},{457,293},{454,281},{454,271},{450,2
59},{452,254},{453,249},{451,243},{446,235},{431,234},{428,228},{418,217},{409,216},{415,
223},{423,229},{431,234},{428,228},{418,217},{409,216},{399,208},{386,204},{378,196},{370
,192},{361,188},{347,180},{337,175},{332,169},{330,165},{329,155},{320,154},{308,144},{29
9,137},{302,147},{311,156},{319,159},{325,159},{326,166},{330,171},{334,177},{345,185},{3
53,190},{368,203},{385,207},{399,208},{386,204},{378,196},{370,192},{361,188},{347,180},{
337,175},{332,169},{330,165},{329,155},{320,154},{308,144},{299,137},{298,128},{299,112},
{300,95},{299,84},{294,83},{292,93},{292,110},{298,128}}, *rive=DniproPoi;

        //
        #include<stdio.h>
        struct   im{
         char ck[25];
         int  nam;
        };
        struct im name[236]={{"",0},
        {"Chop",1},{"Uzhgorod",2},{"Mukachevo",4},{"Beregovo",5},{"Khust",8},{"Tjachev",1
8},{"Rakhov",20},
        {"Jasinja",21},{"Jaremchja",22},{"Kolomyja",26},{"Kosov",25},{"Ivano-
Frankovsk",28},{"Galych",34},
        {"Kalush",32},{"Chernovtzy",140},{"Storozhinetz",145},{"Seret",144},{"Rogatin",35
},{"Khodorov",37},
        {"Stryjj",38},{"Drogobych",39},{"Sambor",42},{"Mostirka",44},{"na
Krakov",49},{"Javorov",48},
        {"L'vov",46},{"na                                                  Ljublin",59},{"Rava-
Rus'ka",58},{"Busk",53},{"Chervonograd",60},{"Radekhov",61},
        {"Brody",63},{"Gorokhov",67},{"Ustilug",70},{"na
Ljublin",84},{"Shatzk",89},{"Kovel'",82},
        {"na                        Brest",92},{"na                          Stolin",94},{"na
Stolin",98},{"Sarny",96},{"Manevichi",93},{"Lutzk",72},
        {"Rovno",75},{"Novograd-
Volynsk",108},{"Dubno",73},{"Kremenetz",76},{"Ostrog",103},
        {"Shepetovka",105},{"Berdichev",680},{"Starokonstantinov",115},{"Khmel'nitzkijj",
116},
        {"Gorodok",121},{"Chortkov",128},{"Terebovlja",127},{"Ternopol'",126},{"Borschev"
,130},
        {"Kamenetz-Podol'skijj",134},{"Khotin",135},{"Ushitza",133},{"Mogilev-
Podol'sk",146},
        {"Zhmerinka",149},{"Jampol'",156},{"Soroka",191},{"Tul'chin",159},{"Ladyzhin",165
},
        {"Nemirov",161},{"Gajjsin",166},{"Bershad'",187},{"Balta",197},{"Uman'",184},
```

```
        {"Vinnitza",150},{"Ruzhin",169},{"Skvira",174},{"Tetiev",177},{"Zhashkov",178},
        {"Tal'noe",183},{"Zvenigorodka",181},{"Mironovka",256},{"Kagarlyk",257},{"Belaja
Tzerkov'",175},
        {"Ovruch",670},{"Korosten'",672},{"Zhitomir",663},{"Korostyshev",662},{"Radomishl
'",661},
        {"Malin",665},{"Borodjanka",667},{"Kiev",624},{"Brovary",625},{"Borispol'",619},{
"Lokhvitza",589},
        {"Lubny",603},{"Orzhitza",602},{"Zolotonosha",262},{"Kanev",260},{"Perejaslav-
Khmel'",621},
        {"Obukhov",622},{"Fastov",657},{"Jagotin",611},{"Pirjatin",606},{"Priluki",607},{
"Mirgorod",592},
        {"Khorol",600},{"Poltava",567},{"Kremenchug",267},{"Cherkassy",264},{"Smela",253}
,{"Shpola",252},
        {"Reshetilovka",570},{"Znamenka",275},{"Oleksandrija",277},{"Kirovograd",276},{"K
rivoe Ozero",196},
        {"Pervomajjsk",239},{"Golovanevsk",195},{"Kotovsk",199},{"Anan'ev",202},{"na
Kishinev",204},
        {"Vel.Mikhajjlovka",206},{"Berezovka",235},{"Voznesensk",236},{"Odessa",214},{"I'
ichevsk",215},
        {"Beleevka",217},{"Belgorod-
Dnestrovsk",219},{"Artziz",222},{"Tatarbunary",223},{"Kilija",231},
        {"Bolgrad",227},{"Izmail",228},{"Reni",230},{"NIKOLAEV",299},{"KHERSON",303},{"No
vaja Kahovka",350},
        {"Snigirevka",352},{"Bashtanka",297},{"Novyjj
Bug",296},{"Krasnoperekopsk",311},{"Genichesk",341},
        {"Dzhankojj",339},{"Chernomorskoe",314},{"Evpatorija",317},{"SIMFEROPOL'",323},{"
SEVASTOPOL'",321},
        {"Jalta",325},{"Alushta",326},{"Sudak",329},{"Feodosija",331},{"Kerch'",334},{"na
Novorosijjsk",335},
        {"Krivojj
Rog",292},{"Apostolovo",364},{"Nikopol'",375},{"Marganetz",374},{"Ehnergodar",377},
        {"Melitopol'",386},{"Tokmak",398},{"Vasil'evka",394},{"Orekhov",400},{"Pologi",42
5},
        {"Guljajjpole",424},{"ZAPOROZH'E",404},{"DNEPROPETROVSK",415},{"Novomoskovsk",416
},{"Pavlograd",419}
        {"Lozovaja",539},{"Berdjansk",430},{"Mariupol'",435},{"na
Taganrog",457},{"Volnovakha",451},
        {"na
Taganrog",466},{"Amvrosievka",467},{"Torez",469},{"Makeevka",476},{"DONETZK",447},
        {"Krasnoarmejjsk",444},{"Gorlovka",478},{"Kramatorsk",487},{"Slavjansk",488},{"Ar
temovsk",479},
        {"Debal'tzevo",474},{"Stakhanov",490},{"Alchevsk",491},{"Krasnyjj
Luch",471},{"LUGANSK",494},
        {"Lisichansk",503},{"Schast'e",498},{"Krasnodon",492},{"na                Kamensk-
Shakhtinskijj",493},
        {"Sverdlovsk",473},{"na                                                  Rostv-na-
Donu",472},{"Izjum",522},{"Kupjansk",526},{"Svatovo",506},
        {"Starobel'sk",507},{"Melovoe",511},{"Kremennaja",505},{"Rubezhnoe",504},{"Chugue
v",547},
        {"Zmiev",545},{"Balaklija",534},{"Pervomajjsk",543},{"Krasnograd",562},{"Valki",5
58},
        {"KHAR'KOV",555},{"Troitzkoe",517},{"Volchanok",531},{"Belgorod",679},{"Bogodukho
v",573},
        {"Trostjanetz",579},{"Akhtyrka",575},{"Zen'kov",578},{"Gadjach",588},{"Lebedin",5
87},{"SUMY",581},
        {"na
Kursk",583},{"Belopol'e",584},{"Romny",608},{"Glukhov",639},{"Krolevetz",641},{"Konotop",
642},
        {"Bakhmach",644},{"Ichnja",649},{"Nezhin",651},{"Sosnitza",635},{"CHERNIGOV",634}
,{"Druzhba",640},
        {"Shostka",638},{"Novgorod-
Siverskijj",637},{"Semenovka",636},{"Gorodnja",633},{"Repki",629},
        {"na Gomel'",632},{"Slavutich",631}},
         *gorod=name;

        //
        #include <dos.h>
        #include <conio.h>

        struct MouseState
```

```
 {
  int x,y;
  int buttons;
 };
int  ResetMouse(void);
void ShowMouseCursor(void);
void HideMouseCursor(void);
void MoveMouseCursor(int x,int y);
void ReadMouseState(struct MouseState *s);
void SetMouseHoriz(int min, int max);
void SetMouseVert(int min, int max);
void SetMouseShape(int mask[32], char x, char y);
int  MouseInArea(int x1, int y1, int x2, int y2);
void SetMouseArea(int x1, int y1, int x2, int y2);
void ReadMouse( int* Px,int* Py,int *but);

int ResetMouse()
 {
  _asm {
  xor ax,ax
  int 0x33
 }
 return (_AX==0xFFFF);
}
void ShowMouseCursor()
 {
  _asm {
  mov ax, 1
  int 0x33
 }
}

void HideMouseCursor()
 {
  _asm {
  mov ax, 2
  int 0x33
 }
}
void MoveMouseCursor(int x, int y)
 {
  _asm {
  mov ax, 4
  mov cx, x
  mov dx, y
  int 0x33
 }
}
void ReadMouseState(struct MouseState *s)
 {
  _asm{
  mov ax,3
  int 0x33
 }
 #if defined(__COMPACT__) || defined(__LARGE__) || defined(__HUGE__) _asm
  {
   push es
   push di
   les di,dword ptr s
   mov es:[di],cx // X
   mov es:[di+2],dx // Y
   mov es:[di+4],bx // button
   pop di
   pop es
  }
  #else
   _asm{
   push di
   mov di,word ptr s
   mov [di],cx
```

```
   mov [di+2],dx
   mov [di+4],bx
   pop di
  }
  #endif
}
void SetMouseHoriz(int min,int max)
 {
  _asm{
  mov ax,7
  mov cx,min
  mov dx,max
  int 0x33
 }
}
void SetMouseVert(int min,int max)
 {
  _asm{
  mov ax,8
  mov cx,min
  mov dx,max
  int 0x33
 }
}

void SetMouseShape(int mask[32],char x,char y)
 {
  union REGS inregs, outregs;
  struct SREGS segregs;
  inregs.x.ax = 0x9;
  inregs.x.bx = x;
  inregs.x.cx = y;
  inregs.x.dx = FP_OFF(mask);
  segregs.es  = FP_SEG(mask);
  int86x(0x33, &inregs, &outregs, &segregs);
 }
int MouseInArea(int x1, int y1, int x2, int y2)
 {
  struct MouseState ms;
  ReadMouseState(&ms);
  if (ms.x>=x1 && ms.x<=x2 && ms.y>=y1 && ms.y<=y2)
   return(1);
  else
   return(0);
}
void SetMouseArea(int x1,int y1,int x2,int y2)
 {
  SetMouseHoriz(x1,x2);
  SetMouseVert(y1,y2);
 }
void ReadMouse( int* Px,int* Py,int *but)
 {
  union REGS r;

  r.x.ax = 0x03;
  int86(0x33,&r,&r);
  *Px  = r.x.cx;
  *Py  = r.x.dx;
  *but = r.x.bx;
}
int mc_cross[32]={
 0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,
 0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,
 0xFFFF,0xFFFF,0xFFFF,0xFFFF,                          // AND
 0x0100,0x0100,0x0100,0x0000,0x0100,0x0100,0x0000,
 0xED6E,0x0000,0x0100,0x0100,0x0000,0x0100,0x0100,
 0x0100,0x0000                                        // XOR
};
int mc_arrow[32]={
 0x3FFF,0x1FFF,0x0FFF,0x07FF,0x03FF,
```

```
 0x01FF,0x00FF,0x007F,0x003F,0x01FF,
 0x20FF,0x707F,0xF03F,0xF83F,0xFC3F,0xFE7F,       // AND
 0x0000,0x4000,0x6000,0x7000,0x7800,
 0x7C00,0x7E00,0x7F00,0x7800,0x4800,
 0x0C00,0x0600,0x0700,0x0380,0x0180,              // XOR
};

int mc_rect[32]={
 0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,
 0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,
 0xFFFF,0xFFFF,0xFFFF,0xFFFF,0xFFFF,
 0xFFFF,                                          // AND
 0x0000,0x0000,0x3FFC,0x2004,0x2004,
 0x2004,0x2004,0x2004,0x2004,0x2004,
 0x2004,0x2004,0x2004,0x3FFC,0x0000,
 0x0000                                           // XOR
};
```

Вихідні форми навчального програмного продукту

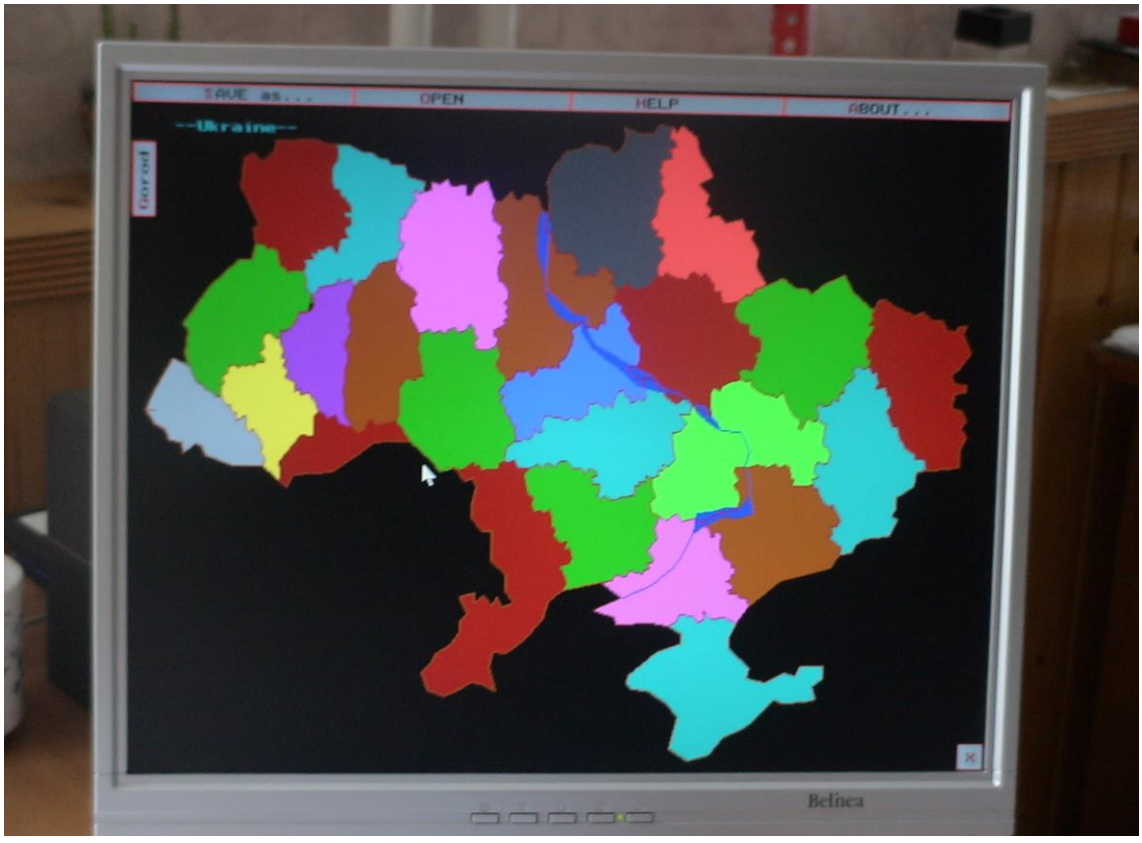


Рис. Г1. Мапа України.



Рис. Г2. Список міст України

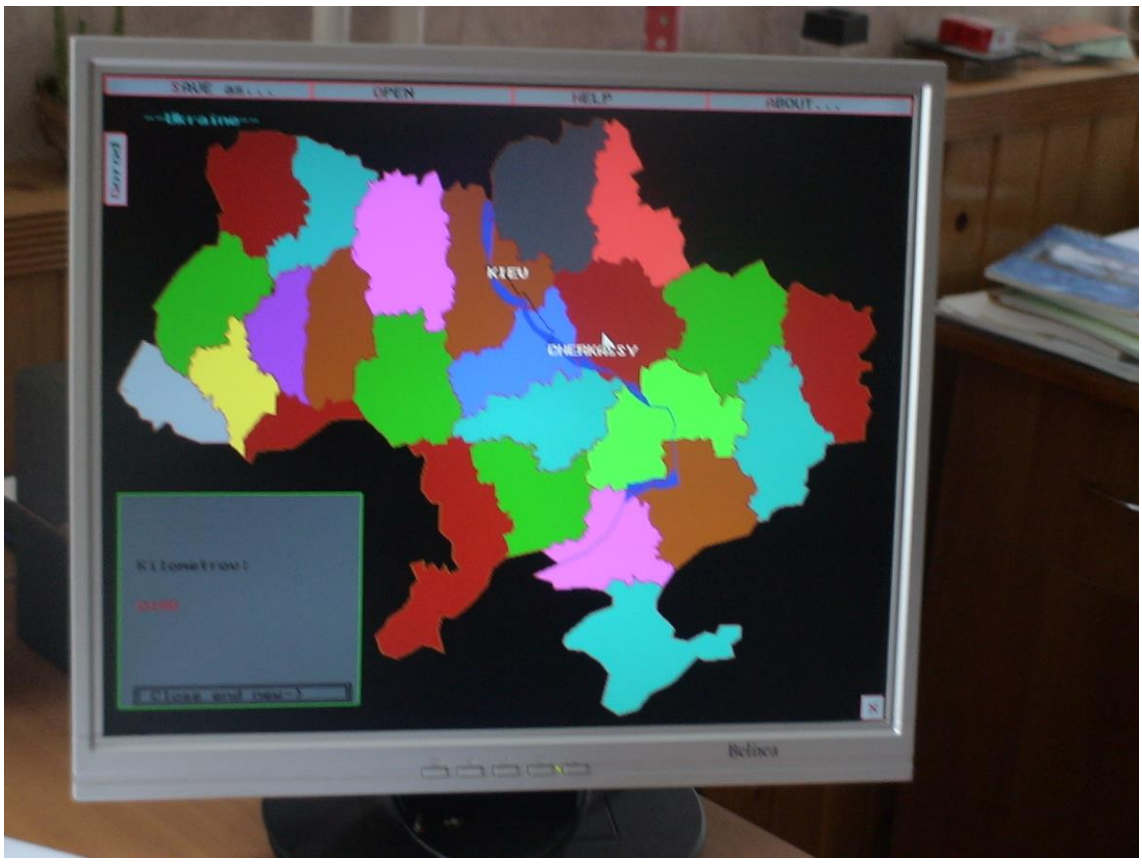


Рис. Г3. Шлях та відстань між місцями Київ – Черкаси.

ДОДАТОК Д

Анкета
Для студентів 1-го курсу факультету інформаційних технологій та біомедичної кібернетики спеціальності „Програмне забезпечення автоматизованих систем”

1. Яку мову програмування Ви вивчали до вступу в університет:
   - ☐ Basic
   - ☐ Pascal
   - ☐ C
   - ☐ алгоритмічну навчальну мову
   - ☐ не вивчав програмування
   - ☐ інше_____

2. Чи займались Ви інформатикою додатково (поза школою):
   - ☐ так
   - ☐ ні

3. Чи доцільно вивчати мову програмування Pascal в курсі „Основи програмування та алгоритмічні мови”:

   так, тому, що:
   - ☐ я не вивчав її раніше
   - ☐ вона більш зрозуміла при навчанні
   - ☐ мова С складна, для початку треба вивчати Pascal
   - ☐ інше

   ні, тому, що
   - ☐ я вивчав мову програмування Pascal у школі
   - ☐ я вивчав мову програмування Pascal на підготовчих курсах
   - ☐ для майбутніх інженерів-програмістів потрібна мова С, як основа мов програмування C++, C#, Java
   - ☐ інше_____

4. Чи вважаєте Ви доцільним паралельне вивчення мов програмування Pascal та С в курсі «Основи програмування та алгоритмічні мови»:

   так, тому, що
   - ☐ так легше вивчати мову С
   - ☐ це дає можливість робити порівняльний аналіз двох мов програмування і визначати позитивні і негативні їх риси
   - ☐ це сприяє формуванню професійних якостей
   - ☐ мов програмування Pascal мені вже знайома і паралельне викладання систематизує мої знання з мови програмування Pascal і полегшує вивчення мови С

- ☐ Pascal та C схожі між собою
- ☐ при послідовному вивченні (Pascal, а потім C) на опанування програмуванням мовою C залишається мало часу
- ☐ інше_____

ні, тому, що

- ☐ послідовне вивчення (Pascal, а потім C) для мене простіше
- ☐ вважаю вивчення мови Pascal в університеті недоцільним, оскільки вона вивчалась раніше
- ☐ інше_____

5. Чи вважаєте Ви, що знання декількох мов програмування сприяє формуванню Вас як майбутнього програміста:

- ☐ так
- ☐ ні

6. На екзамені з курсу „Основи програмування та алгоритмічні мови” Ви будете використовувати мову програмування Pascal тому, що

- ☐ яка мова програмування відома Вам зі школи
- ☐ більше подобається
- ☐ за звичкою
- ☐ немає різниці якою мовою писати програму
- ☐ не подобається мова програмування C
- ☐ інше_____

будете використовувати мову програмування C тому, що

- ☐ більше подобається
- ☐ приємно усвідомлювати, що можу написати програму на C
- ☐ немає різниці якою мовою писати програму
- ☐ хочу продемонструвати свої знання з мови програмування C
- ☐ хочу перевірити свої знання з мови програмування C
- ☐ інше_____

Дата_____група___

_____

підпис

ДОДАТОК Е

Аналіз підручників, навчальних посібників та збірників задач і вправ з основ програмування

ТаблицяЕ.1

Аналіз підручників з основ програмування

№	Назва:	Как программировать на С							
1.	Автор(и):	Дейтел Х., Дейтел П.							
	Зміст:	Глава 1. Принципы машинной обработки данных Глава 2. Введение в програмировании на С Глава 3. Структурная разработка программ Глава 4. Руководство программой Глава 5. Функции Глава 6. Масивы Глава 7. Указатели Глава 8. Символы и строки Глава 9. Формирование ввода/вывода Глава 10. Структуры, объединения, операции с битами и пересчет Глава 11. Работа с файлами Глава 12. Структуры данных Глава 13. Препроцессор Глава 14. Специальные вопросы Глава 15. С++ как "улучшенный" С Глава 16. Классы и абстракции данных Глава 17. Классы: часть II Глава 18. Перегрузка операций Глава 19. Наследование Глава 20. Виртуальные функции и полиморфизм Глава 21. Потоки ввода/вывода в С++ Приложение А. Синтаксис С Приложение Б. Стандартная библиотека Приложение В. Таблица приоритетов операций Приложение Г. Набор символов ASCII Приложение Д. Системы счисления							
	Базова мова програ мува ння	Пора ди з прог раму вання	Пита ння для само конт ролю	Впра ви	Завдан ння на програ мува ння	Розпо діл завда нь за рівня ми склад ності	Кількі сть зав дань із теми (доста тньо/ недост атньо для засвоє ння навч. мате ріалу)	Відпо віді та розв'я зки	Примітка
	1	2	3	4	5	6	7	8	9
	Псевдо код, С	+	+	+	+	-	+	+	рос. мовою; пер. с англ.
№	Назва:	Основи програмування. – К.: Видавнича група BHV, 2005. – 384 с.							
2.	Автор(и):	Ковалюк Т.В.							
	Зміст:	Частина 1. Організація програм Розділ 1. Основні поняття та означення Розділ 2. Елементи мови Pascal Розділ 3. Керування порядком обчислень Розділ 4. Процедурно-орієнтоване програмування Розділ 5. Програмування графіки Розділ 6. Методології розробки програм							

		Частина 2. Організація даних Розділ 7. Масиви Розділ 8. Записи та множини Розділ 9. Файлові структури даних Розділ 10. Динамічні структури даних Розділ 11. Алгоритми на графах Задачи підвищеної складності

1	2	3	4	5	6	7	8	9
Pascal	+	+	+	+	+	+	-	

№	Назва:	Алгоритмы: построение и анализ. М., 1999. – 960 с.
3.	Автор(и):	Кормен Т., Лейзерсон Ч., Ривест Р.
	Зміст:	1. Введение I. Математические основы анализа алгоритмов Введение 2. Скорость роста функций 3. Суммирование 4. Рекурентные соотношения 5. Множества 6. Комбинаторика и вероятность II. Сортировка и порядковые статистики Введение 7. Сортировка с помощью кучи 8. Быстрая сортировка 9. Сортировка за линейное время 10. Медианы и порядковые статистики III. Структуры данных 11. Элементарные структуры данных 12. Хеш-таблицы 13. Двоичные деревья поиска 14. Красно-черные деревья 15. Пополнение структур данных IV. Методы построения и анализа алгоритмов Введение 16. Динамическое программирование 17. Жадные алгоритмы 18. Амортизационный анализ V. Более сложные структуры данных Введение 19. Б-деревья 20. Биномиальные кучи 21. Фибоначчиевы кучи 22. Системы непересекающихся множеств VI. Алгоритмы на графах Введение 23. Основные алгоритмы на графах 24. Минимальные покрывающие деревья 25. Кратчайшие пути из одной вершины 26. Кратчайшие пути для всех пар вершин 27. Максимальный поток Дополнительные главы Введение 28. Сортирующие сети 29. Арифметические схемы 30. Алгоритмы параллельных вичислений 31. Матрицы и действия с ними 32. Многочлены и быстрое преобразование Фур’є 33. Теоретико-числовые алгоритмы 34. Поиск подстрок 35. Вычислительная геометрия 36. NP-полнота 37. Приближенные алгоритмы

	1	2	3	4	5	6	7	8	9
	Псевдо код	+	+	+	+	+	+	-	рос. мовою; пер. с англ.

№	Назва:	Turbo Pascal, 5-е издание. – М.: Издательский дом «Вильямс», 2003. – 896 с.
4.	Автор(и):	Коффман Эллиот, Б.
	Зміст:	Введение Глава 1. Компьютеры и программирование: общий обзор Глава 2. Pascal и решение задач Глава 3. Функции, процедуры и графика Глава 4. Структуры выбора: операторы if и case Глава 5. Операторы while, for и repeat Глава 6. Структурное программирование Глава 7. Простые типы данных Глава 8. Текстовые файлы Глава 9. Разработка программного обеспечения и модули Глава 10. Массивы Глава 11. Многомерные массивы Глава 12. Записи Глава 13. Абстракция данных и объектно-ориентированное программирование Глава 14. Рекурсия Глава 15. Множества и строки Глава 16. Внешние информационные структуры: файлы Глава 17. Указатели и связанные списки Глава 18. Динамические информационные структуры Приложение А. Интегрированная среда Turbo Pascal Приложение Б. Зарезервированные слова, стандартные идентификаторы, операторы, модули, функции, процедури и директивы компилятора Приложение В. Синтаксические диаграммы Turbo Pascal Приложение Г. Набор символов ASCII Приложение Д. Краткий справочник по конструкциям Turbo Pascal Ответы для нечетных упражнений из разделов «проверь себя»

	1	2	3	4	5	6	7	8	9
	Pascal	+	-	+	+	-	-	+	рос. мовою; пер. с англ.

№	Назва:	Основы программирования. – М.: ЮНИМЕДИАСТАЙЛ, 2002. – 424 с.
5.	Автор(и):	Окулов С.М.
	Зміст:	Часть первая. Основные управляющие конструкции Занятие №1. Первая программа Занятие №2. Целый тип данных Занятие №3. Команды редактора для работы с блоками, работа с окнами Занятие №4. Логический тип данных, операции сдвига Занятие №5. Составной оператор и оператор *if* Занятие №6. Оператор *for* Занятие №7. Оператор *while* Занятие №8. Оператор *repeat-until* Занятие №9. Вложенные циклы Часть вторая. Процедуры и функции - элементы структуризации программ Занятие №10. Одмерные массивы Занятие №11. Процедуры Занятие №12. Функции Занятие №13. Рекурсия Занятие №14. Символьный и строковый типы данных Занятие №15. Вещественый тип данных Занятие №16. Текстовые файлы Часть третья. Массив - фундаментальная структура даннях Занятие №17. Методы работы с элементами одномерного массива Занятие №18. Множественный тип данных Занятие №19. Методы сортировки Занятие №20. Методы быстрой сортировки

		Занятие №21. Поиск данных Занятие №22. Двумерные массивы. Работа с элементами Занятие №23. Двумерные массивы. Вставка и удаление Занятие №24. Нескольео задач на технику работы с двумерными массивами Занятие №25. Комбинаторный тип данных (записи) Часть четвертая. Динамические структуры Занятие №26. Динамические структуры данных Занятие №27. Стек Занятие №28. Очередь Занятие №29. Поиск в графе Занятие №30. Двоичные деревья Занятие №31. Сбалансированные деревья

1	2	3	4	5	6	7	8	9
Pascal	+	-	-	+	-	+	-	рос. мовою.

№	Назва:	Язык программирования С. Лекции и упражнения – М.: Изд. дом «Вильямс», 2006. – 960 с.
6.	Автор(и):	Прата С.
	Зміст:	Глава 1. Предварительные сведения Глава 2. Введение в язык С Глава 3. Представление данных в языке С Глава 4. Символьные строки и форматированный ввод-вывод Глава 5. Операции, выражения и операторы Глава 6. Управляющие операторы: циклы Глава 7. Управляющие операторы: ветвление и безусловные переходы Глава 8. Символьный ввод-вывод и верификация ввода Глава 9. Функции Глава 10. Массивы и указатели Глава 11. Символьные строки и строковые функции Глава12. Классы памяти, компоновка и управление памятью Глава13. Файловый ввод-вывод Глава 14. Структуры и другие формы данных Глава 15. Операции с разрядами Глава 16. Препроцессор и библиотека языка С Глава 17. Расширенное представление данных Приложение А. Ответы на вопросы для самоконтроля Приложение Б. Справочный раздел Приложение В. Набор символов ASCII

1	2	3	4	5	6	7	8	9
С	+	+	+	+	-	+	+	рос. мовою; пер. с англ.

№	Назва:	Турбо Паскаль 7.0. Учебник. – К.: Издательская группа BHV, 2000. – 400 с
7	Автор(и):	Ставровский А.Б.
	Зміст:	Часть первая. Основы построения алгоритмов Глава 1. Обзор основных понятий Глава 2. начинаем изучать Паскаль Глава 3. Написано одно, а а делается ... разное Глава 4. Цикл "пока" и его использование Глава 5. Другие виды циклов Глава 6. Символьный тип и другие перечислимые типы Глава 7. Возвращаемся к определениям Глава 8. Подробнее о подпрограммах Глава9. Рекурсивное определение и подпрограммы Глава10. Язык и метаязык Часть вторая. Организация данных Глава 11. Представление чисел и других значений в компьютере Глава 12. Структуры Глава 13. Массивы Глава 14. Основы работы с файлами Глава 15. Обработка текстов

		Глава 16. Буферизация ввода-вывода Глава 17. Использование свободной памяти Часть третья. Несколько классических задач и алгоритмов Глава 18. Поиск, сортировка и понятие сложности Глава 19. Знакомство с сортировкой файлов Глава 20. Перебор вариантов Глава 21. вычисление выражений Глава 22. Элементы синтаксического анализа Глава 23. Поиск образца в строке							
	1	2	3	4	5	6	7	8	9
	Pascal	-	-	-	+	-	-	-	рос. мовою, є видання украинскою мовою.

Таблиця Е.2

Аналіз навчальних посібників з основ програмування

№	Назва:	Структуры данных и алгоритмы. : Пер. с англ. : М.: Издательский дом „Вильямс”, 2001. – 384 с.							
1.	Автор(и):	Ахо А.В., Хопкрофт Д.Э., Д.Д. Ульман							
	Зміст:	Глава 1. Построение и аналіз алгоритмов Глава 2. Основне абстрактне типы даннях Глава 3. Деревья Глава 4. Основные операторы множеств Глава 5. Специальные методы представления множеств Глава 6. Ориентированные графы Глава 7. Неориентированные графы Глава 8. Сортировка Глава 9. Методы анализа алгоритмов Глава10. Метоты разработки алгоритмов Глава11. Структуры данных и алгоритмы для внешней памяти Глава 12. Управление памятью							
	Базова мова програмування	Поради з програмування	Питання для самоконтролю	Вправи	Завдання на програмування	Розподіл завдань за рівнями складності	Кількість завдань із теми (достатньо для засвоєння навч. матеріалу/ недостатньо)	Відповіді та розв’язки	Примітка
	1	2	3	4	5	6	7	8	9
	Pascal	-	-	+	+	-	+	-	рос. мовою; пер. с англ.
№	Назва:	Алгоритмы + структуры данных = программы: Пер. с англ. – М.: Мир, 1985. – 406 с.							
2.	Автор(и):	Вирт Н.							
	Зміст:	1. Фундаментальные структуры даннях 1.1. Введение 1.2. Концепция типа для даннях 1.3. Простые типы даннях 1.4. Стандартне простые типы 1.5. Ограниченные типы 1.6. Массивы							

		1.7. Записи 1.8. Записи с вариантами 1.9. Множество 1.10. Представление массивов, записей и множеств 1.11. Последовательный файл Упражнения Литература 2. Сортировка 2.1. Введение 2.2. Сортировка массивов 2.3. Сортировка последовательных файлов Упражнения Литература 3. Рекурсивные алгоритмы 3.1. Введение 3.2. Когда не нужно использовать рекурсию 3.3. Два примера рекурсивных программ 3.4. Алгоритмы с возвратом 3.5. Задача о восьми ферзях 3.6. Задача об устойчивых браках 3.7. Задача оптимального выбора Упражнения Литература 4. Динамические информационные структуры 4.1. Рекурсивные типы данных 4.2. Ссылки или указатели 4.3. Линейные списки 4.4. Древовидные структуры 4.5. Сильно ветвящиеся деревья 4.6. Преобразования ключа (расстановка) Упражнения Литература 5. Структура языков и трансляторы 5.1. Определение иструктура языка 5.2. Анализ предложений 5.3. Построение синтаксического графа 5.4. Построение программы граматического разбора для заданного синтаксиса 5.5. Построение таблично-управляемой программы грамматического разбора 5.6. Преобразование БНФ в структуру данных, управляющую грамматическим разбором 5.7. Язык программирования ПЛ/0 5.8. Программа грамматического разбора для ПЛ/0 5.9. Восстановление при синтаксических ошибках 5.10. Процессор ПЛ/0 5.11. Формирование команд Упражнения Литература Приложение А Множество символов ASCII Приложение Б Синтаксические диаграммы Паскаля

1	2	3	4	5	6	7	8	9
Pascal	+	-	+	+	-	-	-	рос. мовою; пер. с англ.

№	Назва:	Основи програмування. Навчальний посібник. – К.: Наукова думка, 2004. – 355 с.
3.	Автор(и):	А.М. Гуржій, М.С. Львов, О.В. Співаковський
	Зміст:	Частина I Основи алгоритмізації та структурного програмування Вступ 1. Алгоритми

		2. Комп'ютери і програми 3. Мова програмування Pascal 4. Прості типи даних. Лінійні алгоритми 5. Процедурне програмування 6. Програмування розгалужень 7. Оператори повторення з параметром. Масиви 8. Ітераційні цикли 9. Рекурсія 10. Швидкі алгоритми сортування та пошуку 11. Складні типи даних: записи і файли 12. Множини 13. Динамічні структури даних 14. Методологія структурного програмування: підсумки Частина II 1. Вступ 2. Концепція ООП 3. Абстрактні типи даних і модулі 4. Поняття об'єкта: атрибути, методи та властивості 5. Динамічні об'єкти 6. Спадкування 7. Відношення між об'єктами 8. Відношення між класами 9. Методологія ООП 10. Лабораторний практикум курсу ООП

1	2	3	4	5	6	7	8	9
Pascal	+	-	+	+	-	+	-	

№	Назва:	Программирование на языке Turbo Pascal 6.0, 7.0. – М.: Веста, радио и связь, 1993. – 384 .
4.	Автор(и):	Зуев Е.А.
	Зміст:	Часть I. Ядро языка Turbo Pascal Глава 0. Способ синтаксического описания Глава 1. Алфавит, лексика, структура программы Глава 2. Система типов языка. Скалярные типы Глава 3. Описание переменных, констант и типов Глава 4. Преобразование типов. Совместимость типов Глава 5. Обзор составных типов. Регулярные типы. Массивы Глава 6. Комбинированные типы (записи) Глава 7. Строковые типы Глава 8. Множественные типы Глава 9. Операторы Глава 10. Процедуры и функции Глава 11. Ссылочные типы. Динамические переменные Глава 12. Выражения Глава 13. Файловые типы и ввод-вывод Часть II. Расширения языка Turbo Pascal Глава 14. Модули Глава 15. Средства ООП Глава 16. Системно-зависимые расширения Глава 17. Программирование на низком уровне Часть III. От версии 6.0 к версии 7.0 Глава 18. Основные расширения языка Глава 19. Новые средства работы со строками Приложение. Работа с Turbo Pascal-программами

1	2	3	4	5	6	7	8	9
Pascal	-	-	-	-	-	-	-	рос. мовою

№	Назва:	Основи інформатики. Структурне програмування на Паскалі: Практикум. Навч. посіб.- К.: Центр учбової літератури, 2007. – 248 с.
5.	Автор(и):	Каліон В.А., Черняк О.І., Харітонов О.М.
	Зміст:	Розділ 1. Базові типи даних Розділ 2. Складні оператори

<table>
<tr><td></td><td></td><td colspan="9">Розділ 3. Символи та рядки
Розділ 4. Перерахований та обмежений тип даних
Розділ 5. Елементи організації програми. Процедури, функції та модулі
Розділ 6. Структуровані типи даних. Масиви
Розділ 7. Комбіновані типи даних
Розділ 8. Множини
Розділ 9. Файловий тип даних
Розділ 10. Диманічні структури даних</td></tr>
<tr><td></td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr>
<tr><td></td><td>Pascal</td><td>+</td><td>+</td><td>+</td><td>+</td><td>-</td><td>+</td><td>+</td><td></td></tr>
<tr><td>№</td><td>Назва:</td><td colspan="9">Основи алгоритмізації та програмування: 750 задач с рекомендаціями та прикладами: Посібник. – К.: Форум, 2002. – 287 с.</td></tr>
<tr><td>6.</td><td>Автор(и):</td><td colspan="9">Караванова Т.М.</td></tr>
<tr><td></td><td>Зміст:</td><td colspan="9">Вступ
Розділ I. Основні поняття алгоритмізації
Вступ до алгоритмізації
Етапи розв’язування задач на комп’ютері
Розділ II. Базові алгоритмічні структури
Мова програмування Паскаль. Початкові поняття
Лінійні алгоритми
Розгалужені алгоритми
Масиви
Допоміжні алгоритми
Розділ III. Розширення мови програмування Паскаль
Додаткові можливості Паскаля
Покажчики. Посилальні типи. Динамічні змінні
Додаткові можливості введення та виведення інформації
Графічні можливості Паскаля
Модулі
Цикава суміш
Післямова
„Гарячі клавіші” середовища Turbo-Pascal 7.0
Помилки компіляції
Таблиця ASCII кодів
Таблиця скен-кодів</td></tr>
<tr><td></td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr>
<tr><td></td><td>Pascal</td><td>+</td><td>+</td><td>+</td><td>+</td><td>+</td><td>+</td><td>-</td><td></td></tr>
<tr><td>№</td><td>Назва:</td><td colspan="9">Язык программирования Си. 3 изд., испр. – СПБ.: «Невский диалект», 2001. – 352 с.</td></tr>
<tr><td>7.</td><td>Автор(и):</td><td colspan="9">Керниган Б., Ритчи Д.</td></tr>
<tr><td></td><td>Зміст:</td><td colspan="9">Введение
Глава 1. Обзор языка
Глава 2. Типы, операторы и выражения
Глава 3. Управление
Глава 4. Функции и структура программы
Глава 5. Указатели и массивы
Глава 6. Структуры
Глава 7. Ввод и вывод
Глава 8. Интерфейс с системоф UNIX
Приложение A. Справочное руководство
Приложение B. Стандартные библиотеки
Приложение C. Перечень изменений</td></tr>
<tr><td></td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr>
<tr><td></td><td>C</td><td>+</td><td>-</td><td>+</td><td>+</td><td>-</td><td>-</td><td>-</td><td>рос. мовою; пер. с англ.</td></tr>
<tr><td>№</td><td>Назва:</td><td colspan="9">Практика программирования. – СПб.: „Невский диалект”, 2001. – 381 с.</td></tr>
<tr><td>8.</td><td>Автор(и):</td><td colspan="9">Керниган Б., Пайк Р.</td></tr>
<tr><td></td><td>Зміст:</td><td colspan="9">Введение
1. Стиль
2. Алгоритмы и структуры даннях
3. Проектирование и реализация</td></tr>
</table>

<table>
<tr><td></td><td></td><td colspan="9">4. Интерфейсы
5. Отладака
6. Тестирование
7. Производительность
8. Переносимость
9. Нотація
Приложение: свод правил</td></tr>
<tr><td></td><td></td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr>
<tr><td></td><td></td><td>C</td><td>+</td><td>-</td><td>+</td><td>+</td><td>-</td><td></td><td>-</td><td>рос. мовою; пер. с англ.</td></tr>
<tr><td>№</td><td>Назва:</td><td colspan="9">Программирование для математиков: Учебное пособие для вузов. – М.: Наука. Гл. ред. физ.-мат. лит., 1988. – 384 с.</td></tr>
<tr><td>9.</td><td>Автор(и):</td><td colspan="9">Кушниренко А.Г., Лебедев Г.В.</td></tr>
<tr><td></td><td>Зміст:</td><td colspan="9">Глава 1. Начала программирования
1. Основные понятия программирования, примеры исполнителей и простейших программ
2. Процесс выполнения программы. Управляющие конструкции и утверждения
3. Основная задача программирования и технология „сверху вниз”
4. Процесс разработки программ. Рекурсия, итерация, проектирование цикла с помощью итерации
5. Процесс разработки программ. Один пример
6. Объекты, параметры, типы. Схемы вычисления инвариантной функции
7. Способы конструирования типов, объектов и исполнителей
8. Индуктивное вычисление функций на пространстве последовательностей
Глава 2. Несколько примеров программ
9. Проект „Выпуклая оболочка” последовательно поступающих точек плоскости
10. Компиляция и интерпретация. Реализация простейшего компилятора с языка арифметических формул
11. Проект „Построение изображения полиэдра”
Глава 3. Структуры данных и их реализации
12. Примеры реализации одних структур данных на базе других. Непрерывные реализации на базе вектора
13. Ссылочные реализации на базе вектора
14. Три способа реализации множества на базе вектора. Последовательный поиск, бинарный поиск, хеширование
15. Двумерное хеширование по равномерной сетке. оптимизация алгоритма построения изображения полиэдра
16. Виртуальная память
17. Простейшая файловая система
18. Иерархия структур данных при разработке программ
Глава 4. Логическое устройство и принципы работы ЭПВМ
19. Логическое устройство и принципы работы ЭВМ
20. Работа программиста на ЭВМ
Глава 5. Программирование на языке Фортран
21. Справочные сведения о языке Фортран. Простейшие примеры программ
22. Реализация исполнителей на Фортране. Примеры реализации структур даннях
Приложение. Краткое неформальное описание использованного в книге языка программирования</td></tr>
<tr><td></td><td></td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr>
<tr><td></td><td></td><td>викона вець</td><td>+</td><td>-</td><td>+</td><td>+</td><td>-</td><td>-</td><td>-</td><td>рос. мовою</td></tr>
<tr><td>№</td><td>Назва:</td><td colspan="9">Анализ алгоритмов, Вводный курс. – Москва: Техносфера, 2002. – 304 с.</td></tr>
<tr><td>10.</td><td>Автор(и):</td><td colspan="9">Макконел Дж.</td></tr>
<tr><td></td><td>Зміст:</td><td colspan="9">Предисловие
1. Основы анализа алгоритмов
1.1. Что такое анализ?
1.2. Что подсчитывать и что учитывать
1.3. Необходимые математические сведения</td></tr>
</table>

		1.4. Скорость роста 1.5. Алгоритмы вида „разделяй и влавствуй” 1.6. Рекурентне соотношения 1.7. Анализ программы 2. Алгоритмы поиска и выборки 2.1. Последовательный поиск 2.2. Двоичный поиск 2.3. Выборка 2.4. Упражнение по программированию 3. Алгоритмы сортировки 3.1. Сортировка вставками 3.2. Пузырьковая сортировка 3.3. Сортировка Шелла 3.4. Корневая сортировка 3.5. Пирамидальная сортировка 3.6. Сортировка слиянием 3.7. Быстрая сортировка 3.8. Внешняя многофазная сортировка слиянием 3.9. Дополнительные упражнения 3.10.Упражнения по программированию 4. Численные алгоритмы 4.1. Вычисление значений многочленов 4.2. Умножение матриц 4.3. Решение линейных уравнений 5. Алгоритмы сравнения с образцом 5.1. Сравнение строк 5.2. Приблизительное сравнение строк 6. Алгоритмы на графах 6.1. Основные понятия теории графов 6.2. Структуры данных для представления графов 6.3. Алгоритмы обхода в глубину и по уровням 6.4. Алгоритмы поиска минимального остовного дерева 6.5. Алгоритм поиска кратчайшего пути 6.6. Алгоритм определения компонент 6.7. Разбиения множеств 6.8. Упражнения по программированию 7. Параллельные алгоритмы 7.1. Введение в параллелизм 7.2. Модель PRAM 7.3. Простые параллельные операции 7.4. Параллельный поиск 7.5. Параллельная сортировка 7.6. Параллельные численные алгоритмы 7.7. Параллельные алгоритмы на графах 8. Недетерминированные алгоритмы 8.1. Что такое NP? 8.2. Типичные NP задачи 8.3. Какие задачи относятся к классу NP? 8.4. Проверка возможных решений 9. Другие алгоритмические инструменты 9.1. Жадные приближенные алгоритмы 9.2. Вероятностные алгоритмы 9.3. Динамическое программирование 9.4. Упражнения по программированию А. Таблица случайных чисел Б. Генерация псевдослучайных чисел Б.1. Случайная последовательность Б.2. Пример применения Б.3. Итоги В. Ответы к упражнениям
1	2	3	4	5	6	7	8	9

	Pascal	-	-	+	+	-	+	+	рос. мовою; пер. с англ.

№	Назва:	Программирование в среде Borland Pascal 7.0. – К.: Юниор, 1997. – 496 с.
11.	Автор(и):	Марченко А.И., Марченко Л.А.
	Зміст:	Часть 1. Обзор языка Глава 1. Упрощенная модель компилятора 1.1. Лексический анализатор 1.2. Синтаксический анализатор 1.3. Генератор кода 1.4. Таблицы 1.5. Структура профессиональной среды разработки программ BORLAND PASCAL WITH OBJECTS 7.0 Контрольные вопросы Глава 2. Набор символов, лексемы, разделители 2.1. Набор символов 2.2. Лексемы 2.3. Разделители Контрольные вопросы Глава 3. Структура программы 3.1. Заголовок программы 3.2. Раздел указания используемых модулей 3.3. Раздел описаний 3.4. Раздел операторов Контрольные вопросы Глава 4. Операции и выражения 4.1. Понятия выражения, операции, операнда 4.2. Приоритет операций и их классификация 4.3. Описание операций Контрольные вопросы Глава 5. Операторы 5.1. Простые операторы 5.2. Структурные операторы Контрольные вопросы Глава 6. Модули Глава 7. Динамически связываемые библиотеки 7.1. Создание динамически связываемых библиотек 7.2. Импорт процедур и функций из DLL 7.3. Область действия переменных в DLL 7.4. DLL, совместно используемые в защищенном режиме DOS и WINDOWS Контрольные вопросы Часть 2. Структуры данных и работа с ними средствами языка Глава 8. Классификация структур данных 8.1. Данные статической структуры 8.2. Данные динамической структуры Контрольные вопросы Глава 9. Обобщенные управляющие конструкции алгоритмов и их реализация средствами языка Конструкции условия Конструкции повторения Конструкции переходов Контрольные вопросы Глава 10. Работа с данными статической структуры Работа с данными простых типов Работа с составными данными однородной структуры Работа с составными данными неоднородной структуры Совместимость типов Контрольные вопросы Глава 11. Процедуры и функции 11.1. Структура процедур и функций 11.2. Область действия

		11.3. Классификация способов передачи параметров 11.4. Передача параметров в BP 11.5. Процедурные дерективы Контрольные вопросы Глава 12. Рекурсия 12.1. Понятие рекурсии и основные определения 12.2. Формы рекурсивных процедур 12.3. Задача о ханойских башнях 12.4. Быстрая сортировка Контрольные вопросы Глава 13. Файлы 13.1. Понятие физического илогического файла 13.2. Классификация файлов в BP 13.3. Назначение, открытие и закрытие файлов 13.4. Общие средства для работы с файлами 13.5. Типизированные файлы 13.6. Текстовые файлы 13.7. Нетипизированные файлы Контрольные вопросы Глава 14. Работа с динамическими структурами данных 14.1. Распределение памяти 14.2. Простейшие действия с указателями 14.3. Несвязанные динамические данные 14.4. Связанные динамические данные Контрольные вопросы Глава 15. Структурная методология разработки программ 15.1. Исторические заметки 15.2. Цели структурного программирования 15.3. Основные принципы структурной методологии 15.4. Модульное программирование 15.5. Стандарты структурного программирования Контрольные вопросы Глава 16. Объектно-ориентированная методология разработки программ 16.1. Основные понятия и определения 16.2. Объектно-ориентированные средства языка в BP 16.3. Объекты и модули 16.4. Директивы private и public 16.5. Наследование, правила наследования 16.6. Виртуальные и динамические методы 16.7. Динамические объекты 16.8. Совместимость объектных типов 16.9. Пример-упражнение по ООП Контрольные вопросы Часть 4. Стандартные приемы работы с устройствами IBM-PC Глава 17. Работа с клавиатурой, курсором и звуком 17.1. Клавиатура 17.2. Курсор 17.3. Звук Контрольные вопросы Глава 18. Работа в текстовом видеорежиме 18.1. Экран в текстовом режиме 18.2. Константы цвета 18.3. Окна в текстовом режиме 18.4 Прямой доступ к видеопамяти Контрольные вопросы Глава 19. Работа в графическом видеорежиме 19.1. Предопределенные константы 19.2. Предопределенные типы 19.3. Графические драйверы и режимы 19.4. Система координат 19.5. Текущий указатель 19.6. Инициализация графического режима

		19.7. Ошибки инициализации графического режима и их обработка 19.8. Простейшие графические фигуры 19.9. Взвод текста в графическом режиме 19.10 Эффект движения в графическом режиме Контрольные вопросы Глава 20. Borland Pascal и ассемблер Приложение 1. Стандартные процедуры и функции Приложение 2. Директивы компилятора Приложение 3. Сообщения об ошибках Приложение 4. Интегрированя среда Приложение 5. Таблица символов кода ASCII Приложение 6. Таблица кодов клавиатуры Список литературы по языку Pascal и программированию

1	2	3	4	5	6	7	8	9
Pascal	-	+	-	-	-	-	-	рос. мовою

№	Назва:	Техніка програмування мовою Сі.- К.: Либідь, 1993. – 224 с.
12.	Автор(и):	Проценко В.С., Чаленко П.П., Ставровський А.Б.
	Зміст:	Розділ 1. Мова програмування Сі 1. Основні поняття 2. Скалярні дані 3. Оператори 4. Функції та класи пам’яті 5. Масиви 6. Стандартні файли введення-виведення 7. Неоднорідні структури даних 8. Допоміжні засоби мови Розділ 2. Прості методи проектування програм 9. Лінійні списки 10. Стеки і черги 11. Використання лінійних списків 12. Стисле та індексне зберігання списків 13. Найпростіші методи сортування 14. Швидке і розподільне сортування та сортування злиттям Розділ 3. Складні структури даних 15. Матриці і багатовимірні масиви 16. Дерева та методи зберігання дерев 17. Бінарні дерева 18. Дерева і вирази 19. Графи та алгоритми на графах Розділ 4. Пошук 20. Пошук в лінійних списках 21. Задача вибору 22. Дерева бінарного пошуку 23. Збалансовані дерева 24. Вичерпний пошук Розділ 5. Робота з файлами 25. Обробка стандартних файлів 26. Обробка керуючих символів 27. Програма трансляції символів 28. Доступ до файлів. Обробка файлів 29. Сортування тексту 30. Сортування великих файлів Розділ 6. Елементи компіляції 31. Аналіз формальних мов 32. Реалізація мов програмування 33. Лексичний і синтаксичний аналізи 34. Інтерпретація 35. Формування проміжного коду 36. Генератор лексичних аналізаторів 37. Генератор синтаксичних аналізаторів

	1	2	3	4	5	6	7	8	9
	C	-	-	+	+	-	-	-	є помилки у текстах програм
№	Назва:	Путь программиста: Опыт создания личности программиста. – М.: «Издательский дом ИНФРА-М», 2000. – 864 с.							
13.	Автор(и):	Румянцев Д., Монастырский Л.							
	Зміст:	Часть I. В путь Глава 1. Как будто мы начинаем сначала Глава 2. Информация Глава 3. Алгоритм Глава 4. Программирование компьютера Глава 5. Системы счисления: от десяти к двум Глава 6. Машина и логика Часть II. Живой конструктор Глава 7. Переменные Глава 8. Процедуры и функции Глава 9. Управляющие структуры Глава 10. Статические структуры даннях Глава 11. Середа разработки программ Глава 12. Программа Глава 13. Файл Глава 14. Динамические структуры даннях Глава15. Машинная графика Часть III. Пользовательский интерфейс Глава 16. Интерфейс Глава 17. Клавиатура и меш Глава 18. Модуль Глава 19. Ориентирование среди объектов Глава 20. Объект как основа интерфейса Глава 21. Большой цикл Глава 22. Каркас программы Глава 23. Діалог между машиной и человеком Глава 24. Windows как платформа для разработки программ Глава 25. Сети и передача даннях Глава 26. Программист и Інтернет Часть IV. Система, продукт, товар Глава 27. От программ – к програмному продукту Глава 28. Тестирование. Отладка. Оптимизация Глава 29. Базы данных и управление ими Глава 30. Файл базы данных Глава 31. Проектирование баз даннях Глава 32. Автоматизация аудита Глава 33. Транспортная задача Глава 34. „Под колпаком” Глава 35. Рождение проекта Глава 36. Создание системы, внедрение и сопровождение Глава 37. Программы и рынок Часть V. Путь программиста Глава 38. Кто защитит программиста и его программы Глава 39. Інтерфейс-2 Глава 40. Ведущие и ведомые Заключение Приложение A. Типы файлов Приложение B. Обзор Borland Pascal Приложение C. Список листингов Приложение D. Список таблиц Приложение E. Список иллюстраций							
	1	2	3	4	5	6	7	8	9
	Pascal	+	-	-	-	-	-	-	рос. мовою

Таблиця Е.3

Аналіз збірників задач і вправ з основ програмування

№	Назва:	Задачи по программированию. – М.: Наука. Гл. ред. физ.-мат. лит., 1988. – 224 с.					
1.	Автор(и):	Абрамов С.А., Гнездилова Г.Г., Капустина Е.Н., Селюн М.И.					
	Зміст:	Глава I. Основне приемы программирования 1. Арифметика действительных чисел Вычисления по формулам 2. Разветвления 3. Простейшая целочисленная арифметика 4. Простейшие циклы 5. Простейшие графические построения 6. Пошаговый ввод данных и вывод результатов 7. Сочетание цикла и разветвления 8. Обработка последовательностей символов 9. Вычисления с хранением последовательности значений 10. Вложенные циклы 11. Вложенные циклы в матричних задачах 12. Использование процедур 13. Файлы 14. Вычисления с хранением последовательностей, число членов которых зависит от исходных данных Глава II. Задачи по темам 15. Целые числа 16. Системы счисления 17. Геометрия 18. Сортировка массивов и файлов 19. Многочлены 20. Преобразование и построение матриц 21. Матричная алгебра 22. Численне методы 23. Случайные числа 24. Вычисления с некоторой точностью 25. Физика 26. Биология 27. Тексты 28. Календарь 29. Криптография 30. Графика 31. Звукогенерация 32. Графика и движение. Мультипликация 33. Игры 34. Предметы и группы предметов с фиксированными свойствами 35. Перебор и его сокращение 36. Некоторые приемы программирования					
	Базова мова програ мува ння	Впра ви	Завдан ня на програ мува ння	Розпо діл завда нь за рівня ми склад ності	Кількі сть зав дань із теми (доста тньо для засвоє ння навч. мате ріалу/ недост атньо)	Відпо віді та розв’я зки	Примітка
	1	2	3	4	5	6	7
	-	-	+	+	+	-	базовий збірник для багатьох

<table>
<tr><td></td><td></td><td colspan="5"></td><td>видань;
рос. мовою</td></tr>
<tr><td>№</td><td>Назва:</td><td colspan="6">Сборник задач по программированию.- СПб.: Питер, 2003. – 475 с.</td></tr>
<tr><td rowspan="4">2.</td><td>Автор(и):</td><td colspan="6">Гуденко Д.А., Петроченко Д.В.</td></tr>
<tr><td>Зміст:</td><td colspan="6">1. Про мови програмування
2. Основні поняття
3. Умови
4. Цикли
5. Підпрограми
6. Масиви
7. Рядки, записи й множини
8. Файли
9. Модулі користувача</td></tr>
<tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr>
<tr><td>-</td><td>-</td><td>+</td><td>-</td><td>+</td><td>-</td><td>рос. мовою</td></tr>
<tr><td>№</td><td>Назва:</td><td colspan="6">Практикум программирования на Turbo Pascal. Задачи, алгоритмы и решения. – К.: Издательство «Диасофт», 2001. – 320 с.</td></tr>
<tr><td rowspan="4">3.</td><td>Автор(и):</td><td colspan="6">Зеленяк О.П.</td></tr>
<tr><td>Зміст:</td><td colspan="6">Глава 1. Арифметика вещественных чисел. Вычисления по формулам
Глава 2.Разветвления
Глава 3. Циклы
Глава 4. Сочетание цикла и разветвления
Глава 5. Одномерные массивы
Глава 6. Вложенные циклы
Глава 7. Подпрограммы. Рекурсия
Глава 8. Строки, множества, записи
Глава 9. Графические построения
Глава 10. Файлы
Глава 11. Модули
Глава 12. Задчи олимпиад по информатике</td></tr>
<tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr>
<tr><td>Pascal</td><td>-</td><td>+</td><td>+</td><td>+</td><td>-</td><td>рос. мовою</td></tr>
<tr><td>№</td><td>Назва:</td><td colspan="6">Программирование на С и С++. Практикум: Учеб. пособие для вузов/ А.В. Крячков, И.В. Сухинина, В.К. Томшин; Под ред. В.К. Томшина. – 2-е изд., исправ. – М.: Горячая линия. – Телеком, 2000. – 344.</td></tr>
<tr><td rowspan="4">4.</td><td>Автор(и):</td><td colspan="6">Крячков А.В., Сухинина И.В., Томшин В.К.</td></tr>
<tr><td>Зміст:</td><td colspan="6">Глава 1. Простейшие конструкции
Глава 2. Простые конструкции
Глава 3. Функции, определенные пользователем
Глава 4. Типове задачи I
Глава 5. Обработка символьных даннях
Глава 6. Нахождение корней уравнений
Глава 7. Вычисление определеных интегралов
Глава 8. Структуры
Глава 9. Сортировка и поиск данных
Глава 10. Организация работы с файлами
Глава 11. Типовые задачи II
Глава 12. Графика
Глава 13. Метод статистических испытаний
Глава 14. Решение систем линейных алгебраических уравнений
Глава 15. Динамические структуры
Глава 16. Интегрирование систем обыкновенных дифференциальных уравнений
Глава 17. Имитационное моделирование
Глава 18. Графы
Глава 19. Элементы С++ и объектно-ориентированное программирование</td></tr>
<tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr>
<tr><td>С</td><td>-</td><td>+</td><td>+</td><td>+</td><td>-</td><td>рос. мовою</td></tr>
<tr><td>№</td><td>Назва:</td><td colspan="6">Сборник упражнений по языку Паскаль: Учеб. пособие для вузов.- М.: Наука. Гл. ред. физ-мат. лит., 1989. – 160 с.</td></tr>
<tr><td>5.</td><td>Автор(и):</td><td colspan="6">Пильщиков В.Н.</td></tr>
</table>

	Зміст:	1. Числовые типы. Оператор присваивания 2. Логический тип 3. Простейшие программы 4. Операторы: условный, составной, пустой, перехода 5. Оператор цикла 6. Литерный тип 7. Перечислимые и ограниченные типы. Оператор варианта 8. Регулярные типы: векторы 9. Регулярные типы: матрицы 10. Регулярные типы: строки 11. Функции и процедуры 12. Рекурсия 13. Комбинированные типы. Оператор присоединения 14. Множественные типы 15. Файловые типы 16. Ссылочные типы. Списки 17. Очереди, стеки, двоичные деревья Ответы и решения

1	2	3	4	5	6	7
Pascal	+	+	+	+	+	рос. мовою

№	Назва:	Программирование на языке Паскаль: задачник. – СПб.: Питер, 2002. – 336 с.
6.	Автор(и):	под ред. Усковой О.Ф.
	Зміст:	Введение Глава 1. Числовые типы даннях Глава 2. Оператор присваивания, ввод и вывод информации Глава 3. Логический тип Глава 4. Условный оператор Глава 5. Оператор цикла Глава 6. Литерный тип Глава 7. Перечислимый тип, ограниченный тип, оператор варианта Глава 8. Регулярне типы даннях Глава 9. Процедуры и функции Глава 10. Рекурсивные подпрограммы Глава 11. Строковый тип Глава 12. Записи, оператор присоединения Глава 13. Множества Глава 14. Файлы Глава 15. Динамические структуры даннях Приложение 1. Ответы и решения Приложение 2. Задачи по информатике, предлагавшиеся на олимпиадах прошлых лет Приложение 3. Интегрированная инструментальная середа Turbo Pascal Приложение 4. Зарезервированные слова Turbo Pascal 7.0 Приложение 5. Комбинации клавиш быстрого управления в среде Turbo Pascal Приложение 6. Некоторые сообщения об ошибках периода компиляции

1	2	3	4	5	6	7
Pascal	+	+	+	+	+	рос. мовою

№	Назва:	Задачник по программированию. – СПб.: Питер, 2002. – 192 с.
7.	Автор(и):	Юркин А.Г.
	Зміст:	Введение О технологии программирования Часть I. Задания для начинающих 1. Линейные алгоритмы 2. Разветвляющиеся алгоритмы 3. Циклические и итерационные алгоритмы 4. Простейшие операции над массивами Часть II. Работа с масивами 5. Векторы и матрицы 6. Линейный поиск Часть III. Прикладные математические задачи

		7. Арифметика 8. Геометрия и теория множеств 9. Линейная алгебра и сжатие информации 10. Комбинаторика и теория вероятностей 11. Элементы численного анализа 12. Алгоритмы обработки символьной информации 13. Элементарная машинная графика 14. Элементы компьютерной мультипликации 15. Сортировка и слияние массивов 16. Поиск с возвратом. Задачи на графах 17. Разработка простейших АРМ и ИПС 18. Электронные таблицы					
	1	2	3	4	5	6	7
	-	-	+	+	+	-	п.17-18 ориенто-вані на СУБД і електрон-ні таблиці
№	Назва:	Задачи по языку Си. – М.: Финансы и статистика, 1985, (электронный вариант).					
8.	Автор(и):	Фьюер А.					
	Зміст:	Глава 1. Операции Глава 2. Основные типы Глава 3. Включение файлов Глава 4. Управление Глава 5. Стиль программирования Глава 6. Классы памяти Глава 7. Ссылки и массивы Глава 8. Записи Глава 9. Препроцессор					
	1	2	3	4	5	6	7
	C	+	-	-	-	+	рос. мовою; пер. с англ.
№	Назва:	Программирование: теоремы и задачи. М.: МЦНМО, 1995. – 264 с.					
9.	Автор(и):	Шень А.					
	Зміст:	1. Переменные, выражения, присваивания 2. Порождение комбинаторных объектов 3. Обход дерева, Перебор с возвратами 4. Сортировка 5. Конечные автоматы и обработка текстов 6. Типы данных 7. Рекурсия 8. Как обойтись без рекурсии 9. Разные алгоритмы на графах 10. Сопоставление с образцом 11. Представление множеств. Хеширование 12. Множества. Деревья. Сбалансированные деревья 13. Контекстно-свободные граматики 14. Синтаксический разбор слева направо					
	1	2	3	4	5	6	7
	псевдо код	+	+	+	+	+	рос. мовою

ДОДАТОК Ж

Приклад тестових завдань, що розроблені студентами

Вказівки повторення в мові Паскаль.

1. Умова, що записана між ключовими словами while і do має бути виразом:

а) логічного типа;
б) дійсного типа;
в) рядкового типа.

2. Оператор циклу repeat є оператором:
а) вибору;
б) умова;
в) оператором циклу з лічильником;
г) оператором циклу з умовою поста.

3. Оператор циклу for можна використовувати:

а) у всіх випадках;
б) лише тоді, коли для виконання циклу відомі початкове і кінцеве значення лічильника;
в) коли відомо хоча одне із значень лічильника.

4. Оператор циклу while- це:

а) оператор вибору;
б) оператор умови;
в) оператор циклу з передумовою;
г) оператор циклу з умовою поста.

5. Яку задачу розв'язує наступний фрагмент програми?

```
const n=20;
begin
. .
q:=0;
i:=1;
while i<=n do begin
        q:=q+a[i];
        i:=i+1; end;
..
end.
```

а) обчислює суму елементів масиву розмірністю в N елементів;
б) підраховує кількість елементів масиву;
в) обчислює розмірність масиву.

6. При роботі з оператором while, якщо при вході в цикл умова хибна, то:

   а) вхід в цикл не відбувається;
   б) вхід в цикл не виконується і управління передається операторові конкретно за оператором циклу;
   в) вхід в цикл не виконується і управління передається до кінця програми.

7. Оператори якого циклу виконуватимуться як мінімум один раз?
   а) for;
   б) repeat;
   в) while.

8. Аби оператор for виконувався хоча один раз потрібно щоб:
   а) початкове значення лічильника було не менше ніж кінцеве значення лічильника;
   б) початкове значення лічильника було менше ніж кінцеве значення лічильника;
   в) початкове значення лічильника було не більше чим кінцеве значення лічильника;
   г) початкове значення лічильника було більш ніж кінцеве значення лічильника.

9. При роботі з оператором while як тільки досягнутий кінець тіла циклу, управління перейде в:
   а) початок програми;
   б) кінець програми;
   в) на заголовок циклу, де знову обчислюється умова.

10. Робота циклу repeat продовжується, поки:

    а) умова достеменна;
    б) умова хибна;
    в) працює комп’ютер.

ДОДАТОК З

Конспект комп'ютерно-орієнтованої лекції
з теми «Двійкови дерева пошуку»

Змістовий модуль 5. Структури даних



Тема лекції:

Двійкові дерева пошуку

Підготував: ст. викл. кафедри
криптадної математики
Гришко Л.В.

1

Двійкови дерева пошуку

- **Властивості двійкового дерева пошуку**
- **Властивості впорядкованості ключів**
- **Алгоритми обходу дерева**

- **Допустимі операції**
  - пошук вузла із заданим значенням
  - пошук вузла з мінімальним значенням
  - пошук вузла з максимальним значенням
  - пошук попереднього за значенням вузла
  - пошук наступного за значенням вузла
  - додавання (вставка) вузла
  - видалення вузла із заданим значенням

2

Література:

- Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ. М.: МЦНМО, 1999. – С. 236-254.
- Ахо А.В., Хопркрофт Д.Э., Ульман Д.Д. Структуры данных и алгоритмы: Пер. с англ. – М.: Издательский дом "Вильямс", 2001. – С. 77-103.

3

Властивості двійкового дерева пошуку:

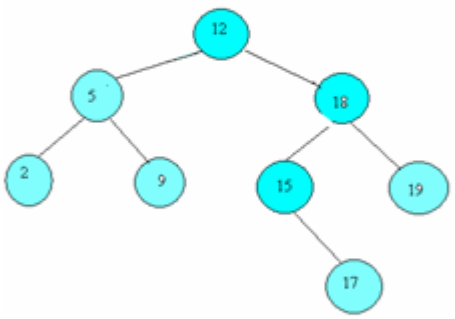
- Кожна вершина може мати (або не мати) лівого й правого нащадка
- Для кожного вузла зберігається:
  - значення ключа key
  - покажчик на лівого нащадка left
  - покажчик на правого нащадка right
  - покажчик на предка p
- Ключі зберігаються з дотриманням властивості впорядкованості

4

Властивості впорядкованості ключів:

- Нехай x - довільний вузол двійкового дерева пошуку.
- Якщо вузол y перебуває в левом поддереве вузла x, тоді
  key[y]<=key[x]

Якщо y перебуває в правом поддереві, тоді
  key[y]>=key[x]



5

Допустимі операції:

- пошук вузла із заданим значенням
- пошук вузла з мінімальним значенням
- пошук вузла з максимальним значенням
- пошук попереднього за значенням вузла
- пошук наступного за значенням вузла
- додавання (вставка) вузла
- видалення вузла із заданим значенням

- Час виконання операцій пропорційно висоті дерева

6

Обхід дерева:

- Inorder – виведення ключів у порядку неубивання
- Preorder – обхід дерева, при якому корінь (root) передує обом піддеревам
- Postorder – обхід дерева, при якому корінь следує за ключами

7

Алгоритм обходу дерева: Inorder

Вхідні дані: покажчик на корінь дерева root[T]

```
Inorder_Tree_Walk(x)
1  if x<>NIL
2    then Inorder_Tree_Walk (left[x])
3         надрукувати key[x]
4         Inorder_Tree_Walk (reght[x])
```

8

Пошук у двійковому дереві: Tree_Search

Вхідни дані:

k - ключ

x – покажчик на корінь піддерева

Вихідні дані:

покажчик на вершину з ключом k

або NIL, якщо такої вершини немає

```
Tree_Search(x,k)
1   if x=NIL или k=key[x]
2     then return x
3   if k < key[x]
4     then return Tree_Search(left[x],k)
5     else  return Tree_Search(right[x],k)
```

9

Пошук мінімума: Tree_Minimum
максимума: Tree_Maximum

Вхідні дані : покажчик на корінь дерева

Вихідні дані : значення мінімального (максимального) ключа

```
Tree_Minimum(x)
1   while left[x]<>NIL
2     do x ← left[x]
3   return x

Tree_Maximum(x)
1   while right[x]<>NIL
2     do x ← right [x]
3   return x
```

10

Пошук наступного вузла: Tree_Successor

Вхідні дані : покажчик x на вузол
Вихідні дані : покажчик y на наступний вузол згідно з величиною ключа
Проміжні дані : покажчик p на предка

```
Tree_Successor(x)
1  if right[x]<>NIL
2   then return Tree_Minimum(right[x])
3  y ← p[x]
4  while y<>NIL and x = right[y]
5     do x ← y
6        y ← p[y]
7  return y
```

11

Додавання вузла: Tree_Insert

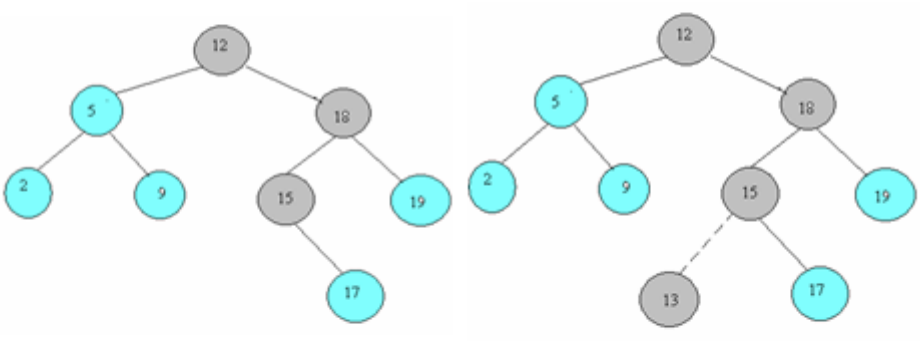
Вхідні дані : значення ключа z нового вузла;
T покажчик на корінь дерева
Проміжні дані: покажчики x і y

```
Tree_Insert(T, z)
1  y ← NIL
2  x ← root[T]
3  while x<> NIL
4     do y ← x
5        If key[z]<key[x]
6           then x ← left[x]
7           else x ← right[x]
8  p[x] ← y
9  If y = NIL
10    then root[T] ← z
11    else if key[z]<key[y]
12            then left[y] ← z
13            else right[y] ← z
```

12

Додавання вузла з ключом 13

(ясно-сірі вершини перебувають на шляху від кореня до позиції нового вузла)



13

Вилучення вузла дерева: Tree_Delete

Вхідні дані : значення ключа вузла z;
корінь root дерева T

Проміжні дані : покажчики x, y, p

```
Tree_Delete(T, z)
1  if left[z] = NIL или rsght[z] = NIL
2     then y ← z
3     else y ← Tree_Successor(z)
4  if left[y]<>NIL
5     then x ← left[y]
6     else x ← right[y]
```

Продовження на наступному слайді

Вилучення вузла дерева: Tree_Delete

продовження

```
7   if x<>NIL
8     then p[x] ← p[y]
9   if p[y]= NIL
10    then root[T] ← x
11    else if y=left[p[y]]
12           then left[p[y]] ← x
13           else right[p[y]] ← x
14  if y<>z
15    then key[z] ← key[y]
16         +копіювання доп. даних, які зв'язані з y
17    return y
```

15

ДОДАТОК І

Завдання-задачі з теми «Функції»

1-й рівень складності

1. Напишіть функцію з ім'ям `gcd`, яка повертає найбільший спільний дільник двох цілих чисел. Напишіть програму, яка використовує цю функцію.

2. Напишіть функцію, яка приймає три аргументи: символ і два цілих числа. Символ повинен бути роздрукований в рядку. Перше ціле значення визначає, скільки разів символ друкується в рядку, а друге ціле число задає, скільки рядків повинно бути роздруковане. Напишіть програму, яка використовує цю функцію.

3. Середнього гармонійного значення двох чисел може бути набуте, якщо узяти зворотні величини цих двох чисел, обчислити їх середнє значення і узяти його зворотну величину. Напишіть функцію із двома аргументами типу `double`, яка повертає середнє гармонійне значення цих двох чисел. Протестуйте цю функцію.

4. Ціле число називають простим, якщо воно ділиться лише на `1` і на само себе. Наприклад, `2, 3, 5, 7` є простими числами, а `4, 6, 8 і 9` – ні. Напишіть функцію, яка визначає, чи є числом простим. Використовуйте цю функцію в програмі, яка знаходить і друкує всі прості числа в діапазоні від `1` до `1000`.

5. Напишіть і протестуйте функцію, яка отримує ціле число і повертає число з оберненим порядком цифр. Наприклад, для числа 7631 функція повинна повернути число 1367.

6. Напишіть функцію `index(s,t)`, що повертає номер першого входження символу `t` в рядок `s`; якщо символ `t` в рядок не входить, функція повертає `-1`. Протестуйте цю функцію.

7. Натуральне число називається *досконалим*, якщо воно дорівнює сумі всіх своїх дільників крім себе самого. Число `6` – досконале, бо `6=1+2+3`. Число `8` – не є досконалим, бо 1+2+4≠8. Напишіть функцію `perfect`, яка визначає, чи є її параметр досконалим числом. Використайте цю функцію у програмі, яка знаходить усі досконалі числа у діапазоні від `1` до `n`. Надрукуйте усі дільники для кожного досконалого числа.

8. Напишіть функцію, яка перевіряє, чи закінчується рядок на `.abc`, і якщо ні - приписує `.abc` до кінця рядка. Якщо ж рядок вже має таке закінчення – залишити його без змін. Протестуйте цю функцію.

2-й рівень складності

1. Напишіть програму, яка прочитує символи із стандартного пристрою введення до тих пір, поки не зустрінеться символ кінця файлу. Програма повинна повідомити, коли символ, що вводиться, є буквою. Коли символ, що вводиться, є буква, вона повідомляє також її порядковий номер в алфавіті. Наприклад, букви *c* і `C` йдуть під номером `3`. Включіть також функцію, яка приймає символ в якості аргументу і повертає його номер в алфавіті, якщо цей символ є літерою, і 1 в протилежному випадку.

2. Напишіть функцію `to_binary()`, представлену у лістингу 9.8, до функції `to_base_n()`, яка приймає другий аргумент в діапазоні від `2` до `10`. Функція повинна виводить на друк число, яке є її першим аргументом, в системі зчислення, основа якої задається її другим аргументом. Наприклад, визів `to_base_n(129,8)` виведе на екран `201`, еквівалент числа `129` в восьмирічній системі. Протестуйте цю функцію.

3. Дано натуральне число `n`. Напишіть функцію, яка знаходить в порядку зростання перші `n` натуральних чисел, що не діляться ні на які прості числа, крім `2`, `3` і `5`. Протестуйте цю функцію.

3-й рівень складності

1. Напишіть функцію обчислення кореня однорідного рівняння `f(x)=0` методом поділу відрізку навпіл. Протестуйте цю функцію.

2. Напишіть програму-калькулятор, яка виконує операції складання, віднімання, множення і ділення дійсних чисел.

ДОДАТОК И

Загальні положення статистичної обробки результатів педагогічного експерименту за допомогою критерію φ*-кутового перетворення Фішера у поєднанні з критерієм λ Колмогорова-Смірнова

Розрахунок критерію λ при співставленні двох емпіричних розподілів

1. Занести в таблицю найменування розрядів та відповідні емпіричні частоти, отримані в розподілі 1 (перший стовпчик) і розподілі 2 (другий стовпчик).

2. Обрахувати емпіричні частоти по кожному розряду для розподілу 1 за формулою:

$$f_n^* = f_n / n1,$$

де f_n – емпірична частота в даному розряді;

$n1$ – кількість спостережень у 1-ї вибірці.

Занести емпіричні частоти розподілу 1 у третій стовпчик.

3. Обрахувати емпіричні частоти по кожному розряду для розподілу 2 за формулою:

$$f_n^* = f_n / n2,$$

де f_n – емпірична частота в даному розряді;

$n2$ – кількість спостережень у 2-ї вибірці.

Занести емпіричні частоти розподілу 2 у четвертий стовпчик.

4. Обрахувати накопичені емпіричні частоти для розподілу 1 за формулою:

$$\sum f^*_j = \sum f^*_{j-1} + f^*_j,$$

де $\sum f^*_{j-1}$ – частість, накопичена на попередніх розрядах;

j – порядковий номер розряду;

f^*_{j-1} – частість даного розряду.

Отримані результати записати в п’ятий стовпчик.

5. Обрахувати накопичені емпіричні частоти для розподілу 2 за тією ж формулою і записати результат у шостий стовпчик.

6. Обрахувати різниці між накопиченими частостями по кожному розряду. Записати в сьомий стовпчик абсолютні величини різниць, без їхнього знаку. Позначити їх як d.

7. Визначити за сьомим стовпчиком найбільшу абсолютну величину різниці d_{max}.

8. Обчислити значення критерію λ за формулою:

$$\lambda = d_{max} \cdot \sqrt{\frac{n_1 \cdot n_2}{n_1 + n_2}},$$

де n_1 – кількість спостережень у першій вибірці;

n_2 – кількість спостережень у другій вибірці.

9. За табл. XI [178, додаток 1] визначити, якому рівню статистичної значущості відповідає отримане значення λ. Якщо $\lambda_{эмп} \geq 1.3$, розходження між розподілами достовірні.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

4. Абалуев Р.Н. Интернет-технологии в образовании : Учебно-методическое пособие. Ч.3. / Абалуев Р.Н., Астафьева Н.Г., Баскакова Н.И. и др. – Тамбов : Изд-во ТГТУ, 2002. – 114 с.
5. Аврамчук Л.А. Формування активної пізнавальної діяльності студентів / Аврамчук Л.А. // Педагогіка і психологія. – 1997. – № 3. – С. 122–125.
6. Алексюк А.М. Педагогіка вищої освіти України / А.М. Алексюк. – К. : Либідь, 1998. – 560 с. – (Історія. Теорія: Підручник).
7. Алексюк А.М. Педагогіка вищої школи / А.М. Алексюк, — К., 1993. – 220 с. – (Курс лекцій: модульне навчання).
8. Амосов М.М. Моделирование мышления и психіки / М.М. Амосов – К. : Наукова думка, 1965. – 304 с.
9. Ананьев Б.Г. Избранные психологические труды: в 2 т. – / Б.Г. Ананьев – М. : Педагогика, 1980. – – Т. 2. – 1980.– 288 с.
10. Ананьев Б.Г. Избранные психологические труды: в 2 т. / Б.Г. Ананьев – М. : Педагогика, 1980.– – Т. 1. – 1980.– 232 с.
11. Ананьев Б.Г. Человек как предмет познания / Б.Г.Ананьев – Л. : Изд-во Ленинград. ун-та, 1968. – 339 с.
12. Андреева Д.А. О понятии адаптация. Исследование адаптации студентов к условиям учебы в вузе. / Д.А. Андреева – Л.: ЛГУ, 1973. – С. 62–69. – (Человек и общество: Ученые записки).
13. Андрющенко А.О. Рейтингова технологія оцінкі знань в навчально-виховному закладі / А.О. Андрющенко // Педагогіка і психологія. – 1996. – Київ : „Педагогічна думка” 3(12) – С. 86–93.
14. Артемов А. Модульно-рейтинговая система / А. Артемов, Н. Павлов, Т. Сидорова // Высшее образование в России. –1999. – № 4. – С. 67–75.

15. Архангельский С.И. Учебный процесс в высшей школе, его закономерные основы и методы / Архангельский С.И. – М.: Высш. школа, 1980. – 368 с.
16. Ахо А.В., Хопкрофт Д.Э., Ульман Д.Д. Построение и анализ вычислительных алгоритмов / Ахо А.В., Хопкрофт Д.Э., Ульман Д.Д. ; [пер. с англ.]. – М.: Издательство “Мир”, 1978. – 535 с.
17. Ахо А.В., Хопкрофт Д.Э., Ульман Д.Д. Структуры данных и алгоритмы / Ахо А.В., Хопкрофт Д.Э., Ульман Д.Д. ; [пер. с англ.]. – М. : Издательский дом “Вильямс”, 2001. – 384 с.
18. Бабанский Ю.К. Дидактические проблемы совершенствования учебных комплексов //– М.: Просвещение, 1980. – С. 17–33.
19. Бабанский Ю.К. Методы обучения в современной общеобразовательной школе / Бабанский Ю.К. – М.: Просвещение, 1985. – 208 с. – (Проблемы школьного учебника. Вып. 8 (О конструировании учебника)).
20. Бабанский Ю.К. Оптимизация процесса обучения / Бабанский Ю.К. – М. : Педагогика, 1977. – 256 с. – (Общедидактический аспект).
21. Бабанский Ю.К., Поташник М.М. Оптимизация педагогического процесса : (В вопросах и ответах) / Ю.К. Бабанский, М.М. Поташник – К. : Радянська школа, 1982. – 200 с.
22. Балл Г.А. Теория учебных задач. Психолого-педагогический аспект / Балл Г.А. – М. : Педагогика, 1990. – 184 с.
23. Белоус В.В. Психологические симптомы комплексы и инварианты темперамента : авт. дис. на получение научн. степени доктора психол. наук / В.В. Белоус. – Москва, 1982. – 41 с.
24. Бен-Ари М. Языки программирования. Сравнительный анализ / Бен-Ари М. ; [пер. с англ.]. – М. : Мир, 2000. – 366 с.
25. Бентли Дж. Жемчужины программирования / Бентли Дж. ; [пер. с англ.]. – СПб. : Питер, 2002. – 272 с.

26. Беспалько В.П. Системно-методическое обеспечение учебно-воспитательного процесса подготовки специалистов : учеб.-метод. пособие / В.П. Беспалько, Ю.Г. Татур. – М. : Высш. шк., 1989. – 144 с.
27. Богатырев А.О., Гришко Л.В. Некоторые аспекты преподавания курса “Основы программирования и алгоритмические языки” // Інформаційні технології в освіті, науці і техніці. ІІ Всеукраїнська конференція молодих науковців ІТОНТ-2000: Черкаси, 2000. – С. 29–30.
28. Бодалев А.А. Акмеология как научная дисциплина / Бодалев А.А. – М. : Российская Академия Управления, 1993. – 47 с.
29. Бондар В.І. Дидактика: ефективні технології навчання студентів / Бондар В.І. – К. : Вересень, 1996. – 129 с.
30. Бондарев В.М. Основы программирования / Бондарев В.М., Рублинецкий В.И., Качко Е.Г. – Харьков: Фолио, 1998. – 368 с.
31. Брукс Ф.П. Как проектируются и создаются программные комплексы / Брукс Ф.П. ; [пер. с англ.]. – М. : Наука, 1979. – 151 с.
32. Брукшир Дж. Введение в компьютерные науки / Брукшир Дж. ; [пер. с англ.]. – М.: Издательский дом “Вильямс”, 2001. – 688 с.
33. Бурдин А.О. О классификации задач / Бурдин А.О. – М., 1981. – С. 3–7. – (Совершенствование содержания и методов обучения естественно-математическим дисциплинам в средней школе).
34. Бухтияров А.М., Сборник задач по программированию на алгоритмических языках / А.М. Бухтияров, Г.Д. Фролов. – М. : Наука, 1974. – 240 с.
35. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений на С++ / Гради Буч ; [пер. с англ.]. – М. : Издательство БИНОМ, СПб. : Невский диалект, 1999. – 560 с.
36. Вейтман В. Программирование для Web / Вейтман В. – М. : Издательский дом “Вильямс”, 2000. – 368 с.
37. Вербицкий А.А. Активное обучение в высшей школе: контекстный подход / Вербицкий А.А. – М. : Высш. шк., 1991. – 207 с.

38. Вирт Н. Алгоритмы + структуры данных = программы / Вирт Н. ; [пер. с англ.]. – М. : Мир, 1985. – 406 с.
39. Вирт Н. Программирование на языке Модула-2 / Вирт Н. ; [пер. с англ.]. – М. : Мир, 1987. – 224 с.
40. Волкова Н.П. Педагогіка: [посібник для студентів вищих навчальних закладів] / Волкова Н.П. – К. : Видавничий центр “Академія ”, 2001. – 576 с.
41. Выготский Л.С. Избранные психологические произведения / Выготский Л.С. – М. : Учпедгиз, 1956. – 426 с.
42. Выготский Л.С. Педагогическая психология / Выготский Л.С. ; под ред. В.В. Давыдова. – М.: Педагогика, 1999. – 536 с.
43. Гальперин П.Я. Развитие исследований по формированию умственных действий / П.Я. Гальперин // Психологическая наука в СССР. – М.: Изд-во АПН РСФСР, 1959. – Т. 1. – С. 441–469.
44. Гальперин П.Я., Данилова В.Л. Воспитание систематического мышления в процессе решения малых творческих задач / П.Я. Гальперин, В.Л. Данилова // Вопросы психологии. – 1980. – № 1. – С. 31–38.
45. Гашков С.Б., Чубариков В.Н. Арифметика. Алгоритмы. Сложность вичислений : учеб. пособие [для вузов] / С.Б. Гашков, В.Н. Чубариков ; под ред. В.А. Садовничего. – [2-е изд., перераб.]. – Высш. шк., 2000. – 320 с.
46. Глазков В.В. Параллельное обучение алгоритмическим языкам / В.В.Глазков, Г.М. Киселев // Информатика и образование. – 2002. – № 6. – С. 33–39.
47. Глазков В.В., Киселев Г.М. Параллельное обучение алгоритмическим язикам // Информатика и образование. – 2002. – № 5. – С. 10–17.
48. Глушков В.М. Теория автоматов и формальные преобразования микропрограмм / **В.**М. Глушков // Кибернетика, 1965. – № 5. – С. 1–10.
49. Гончаренко С.У. Методика як наука / Гончаренко С.У. – Хмельницький : Вид-во ХГПК, 2000. – 30 с.

50. Грехэм Р. Практический курс языка Паскаль для микро-ЭВМ / Грехэм Р. ; [пер. с англ.]. – М.: Радио и связь, 1986. – 200 с.
51. Григас Г. Начала программирования / Григас Г. ; [пер. с англ.]. – М. : Просвещение, 1987. – 112 с.
52. Грин Д., Кнут Д. Математические методы анализа алгоритмов / Д. Грин, Д. Кнут; [пер. с англ.]. – М. : Мир, 1987. – 120 с.
53. Грис Д. Наука программирования / Грис Д. ; [пер. с англ.]. – М. : Мир, 1984. – 180 с.
54. Гриценко Л.И. Влияние типа познавательной деятельности учащихся старших классов на усвоение ими знаний : автореф. дис. на соискание науч. степени канд. пед. наук : 13.00.02 / Л.И. Гриценко. – Красноярск, 1972. – 28 с.
55. Гришко Л.В. Аналіз підручників, посібників і збірників задач з основ програмування // Інформаційні технології в освіті, науці і техніці. Матеріали VI Всеукраїнської конференції молодих науковців ІТОНТ-2008: Черкаси, 5-7 травня 2008. – С. 126.
56. Гришко Л.В. Порівняльний підхід до навчання основ програмування студентів технічних спеціальностей ВНЗ // Теорія та методика навчання математики, фізики, інформатики. Збірник наукових праць. – Випуск 3. – Т. 3. – Кривий Ріг. – 2003. – С. 101-106.
57. Гришко Л.В. Колективний проект як активний метод навчання програмуванню // Матеріали Всеукраїнської науково-практичної конференції. – Бердянськ, 2004. – С. 28-30.
58. Гришко Л.В. Колективний проект як активний метод навчання програмуванню // Збірник наукових праць Бердянського державного педагогічного університету (Педагогічні науки). – №1. – Бердянськ: БДПУ, 2004. – С. 52-57.
59. Гришко Л.В. Коллективный проект как практический прием обучения будущих программистов // Информационные технологии в учебном

процессе. Четвертый международный научно-методический семинар. – Одесса, 2003. – С. 189-191.

60. Гришко Л.В. Концептуальні підходи до навчання основ програмування у вищій школі // Науковий часопис НПУ імені М.П. Драгоманова. Серія №2. Комп'ютерно-орієнтовані системи навчання. Зб. наукових праць / Редрада. – К.: НПУ імені М.П. Драгоманова, 2004. №1(8). – С. 134-148.
61. Гришко Л.В. Навчання стилю програмування як складова формування професійної культури майбутнього інженера-програміста. Вісник Черкаського університету. Серія Педагогічні науки. – Випуск №143. – Черкаси, 2009. – С. 37–44.
62. Гришко Л.В. Початки програмування. Завдання до лабораторних робіт: Методичні матеріали для студентів 1-го та 2-го курсу спеціальності „Прикладна математика". – Черкаси, 2004. – 72 с.
63. Гришко Л.В. Про зміст курсу з основ програмування для майбутніх інженерів-програмістів // Інформаційні технології в освіті, науці і техніці. Матеріали V Всеукраїнської конференції молодих науковців ІТОНТ-2006: Черкаси, 3-5 травня 2006. – С. 124.
64. Гришко Л.В. Про методичну систему навчання основ програмування майбутніх інженерів-програмістів // Матеріали науково-практичної конференції. – Черкаси, 2006. – С. 50-53.
65. Гришко Л.В. Система завдань з основ програмування як засіб підвищення якості навчання майбутніх програмістів // Інформаційні технології в освіті, науці і техніці. IV Всеукраїнська конференція молодих науковців ІТОНТ-2004. Черкаси, 2004. – С. 163-164.
66. Гришко Л.В. Удосконалення системи завдань з основ програмування як засіб підвищення практичної значущості результатів навчання майбутніх інженерів-програмістів. Вісник Черкаського університету. Серія Педагогічні науки. – Випуск №101. – Черкаси, 2007. – С. 40–45.

67. Гришко Л.В., Нечаєнко Г.П., Триус Ю.В. Проблеми створення навчально-методичних комплексів на основі нових інформаційних технологій // : Матеріали Всеукраїнської конференції молодих науковців „Інформаційні технології в науці та освіті” (ІТОН-97). – Ч.1. – Черкаси, 1997. – С. 196–211.
68. Гришко Л.В., Чернявский Н.В. Пути индивидуализации процесса обучения основ программирования // Комп'ютерне моделювання та інформаційні технології в науці, економіці та освіті: Збірник наукових прац. – Черкаси: Брама ІСУЕП, 2003. – С. 27-29.
69. Гуржій А.М., Львов М.С., Співаковський О.В. Основи програмування. Навчальний посібник. – К.: Наукова думка, 2004. – 355 с.
70. Гусева А.Ф., Закс Е.В. Рейтинговая система – новый подход к организации контроля в обучении общей химии // Тез. докл. XVI Менделеевского съезда по общей и прикладной химии. – Т. 1. – СПб, 1998. – С. 370-371.
71. Дал У., Дейкстра Э., Хоор К. Структурное программирование: Пер. с англ. – М.: Издательство “Мир”, 1975. – 245 с.
72. Дейкстра Э. Заметки по структурному программированию // Дал У., Дейкстра Э., Хоор К. Структурное программирование. – М.: Мир, 1975. – 247 с.
73. Дейкстра Э. Дисциплина программирования: Пер. с англ. – М.: Издательство “Мир”, 1978. – 274 с.
74. Дидактика современной школы: Пособие для учителей/ Кобзарь Б.С., Кумарина Г.Ф., Кусый Ю.А. и др./ Под ред. Онищука В.А. – К.: Рад. шк., 1987. – 351 с.
75. Дидактика средней щколы / Под ред. М.А. Данилова и М.Н. Скаткина. – М., 1975. – 256 с.
76. Ершов А.П. Программирование – вторая грамотность http://ershov.iis.nsk.su/russian/second_literacy/article.html

77. Жалдак М.И., Квитко А.Н. Теория вероятностей с элементами информатики. Практикум. Учебное пособие / Под общ. ред. М.И. Ядренко. – К.: Вища школа, 1989. – 263 с.
78. Жалдак М.І., Рамський Ю.С. Інформатика: Навчальний посібник / За ред. Шкіля М.І. – К.: Вища школа, 1991. – 319 с.
79. Жалдак М.И. Система подготовки учителя к использованию информационной технологии в учебном процессе: Дисс. д-ра пед. наук: 13.00.02 – М.: НИИ СИМО АПН СССР, 1989. – 48с.
80. Заварыкин В.М. Техника вычислений и алгоритмизация: Вводный курс: Учебное пособие для студентов пед. ин-тов по физ.-мат. спец. / В.М. Заварыкин, В.Г. Житомирский, М.П. Лапчик. – М.: Просвещение, 1987. – 160 с.
81. Задачи по программированию / [Абрамов С.А., Гнездилова Г.Г., Капустина Е.Н., Селюн М.И.] – М. : Наука. Гл. ред. физ.-мат. лит., 1988. – 224 с.
82. Зарецкий М.И. Систематичность упражнений // Сов. педагогика. 1948. Т.3. – С. 8-40.
83. Зеленяк О.П. Практикум программирования на Turbo Pascal. Задачи, алгоритмы и решения. – К.: Издательство «Диасофт», 2001. – 320 с.
84. Зубов В.С. Программирование на языке TURBO PASCAL (версии 6.0 и 7.0). Издание 3-е, исправленное – М.: Информационно-издательский Дом “Филинъ”, 1997. – 320 с.
85. Иванов Б.Н. Дискретная математика. Алгоритмы и программы: Учеб. пособие. – М.: Лаборатория Базовых Знаний, 2001 – 288 с.
86. Ивановский С.А., Лисс А.Р., Романцев В.В., Экало А.В. Профессиональная подготовка программистов в рамках специальностей и направлений государственных образовательных стандартов // Материалы открытой всероссийской конференции "Преподавание Информационных Технологий в России". – 12 с.

87. Информатика для средней школы. – http://www.courier.com.ru/bibl/unesco_i.htm
88. Іваськів І.С. Активізація навчально-пізнавальної діяльності учнів на основі систем штучного інтелекту при навчанні інформатики в старшій школі: Дис. канд. пед. наук: 13.00.02 / НПУ ім. М.П. Драгоманова. – К., 2000. – 250 с.
89. Караванова Т.П. Основи алгоритмізації та програмування. 750 задач з рекомендаціями та прикладами. Посібник. – К.: ТОВ. «Форум». – 2001. – 286 с.
90. Кабанова-Меллер Е.Н. Психология формирования знаний и навыков у школьников. Проблема приемов умственной деятельности. М.: Изд-во АПН РСФСР, 1962. – 376 с.
91. Касимов Р.Я., Зинченко В.Я., Грантберг И.И. Рейтинговый контроль // Высшее образование в России. –1994. – №2. – С. 83-92.
92. Кассера Винфрид и Фолькер. Turbo Pascal 7.0: Пер. с нем./ Винфрид и Фолькер Кассера – К.: Издательство«ДиаСофт», 2003. – 448 с.
93. Керниган Б., Ритчи Д. Язык программирования С.\Пер. с англ., 3 изд., испр. – СПБ.: "Невский диалект", 2001. – 352 с.
94. Керниган Б.В., Пайк Р. Практика программирования.: Пер. с англ. – СПб.: “Невский диалект”, 2001. – 381 с.
95. Китаевская Т.Ю. Построение дифференцированного содержания обучения информатике в вузе с использованием компьютерно-ориентированных технологий. Автореф. дисс. на соискание ученой степени канд. пед. наук: 13.00.02. – М.: 2000. – 15 с.
96. Ключевые компетенции 2000. Программа. Уровни 1, 2, 3, 4, 5. KEY SKILLS 87 0005 (OCR. RECOGNISING ACHIEVMENT. – Oxford Cambridge and RSA Examinations).

97. Кнут Д.Э. Искусство программирования, том 1. Основные алгоритмы, 3-е изд.: Пер. с англ.: Уч. Пос. – М.: Издательский дом «Вильямс», 2000. – 720 с.
98. Кнут Д.Э. Искусство программирования, том 2. Получисленные алгоритмы, 3-е изд.: Пер. с англ.: Уч. Пос. – М.: Издательский дом “Вильямс”, 2000. – 832 с.
99. Кнут Д.Э. Искусство программирования, том 3. Сортировка и поиск, 3-е изд.: Пер. с англ.: Уч. Пос. – М.: Издательский дом “Вильямс”, 2000. – 832 с.
100. Кнут Д. Э. Алгоритмическое мышление и математическое мышление http://www.philosophy.ru/library/math/knut.html
101. Кобозев Н.И. Исследование в области термодинамики процессов информации и мышления – Изд-во Московского университета. – М., 1971. – 195 с.
102. Копаєв О. В. Алгоритмічна діяльність та рівень розвитку теоретичного мислення // Вісник Черкаського Національного університету. Серія «Педагогічні науки». Випуск 55. – Черкаси, 2004. – С. 77-81.
103. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн Р. Алгоритмы: построение и анализ. М.: Издательский дом “Вильямс”, 2005 – 1296 с.
104. Король В. и др. Психологическое обеспечение технологий образования // Alma mater. - №2. – С. 34-40.
105. Корсак К., Система освіти Франції // Міжнародна освіта. – 1997. - №4. – С. 6-13.
106. Костюк Г.С. Навчально-виховний процес і психічний розвиток особистості / Під ред. М.М. Проколієнко. – К.: Рад. шк., 1989. – 608 с.
107. Костюк Г.С. Про психологічні основи оптимізації шкільного навчання // Радян. школа. – 1972. – №3. – С. 4-16.
108. Коффман Эллиот, Б. Turbo Pascal, 5-е издание.: Пер. с англ. – М.: Издательский дом «Вильямс», 2003. – 896 с.

109. Крячков А.В., Сухинина И.В., Томшин В.К. Программирование на С и С++. Практикум: Учеб. пособие для вузов/ А.В. Крячков, И.В. Сухинина, В.К. Томшин; Под ред. В.К. Томшина. – 2-е изд., исправ. – М.: Горячая линия. – Телеком, 2000. – 344.
110. Кузнецов А.А., Захарова Т.Б. Принципы дифференциации содержания обучения информатике // Информатика и образование.– 1997.– № 7.– С. 9-11.
111. Кузьмина Н.В. Методы системного педагогического исследования: Учебное пособие.– Л.: ЛГУ, 1980.– 172 с.
112. Кузьмина Н.В. Профессионализм личности преподавателя и мастера производственного обучения. М.: 1990. – 117 с.
113. Кузьминський А.І., Омеляненко В.Л. Педагогіка: Підручник. – К.: Знання-Прес, 2003. – 418 с. – (Навчально-методичний комплекс з педагогіки).
114. Куписевич Ч. Основы общей дидактики: Пер. с польск. О. В. Долженко. – М.: Высш. шк., 1986. – 368 с.
115. Кыверялг А.А. Методы исследования в профессиональной педагогике. – Талин: Валгус, 1980. – 334 с.
116. Лавров С.С. Программирование. Математические основы, средства, теория. – СПб.: БХВ-Петербург, 2001. – 320 с.
117. Лебедев Г.В., Кушниренко А.Г. Программирование для математиков: Учебное пособие для вузов – М.: Наука. Гл. ред. физ.-мат. лит., 1988. – 384 с.
118. Леонтьев А.Н. Деятельность. Сознание. Личность. – М.: Политиздат, 1975. – 304 с.
119. Леонтьев А.Н. Проблемы развития психики. – 3-е изд. – М.: Изд-во МГУ, 1972. – 575 с.
120. Лернер И.Я. Дидактические основы методов обучения. – М.: Педагогика, 1981. – 186 с.

121. Лернер И.Я. Проблема познавательных задач в обучении основ гуманитарных наук и пути ее исследования (постановка проблемы) // Познавательные задачи в обучении гуманитарным наукам. М.: Педагогика, 1972. – С. 24-37.
122. Макаров А.В., Трофимова З.П. Модульная организация учебного процесса как основа разработки учебно-методического комплекса. // Социально-гуманитарные знания. – 2000. – №4. – С. 141-155.
123. Макконелл Дж. Анализ алгоритмов, Вводный курс. – Москва: Техносфера, 2002. – 304 с.
124. Макконнелл С. Совершенный код. Мастер-класс: Пер. с анг. – М.: Издательско-торговый дом "Русская редакция"; СПб.: Питер, 2005. – 896 с.
125. Марченко А.И., Марченко Л.А. Программирование в среде Borland Pascal 7.0 – К.: ЮНИОР, 1997. – 496 с.
126. Махмутов М.И. Современный урок. – 2-е изд., исп. и доп. – М.:Педагогика, 1985. – 184 с.
127. Машбиц Е.И. Психолого-педагогические проблемы компьютеризации обучения. – М.: Педагогика, 1988. – 192с.
128. Мейер Б., Бодуэн К. Методы программирования: В 2-х томах. Т.1. Пер. с франц. Ю.А. Первина. Под ред. и с предисловием А.П. Ершова. – М.: Мир, 1982. – 356 с.
129. Мейер Б., Бодуэн К. Методы программирования: В 2-х томах. Т.2. Пер. с франц. Ю.А. Первина. Под ред. А.П. Ершова. – М.: Мир, 1982. – 368 с.
130. Методічні рекомендації щодо структури, змісту та обсягів підручників і навчальних посібників для вищих навчальних закладів. – К.: Знания, 2005. – 8 с.
131. Метологические и теоретические проблемы психологии / Под ред. Е.В. Шороховой. – М.: Наука, 1969. – 376 с.
132. Милов А.В. Основы программирования в задачах и примерах, Харьков: ФОЛИО, 2002.- 397 с.

133. Могилев А.В. и др. Информатика: Учеб. пособие для пед. вузов / А.В. Могилев, Н.И. Пак, Е.К. Хённер; Под ред. У.К. Хённера. – М.:, 1999. – 816 с.
134. Мозолин В. В. Компетентностный подход в обучении информатике: возможности и проблемы/В. В. Мозолин//Информатика и образование. – 2005. – №2. – С. 115 – 116.
135. Монахов В.М., Орлов В.А., Фирсов В.В. Дифференциация обучения в средней школе // Советская педагогика. – 1990. – №8. – С. 42-47.
136. Морзе Н.В. Методика навчання інформатики: Навч. посіб.: У 4 ч. / За ред. акад. М.І. Жалдака. – К.: Навчальна Книга, 2003. – Ч. 1: Загальна методика навчання інформатики. – 254 с.
137. Морзе Н.В. Методика навчання інформатики: Навч. посіб.: У 4 ч. / За ред. акад. М.І. Жалдака. – К.: Навчальна Книга, 2003. – Ч. ІІ: Методика навчання інформаційних технологій. – 287 с.
138. Морзе Н.В. Методика навчання інформатики: Навч. посіб.: У 4 ч. / За ред. акад. М.І. Жалдака. – К.: Навчальна Книга, 2004. – Ч. IV: Методика навчання основ алгоритмізації та програмування. – 368 с.
139. Наделяев В., Мартынова Т., Герстенбергер В. и др. Рейтинговая система оценки знаний при изучении общетехнических дисциплин // Высшее образование в России. – 1997. – №2. – С. 103-107.
140. Наследов А.Д. Математические методы психологического исследования. Анализ и интерпретация данных. Учебное пособие. – СПб.: Речь, 2004. – 392 с.
141. Новые педагогические и информационные технологии в системе образования Полат Е.С., Бухаркина М.Ю., Моисеева М.В. и др. Под ред. Е.С.Полат М.: Педагогика. - 2007. – 272 с.
142. Обучающие программы языкам программирования Pascal и C (компакт-диск).

143. Одинцов И.О. Профессиональное программирование. Системный подход. – СПб.: БХВ·Петербург, 2002. – 512 с.
144. Окулов С.М. Основы программирования. – М.: ЮНИМЕДИАСТАЙЛ, 2002. – 424 с.
145. Організація навчального процесу у вищих закладах освіти: Навч. посібник для слухачів закладів підвищення кваліфікації системи вищої освіти. – К.: ВВП «КОМПАС», 1997. – 64 с.
146. Освітньо-професійна програма підготовки бакалавра напряму підготовки 0802 "Прикладна математика". – Київ, 2002.
147. Освітньо-професійна програма підготовки бакалавра напряму підготовки 0804 "Комп'ютерні науки". – Київ, 2001.
148. Освітньо-професійна програма підготовки бакалавра напряму підготовки 0915 "Комп'ютерна інженерія". – Київ, 2002.
149. Основи нових інформаційних технологій навчання: Посібник для вчителів / Авт. кол.; За ред. Ю.І. Машбиця / Інститут психології ім. Г.С. Костюка АПН України. – К.: ІЗМН, 1997. – 264 с.
150. Основні засади розвитку вищої освіти України в контексті Болонського процесу (документи і матеріали 2003-2004рр.), за редакцією В.Г. Кременя. Авторський колектив: М.Ф. Степко, Я.Я. Болюбаш, В.Д. Шинкарук, В.В. Грубінко, І.І. Бабин. – Тернопіль: Навчальна книга – Богдан, 2004. – 156 с.
151. Основы вузовской педагогіки / [авт. текста А.Г. Сороковой]. – Издательство Ленинградского университета. – 1972. – 311 с.
152. Основы педагогики и психологии высшей школы/ Под. ред. А.В. Петровского. – М.: Изд-во МГУ, 1986. – 303 с.
153. Пауэлл Т.А. Полное руководство по HTML. / Пер. с англ. – Мн.: ООО "Попурри", 2001. – 912 с.

154. Педагогика : Учебное пособие для студентов пед. вузов / [Бабанский Ю.К., Мластенин В.А., Сорокин Н.А. и др.] ; под ред. Ю.К. Бабанского. – [2-е изд., доп. и перераб.]. – М. : Просвещение, 1988. – 479 с.
155. Педагогика и психология высшей школы: Учеб. пособие для студентов и аспирантов вузов // С.И. Самыгин, М.В. Буланова-Топоркова, А.В. Духавнева и др. – Ростов на Дону: Феникс, 1998. – 544 с.
156. Пильщиков В.Н. Сборник упражнений по языку Паскаль: Учеб. пособие для вузов.- М.: Наука. Гл. ред. физ-мат. лит., 1989. – 160 с.
157. Пойа Д. Как решать задачу. - Львов, журнал «Квантор», 1991, - 215 с.
158. Пойа Д. Математическое открытие: Пер. с англ./Под ред. И.М. Яглома. – М.: Наука, 1970. – 452 с.
159. Попов Ю.В., Подлеснов В.Н., Андросюк Е.Р., Кучеров В.Г., Садовников В.И. Рейтинговая система как инструмент современных технологий обучения: Волгоградский технический університет. – 1999. – 45 с.
160. Прата С. Язык программирования С. Лекции и упражнения: Пер. с англ. – К.: Издательство “Диасофт”, 2000. – 432 с.
161. Прата С. Язык программирования С++. Лекции и упражнения. Учебник. Пер. с англ. – Издательство “ДиаСофт”, 2001. – 656 с.
162. Пратт Т., Зелковиц М. Языки программирования: разработка и реализация. – СПб.: Питер, 2002. – 688 с.
163. Преподавание информационных технологий в России. Открытая всероссийская конференция. – http://www.sparc.spb.su/oio/cs/. <12.04.07>
164. Принцип развития в психологии / Под ред. Л.И. Анцыферова. – М.: Наука, 1978. – С. 302-303.
165. Про організацію навчального процесу у вищих навчальних закладах у 2002/2003 навчальному році. // Вища школа. – 2001. - №4-5. – С. 119-125.

166. Про перелік напрямів, за якими здійснюється підготовка фахівців у вищих навчальних закладах за освітньо-кваліфікаційним рівнем бакалавра. Постанова кабінету Міністрів України від 13 грудня 2006 р., N1719, Київ.
167. Проблемы и перспективы реализации компетентностного подхода в образовании : (Проблемы компетентностного подхода) [Єлектронний ресурс] / А.Г. Бермус // Интернет-журнал «Эйдос» – Научные исследования. – Режим доступу до журн. : http://www.auditorium.ru
168. Проблемы методики и методологии в высшей школе: Сб. статей/ Под ред. Козлова В.В. – Петрозаводск, 1976. – 120 с.
169. Программирование на языке Паскаль: задачник / под ред. Усковой О.Ф. – СПб.: Питер, 2002. – 336 с.
170. Прокопенко І.Ф., Євдокимов В.І. Педагогічна технологія. – Харьків, 1995. – 374 с.
171. Проценко В.С. та ін. Техніка програмування мовою Сі: Навч. посібник/В.С. Проценко, П.П. Чаленко, А.Б. Ставровский. – К.: Либідь, 1993. – 224 с.
172. Психологические и психофизиологические особенности студентов / Под ред. Пейсахова Н. Казань: Изд-во Казан. ун-та, 1977. – 261 с.
173. Пустовіт Л.О., Скопенко О.І., Сюта Г.М., Цимбалюк Т.В. Словник іншомовних слів. – К., 2000. – 1018 с.
174. Пышкало А.М. Методическая система обучения геометрии в начальной школе: Автореф. дис. … док. пед. наук – М.:1975. – 60 с.
175. Решетова З.А. Психологические основы профессионального обучения. М., 1985
176. Рубинштейн С.Л. Основы общей психологии. – 2-е изд. – М.: Учпедгиз, 1947. – 704 с.
177. Румянцев Д., Монастырский Л. Путь программиста: Опыт созидания личности программиста. – М.: “Издательский дом ИНФРА-М”, 2000. – 864 с.

178. Сафонов А.Ф., Зинченко В.Я., Грантберг И.И.и др. Рейтинг в вузе: закономерное и случайное // Высшее образование в России. – 1994. – №3. – С. 66-77.
179. Сборник задач по программированию / Д.А. Гуденко, Д.В. Петроченко. – СПб.: Питер, 2003. – 475 с.
180. Себеста Р. Основные концепции языков программирования. 5-е издание.: Пер. с англ. – М.: Издательский дом “Вильямс”, 2001. – 672 с.
181. Сейдаметова З.С. Методическая система уровневой подготовки будущих инженеров-программистов по специальности «Информатика» : дис. ... доктора пед. наук : 13.00.02 / З.С. Сейдаметова; НПУ им. М.П. Драгоманова. – К., 2007. – 559 с.
182. Семеріков С.О. Актівізація пізнавальної діяльності студентів при вивченні чисельних методів у об’ектно-оріентованій технології програмування. Автореф. дисс. на соискание ученой степени канд. пед. наук: 13.00.02. – теория и методика обучения информатике. – Национальный педагогический университет имени М.П. Драгоманова, Киев, 2001. – 20 с.
183. Семеріков С.О. Фундаменталізація навчання інформативних дисциплін у вищий школі: Монографія / Науковий редактор академік АПН України, д.пед.н., проф. М.І. Жалдак. – К.: НПУ ім. М.П. Драгоманова, 2009. – 340 с.
184. Сидоренко Е.В. Методы математической обработки в психологии. – СПб.: ООО “Речь”, 2003. – 350 с.
185. Симухин Г., Стандартизация профессионального образования: история, опыт, проблемы // Высшее образование в России. – 2001. – №4. – С. 13-28.
186. Системы диалогового решения задач на ЭВМ /Перевозчикова О.Л., Ющенко Е.Л. – Киев: Наук. думка, 1986. – 264 с.
187. Скаткин М.Н. Совершенствование процесса обучения. М., «Педагогика», 1971. – 208 с.
188. Сластенин В.А. Интегративные тенденции в системе психолого-педагогической подготовки учителя // Приобщение к педагогической

профессии: Практика, концепции, новые структуры. – Воронеж, 1992. – С. 6-9.

189. Слепкань З.И. Психолого-педагогические основы обучения математике: Метод. пособие. – К.: Рад. школа, 1983. – 192 с.
190. Слєпкань З.І. Наукові засади педагогічного процесу у вищій школі. – К.: НТУ, 2000. – 210 с.
191. Смолкин А.М. Методы активного обучения: Научн.-метод. пособие. – М.: Высшая школа, 1991. – 176 с.
192. Смолюк І.О. Педагогічні технології: дослідження соціально-особистісного аспекту. – Луцьк: Ред.-вид. відд. "Вежа" Волин. держ. ун-ту ім. Лесі Українки, 1999. – 294 с.
193. Смульсон М.Л. Психологія розвитку інтелекту: Монографія. – К., 2001. – 276 с.
194. Соммервилл И. Инженерия программного обеспечения, 6-е издание.: Пер. с англ. – М.: Издательский дом «Вильямс», 2002. – 624 с.
195. Сорокина И. Инновационные методы обучения: проблемы внедрения. – Высшее образование в России. №1 2001. – С. 117-119.
196. Спірін О.М. Диференційований підхід у вивченні основ штучного інтелекту в курсі інформатики фізико-математичного факультету вищого педагогічного закладу: Дис. канд. пед. наук: 13.00.02 / НПУ ім. Драгоманова. – К., 2001. – 223 с.
197. Ставровский А.Б. Турбо Паскаль 7.0. Учебник. – К.: Издательская группа BHV, 2000. – 400 с.
198. Страуструп Б. Язык программирования С++, 3-е изд.: Пер. с англ. – СПб.; М.: “Невский диалект” – “Издательство БИНОМ”, 1999. – 991 с.
199. Талызина Н.Ф. Управление процессом усвоения знаний. – М.: Изд-во МГУ, 1975. – 343 с.

200. Талызина Н.Ф. Пути использования теории планомерного формирования умственных действий в практике образования // Вестник московского университета. Сер. 14. Психология. – 1992. – №4. – С. 18-26.
201. Тассел Д. Стиль, разработка, эффективность, отладка и испытание программ. – М.: Мир, 1985. – С. 286.
202. Толкачев Ф.В. Система упражнений по императивному программированию в фундаментальной подготовке будущих учителей информатики. Автореф. дисс. на соискание ученой степени канд. пед. наук: 13.00.02. – Санкт-Петербург: 2000. – 12 с.
203. Триус Ю.В. Технологія використання рейтингової системи оцінювання навчальної діяльності студентів // Вісник Черкаського університету. Педагогічна серія. – Черкаси, 2001. – №26. – С. 141-151.
204. Триус Ю.В. Комп'ютерно-орієнтовані методичні системи навчання математики: Монографія. – Черкаси: Брама-Україна, 2005. – 400 с.
205. Триус Ю.В., Богатирьов О.О., Гришко Л.В. Особливості створення методичної системи навчання основ програмування для підготовки майбутніх інженерів-програмістів // Вісник Черкаського університету, серія "Педагогічні науки". Випуск 35. – Черкаси, 2002. – С. 135-141.
206. Турбовский Я. С. Взаимоотношение педагогической науки и практики как методологическая проблема // Методологические проблемы развития педагогической науки. М.: Педагогика. – 1985. – С. 34-45.
207. Указ президента України „Про Національну доктрину розвитку освіти", м. Київ, 17 квітня 2002 року, № 347/2002.
208. Український педагогічний словник / [авт.-уклад. Гончаренко С.]. – Київ : Либідь, 1997. – 376 с.
209. Унт И.Э. Индивидуализация и дифференциация обучения. – М.: Педагогика, 1990. – 192 с.
210. Уоррен, Генри, С. Алгоритмические трюки для программистов.: Пер. с англ. – М.: Издательский дом «Вильямс», 2003. – 288 с.

211. Уфимцева М.А. Формы организации обучения в современной общеобразовательной школе. – М.: Просвещение, 1986. – 80 с.
212. Уэзерелл Ч. Этюды для программистов. – М.: Мир, 1982. – 288 с.
213. Фіцула М.М. Педагогіка: Навчальний посібник для студентів вищих педагогічних закладів освіти. – К.: Видавничий центр “Академія”, 2000. – 544 с. (Альма-матер).
214. Форми навчання в школі: Книга для вчителя / За ред. Мальованого Ю.І. – К.: Освіта, 1992. – 160 с.
215. Формирование учебной деятельности студентов / В.Я. Ляудис, Х. Варнеке, И.И. Ильясов и др. / Под ред. В.Я. Ляудис. – М.: Изд-во МГУ, 1989.– 239 с.
216. Формирование учебной деятельности студентов / Под ред. Ляудис В.Я. – М.: Изд-во Моск. ун-та, 1989. – 240 с.
217. Фридман Л.М. Педагогический опыт глазами психолога: Книга для учителя. – М.: Просвещение, 1987. – 224 с.
218. Фьюер А. Задачи по языку Си. – М.: Финансы и статистика, 1985, (электронный вариант).
219. Хуторской А.В. Ключевые компетенции и образовательные стандарты // Интернет-журнал "Эйдос". – 2002. – 23 апреля. http://www.eidos.ru/journal/2002/0423.htm
220. Хювёнен Э., Сеппянен Й. Мир Лиспа. В 2-х т. Т. 1: Введение в язык Лисп и функциональное программирование. Пер. с финск. – М.: Мир, 1990. – 447 с.
221. Хювёнен Э., Сеппянен Й. Мир Лиспа. В 2-х т. Т. 2: Методы и системы программирования. Пер. с финск. – М.: Мир, 1990. – 319 с.
222. Шахмаев Н.М. Дифференциация обучения в средней общеобразовательной школе // Дидактика средней школы / Под ред. М.Н. Скаткина. – 2-ое изд., перера. И доп. – М.: Просвещение, 1982. – Гл. 8. – С 269-296.
223. Шень А. Программирование: теоремы и задачи. М.: МЦНМО, 1995. – 264 с.

224. Шилдт Г. Самоучитель C++, 3-е издание: пер. с англ. – СПб.: BHV – Санкт-Перетбург, 1998. – 668 с.
225. Шнейдерман Б. Психология программирования: Человеческие факторы в вычислительных и информационных системах. Пер. с англ.– М.: Радио и связь, 1984. – 304 с.
226. Экспертно-обучающие системы / Петрушин В.А.; Отв. ред. А.М. Довгяло; АН УССР, Ин-т кибернетики. – Киев: Наук. думка, 1992. – С. 196.
227. Эсаулов А.Ф. Активизация учебно-познавательной деятельности студентов. – М.: Высшая школа. – 1988. – 223 с.
228. Юркин А.Г. Задачник по программированию. – СПб.: Питер, 2002. – 192 с.
229. Юцявичене П.А. Принципы модульного обучения // Сов. педагогика. – 1990. – №1. – С. 56-60.
230. Ягупов В.В. Педагогіка: Навч. посібник. – К.: Либідь, 2002. – 560 с.
231. Якиманская И.С. Развивающее обучение.- М.: Педагогика, 1979 . – 144с.
232. Blackbox Conponent Builder (Blackbox) http://oberon.nnm.ru/reliz_blackbox_component_builder_15_dlya_windows
233. Computing Curricula 2001. – http://www.computer.org/education/cc2001.
234. Greeno, J.G., The structure of memory and the process of problem solving, University of Michigan: Human Perfomance CenterTechnical Report 37, 1972. – p. 128.
235. http://www.osp.ru/pcworld/2001/04/161427 Летопись языков Паскаль // Интернет-журнал «Мир ПК».
236. Miller, G.A., Needed: A better theory of cognitive organization, IEEE Trasaktions on Systems, Man, and Cybernetics, SMC-4, 1, (January 1974), p. 95-97.
237. Mood A.M. Macro-Analysis of the American Educational System, Operations Res., 17, 770–784 (1969).
238. Software Engineering 2004: Curriculum GuideLines for Undergraduate Degree Programs in Software Engineering –

http:/www.computer.org/portal/cms_docs_ieeecs/education/cc2001/SE2004Volume.pdf

239. Sway, Mary. We can teach software bitter // Computing Research News V.4(4) : 2-12, September 1992.