ЗВО «УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА»
Факультет суспільних і прикладних наук
Кафедра інформаційних технологій

МЕТОДИЧНІ ВКАЗІВКИ

щодо виконання курсової роботи з дисципліни
«Об’єктно-орієнтоване програмування»

для студентів денної та заочної форми навчання
першого (бакалаврського) рівня вищої освіти
галузі знань 12 «Інформаційні технології»
спеціальності 121 «Інженерія програмного забезпечення»

2021
Івано-Франківськ

Методичні вказівки щодо виконання курсової роботи з дисципліни «Об’єктно-орієнтоване програмування» для студентів денної та заочної форми навчання першого (бакалаврського) рівня вищої освіти галузі знань 12 «Інформаційні технології» спеціальності 121 «Інженерія програмного забезпечення»

Івано-Франківськ: Університет Короля Данила, 2020. 36 с.

Розробник: к.т.н., Пашкевич О.П.

Методичні вказівки щодо виконання курсової роботи з дисципліни «Об’єктно-орієнтоване програмування» затверджені на засіданні кафедри інформаційних технологій та програмної інженерії

Протокол № 7 від27 лютого 2021 року.

Завідувач кафедри ***Пашкевич О.П.***

ЗМІСТ

ЗАГАЛЬНІ ПОЛОЖЕННЯ

Підготовка фахівців високого рівня потребує ґрунтовного вивчення дисциплін циклу професійної підготовки та виконання задач прикладного характеру. Самостійна робота студента передбачає виконання цілої низки дій з розробки програмного забезпечення від постановки задачі до програмної реалізації, випуску програмної документації та захисту роботи. В процесі виконання роботи студент свої знання поглиблює та набуває практичних навичок використання методів і засобів розробки програмного забезпечення, закріплює раніш пройдений теоретичний матеріал.

Курсова робота є самостійною, творчою практичною роботою, в процесі написання якої студент виявляє своє вміння відбирати, систематизувати і творчо осмислювати технічну документацію, працювати із спеціальною літературою, робити правильні і обґрунтовані висновки, знаходити в них головне, формулювати та логічно викладати свої думки.

Виконання письмових робіт сприяє поглибленому вивченню навчального предмету студентами і є однією з важливих форм перевірки їх знань. Таким чином, курсова робота дає змогу викладачу оцінити якість теоретичних знань студента та його вміння застосовувати їх на практиці.

Згідно з навчальними планами для студентів спеціальності 121 «Інженерія програмного забезпечення» передбачена курсова роботи з дисципліни «Об’єктно-орієнтоване програмування». Написання курсової роботи є завершальним етапом вивчення дисципліни «Об’єктно-орієнтоване програмування». Для написання курсової роботи студенту необхідно вивчити технічну документацію, методичні, інструктивні матеріали та літературні джерела з обраної теми. Тема курсової роботи обирається згідно рекомендованого переліку тем, що не позбавляє можливості студента за погодженням із кафедрою сформулювати тему самостійно. Курсова робота має бути подана на кафедру до початку екзаменаційної сесії та після проходження формальної перевірки допущена до захисту.

МЕТА ТА ЗАВДАННЯ КУРСОВОЇ РОБОТИ

Мета курсової роботи – поглибити теоретичні знання набуті студентами у процесі вивчення дисципліни і виробити вміння застосувати їх у практичній діяльності.

Завдання курсової роботи – навчити студентів користуватися та опрацьовувати технічну літературу, сприяти поглибленому засвоєнню набутих знань студентам з дисципліни «Об'єктно-орієнтоване програмування», ґрунтовному вивченню обраної мови програмування та популярних інструментів розробки. Виконання курсової роботи повинно сприяти формуванню у студента навичок творчої, дослідницької роботи та компетентностей. Професійні компетентності, яких набувають студенти внаслідок вивчення навчальної дисципліни.

Професійні компетентності, яких набувають студенти внаслідок вивчення навчальної дисципліни

Код компетентності	**Назва компетентності**	**Результати навчання**
ЗК1	Здатність до абстрактного мислення, аналізу та синтезу	ПРН1. Аналізувати, цілеспрямовано шукати і вибирати необхідні для вирішення професійних завдань інформаційно-довідникові ресурси і знання з урахуванням сучасних досягнень науки і техніки. ПРН5. Знати і застосовувати відповідні математичні поняття, методи доменного, системного і об'єктно-орієнтованого аналізу та математичного моделювання для розробки програмного забезпечення. ПРН11. Вибирати вихідні дані для проектування, керуючись формальними методами опису вимог та моделювання ПРН13. Знати і застосовувати методи розробки алгоритмів, конструювання програмного забезпечення та структур даних і знань ПРН17. Вміти застосовувати методи компонентної розробки програмного забезпечення.
ЗК2	Здатність застосовувати знання у практичних ситуаціях	
ФК6	Здатність аналізувати, вибирати і застосовувати методи і засоби для забезпечення інформаційної безпеки (в тому числі кібербезпеки)	
ФК8	Здатність застосовувати фундаментальні і міждисциплінарні знання для успішного розв'язання завдань інженерії програмного забезпечення	
ФК11	Здатність реалізовувати фази та ітерації життєвого циклу програмних систем та інформаційних технологій на основі відповідних моделей і підходів розробки програмного забезпечення	

СТРУКТУРА КУРСОВОЇ РОБОТИ

У вирішенні задачі курсової роботи потрібно виділяти наступні основні етапи, кожен з яких, у свою чергу, складається з множини взаємозв'язаних задач:

- постановка задачі;
- проектування програми;
- програмування;
- налагодження та тестування програми;
- оформлення пояснювальної записки та захист проекту.

Основним змістом першого етапу є формулювання призначення програми та постановка цілей, а також виявлення оригінальних задач, які будуть вирішуватися в процесі виконання роботи. У зв'язку з цим виконання етапу потрібно почати із з'ясування (уточнення) основних понять і визначень, що зустрічаються в завдані і даній предметній області та огляду наявних аналогів. Для цих цілей може бути використана рекомендована література та інші джерела.

Після завершення аналітичного дослідження можна переходити до проектування програми. При цьому необхідно визначити структуру вхідних та вихідних даних, структуру функціональних модулів та взаємозв'язків між ними, розробити алгоритми роботи цих модулів та інтерфейс користувача.

На етапі програмування відбувається реалізація запропонованих архітектурних рішень у вигляді програмного коду. Варто зазначити, що крім формальної реалізації алгоритмів необхідно приділяти увагу стилю, дотриманню загально прийнятих та рекомендованих норм та правил найменування функціональних елементів та оформлення програмного коду загалом. Особливу увагу необхідно приділити розробці інтерфейсу користувача.

Наступним етапом розробки є тестування програми та детальний опис структури головного, допоміжного меню та діалогових вікон, а також рекомендації з використання програми.

Оформлення пояснювальної записки є останнім етапом виконання курсового проекту і має за мету надати студентові навичок документування програмного продукту. Документування є завершальним етапом створення програмного виробу для курсової роботи. Вимоги до оформлення пояснювальної записки наведені в наступному розділі.

Структура курсової роботи

Курсова робота повинна містити:

1. Титульний аркуш.
2. Завдання на курсову роботу та календарний план її виконання.
3. Вступ.
4. Основна частина курсової роботи.
5. Висновки.
6. Список використаних джерел (не менше 25, включаючи нормативні акти).
7. Додатки (при необхідності).

Вимоги до змісту курсової роботи. Після затвердження теми курсової роботи студент за погодженням з науковим керівником розробляє план роботи,

підбирає список літератури і опрацьовує необхідні нормативно-правові акти, а також визначає строки і складає орієнтовний графік написання курсової роботи. Обов'язковим правилом вибору теми є те, що йому має передувати належна теоретична підготовка, тобто відповідне оволодіння необхідним матеріалом з предмету даного дослідження.

Курсова робота повинна бути написана відповідно до плану, який розробляється студентом. При роботі над обраною темою, студенту необхідно спочатку вивчити підібрані джерела. При цьому доцільно робити виписки у вигляді цитат або вільного переказу окремих положень. Після вивчення необхідної літератури, зібравши достатній матеріал для висвітлення теми, слід приступити до його повної обробки, упорядкування, аналізу і викладу згідно зі складеним планом. Студент повинен виразно, розбірливо, ясно, зрозуміло та логічно висловлювати свої та запозичені з використаних джерел думки. Необхідно, щоб автор при виконанні роботи висловлював власне ставлення до описуваних подій та оцінок їх у зібраній літературі. Тобто при написанні курсової роботи потрібно уникати поверхневого висвітлення, загальних фраз, а особливо, дослівного переписування з використаних джерел. Останнє призведе до негативної оцінки роботи викладачем і повернення її на доопрацювання.

В загальному, курсова робота виконується акуратно, грамотно, а основні її положення та ідеї мають бути чітко і правильно сформульовані. Структурні елементи курсової роботи повинні відповідати встановленим вимогам:

Титульний аркуш курсової роботи містить найменування міністерства, вищого навчального закладу, факультету та кафедри, де виконана робота; тему курсової роботи; прізвище, ім'я, по батькові автора; вчене звання, наукову ступінь прізвище, ім'я, по батькові наукового керівника; місто і рік **(додаток А).**

Зміст подають на початку курсової роботи, він містить найменування усіх розділів, підрозділів та пунктів, підпунктів (якщо вони мають заголовки), зокрема: вступу, розділів, підрозділів, загальних висновків, списку використаних джерел і додатків.

Перелік умовних позначень, символів, одиниць, скорочень і термінів, а також використані маловідомі скорочення, нові символи, позначення та їх перелік може бути поданий у курсовій роботі у вигляді окремого списку, який розміщується перед вступом. Перелік треба друкувати двома колонками, в яких зліва за абеткою наводять, наприклад, скорочення, а з права – їх детальну розшифровку. Якщо в курсовій роботі спеціальні терміни, позначення, скорочення повторюються менше трьох разів – перелік не складають, а їх розшифровку наводять у тексті при першому згадуванні.

У вступі коротко викладають оцінку сучасного стану проблеми, актуальність даної роботи і підстави для її проведення, ціль роботи й галузь застосування.

Основна частина складається з розділів і підрозділів (пунктів і підпунктів). Кожен розділ повинен починатися з нової сторінки.

Перший розділ – дається аналітичний огляд можливостей побудови систем, що вирішують поставлену задачу. Розглядаються особливості об'єктно-

орієнтованого підходу і мов, що підтримують ООП. Наводиться обґрунтування обраного програмного забезпечення.

Другий розділ – проектний. В цьому розділі можна виділити 2–3 відносно самостійних параграфів, що розкривають теоретичні та практичні аспекти розробки програмного забезпечення.

З'ясовується семантика класів і об'єктів — визначається поведінку і атрибути кожної абстракції, визначається структура вхідних та вихідних даних, структура функціональних модулів та взаємозв'язків між ними, розробляється інтерфейс користувача.

Розділ повинен бути максимально насичений фактичною інформацією (таблиці, схеми, UML діаграми) та всебічно відображати аспекти діяльності програмного забезпечення, що розробляється.

Третій розділ – присвячений реалізації рішень, запропонованих та спроектованих в попередньому розділі. Структурно третій розділ дипломної роботи вміщує 2 – 3 параграфи, які містять детальний опис роботи всіх блоків програмного забезпечення, надається опис отриманих результатів і інструкція з використання розробленої програми.

У висновках викладають найбільш важливі практичні результати, одержані в процесі виконання курсової роботи У висновках необхідно наголосити на якісних та кількісних показниках здобутих результатів: наголосити на архітектурних рішеннях., які було використано, зазначити, які функціональні вимоги вдалось реалізувати, оцінити швидкодію та обчислювальну складність реалізованих алгоритмів

ПРАВИЛА ОФОРМЛЕННЯ КУРСОВОЇ РОБОТИ

Загальні принципи оформлення

Як правило, курсова робота спочатку виконується у чорновому варіанті. Лише після узгодження з керівником дискусійних і сумнівних моментів, врахування зауважень і усунення недоліків, удосконалення структури викладу матеріалу приступають до чистового варіанту. Чистовий варіант курсової роботи потрібно акуратно підшити у папку.

Курсова робота виконується державною (українською) мовою.

Курсова робота виконується на стандартному папері формату А4 (210×297 мм) білого кольору до тридцяти рядків на сторінці. При наборі використовують шрифт Times New Roman 14 з міжрядковим інтервалом 1,5.

Текст необхідно друкувати, залишаючи поля таких розмірів: ліве – не менше 25 мм, праве – не менше 10 мм, верхнє – не менше 20 мм, нижнє – не менше 20 мм.

Обсяг основного тексту курсової роботи повинен становити не менше 30 та не більше 50 сторінок (без врахування таблиць та рисунків).

Кількість використаних джерел у курсовій роботі повинна складати не менше 25 найменувань.

У викладі матеріалу важливо дотримуватись принципу пропорційності, який полягає у дотриманні пропорцій між обсягами окремих частин роботи. При цьому на вступ відводиться 1–2 сторінки, на основну частину роботи 30– 40 сторінок, на титульний лист і зміст роботи – по одній сторінці, висновки та пропозиції – 2 сторінки.

Заголовки структурних частин дипломної роботи «ЗМІСТ», «ВСТУП», «РОЗДІЛ», «ВИСНОВКИ ТА ПРОПОЗИЦІЇ», «СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ», «ДОДАТКИ» (14 шрифт, жирно) пишуть великими літерами симетрично до тексту. Заголовки параграфів друкують маленькими літерами з абзацного відступу. Крапку в кінці заголовка не ставлять. Всі сторінки курсової роботи повинні бути пронумеровані арабськими цифрами без знака № в правому верхньому куті.

Першою сторінкою курсової роботи є титульний лист (приклад подано у **додатку А**), який включають до загальної нумерації сторінок. На ньому номер сторінки не ставлять.

На другій сторінці розміщують завдання на курсову роботу та календарний поан її виконання (бланк завдання подано у **додатку Б**). зміст курсової роботи (приклад подано у **додатку В**). Зверху потрібно написати: **ЗМІСТ**.

З наступної сторінки розпочинається виклад тексту роботи. Кожний розділ пишуть з нової сторінки і обов’язково вказують його найменування (виділяють заголовок). Номер розділу ставлять після слова «РОЗДІЛ», після номера крапку не ставлять, потім з нового рядка друкують заголовок розділу.

Параграфи нумерують у межах кожного розділу. Номер параграфу складається з номера розділу і порядкового номера параграфу, між якими ставлять крапку. В кінці номера параграфу повинна стояти крапка, наприклад:

«2.1» (перший параграф другого розділу). Далі у тому ж рядку друкується заголовок параграфу.

Всі аркуші, на яких розміщені такі структурні частини курсової роботи як зміст, вступ, висновки, список використаних джерел, додатки нумерують звичайним чином. Не нумерують лише їх заголовки, тобто не можна друкувати «1. ВСТУП» або «4. ВИСНОВКИ».

Оформлення таблиць та ілюстрацій

Для аналізу необхідно використовувати таблиці та ілюстрації: схеми, графіки, діаграми тощо.

Цифровий матеріал, як правило, оформляють у вигляді таблиць. Кожна таблиця повинна мати назву, яку розміщують над таблицею симетрично до тексту. Назву і слово «Таблиця» починають з великої літери. Назву не підкреслюють. В дужках над змістом таблиці вказують одиниці виміру. Заголовки граф повинні починатися з великих літер, підзаголовки – з маленьких (якщо вони складають одне речення із заголовком) і з великих (якщо вони **є** самостійними).

Таблиці нумерують послідовно в межах розділу. В правому верхньому куті над відповідним заголовком таблиці розміщують напис «Таблиця» із зазначенням номера. Номер таблиці повинен складатися з номера розділу і порядкового номера таблиці, між якими ставиться крапка, наприклад: «Таблиця 2.5» (п’ята таблиця другого розділу).

При переносі частини таблиці на інший аркуш слово «Таблиця» і її номер вказують один раз справа над першою частиною таблиці, над іншими частинами пишуть «продовження таблиці» і вказують номер таблиці, наприклад: «Продовження таблиці 2.5». Таблицю з великою кількістю граф можна ділити на частини і розміщувати одну частину під іншою в межах однієї сторінки.

Ілюстрації позначають словом «Рис.» і нумерують послідовно в межах розділу (за винятком ілюстрацій, поданих у додатках). Номер ілюстрації повинен складатися з номера розділу і порядкового номера ілюстрації, між якими ставиться крапка. Наприклад, **Рис. 1.1** (перший рисунок першого розділу). Номер ілюстрації, її назва і пояснювальні підписи розміщують послідовно під ілюстрацією. Якщо у роботі подано ілюстрацію, то її нумерують за загальними правилами.

Посилання на таблиці та ілюстрації не варто оформляти як самостійні фрази, в яких лише повторюється те, що міститься у підписі. У тому місці, де викладається матеріал, пов’язаний з таблицею чи ілюстрацією, на якому необхідно зосередити увагу читача, розміщують посилання у вигляді виразу у круглих дужках «(табл. 2.5)» або зворот типу: «... як це видно з рис. 1.1». (приклади оформлення таблиць, рисунків наведено у **додатках Г, Д**).

Написання формул

При необхідності використовують формули. їх виділяють з тексту вільними рядками. Над і під кожною формулою потрібно залишити не менше одного вільного рядка. Якщо рівняння чи формула не вміщаються в одному рядку, їх слід перенести після знака рівності (=) або після інших розділових, знаків (+, –, :, ×). Невеликі і нескладні формули, що не мають самостійного значення, вписують всередині рядків тексту.

Пояснення значень символів і числових коефіцієнтів подають безпосередньо під формулою в тій послідовності, в якій вони наведені у формулі. Значення кожного символу і числового коефіцієнта треба подавати з нового рядка, який починають зі слова «де» без двокрапки. Нумерувати слід лише ті формули, на які є посилання у наступному тексті. Формули у курсовій роботі (якщо їх більше однієї) нумерують у межах розділу. Номер формули складається з номера розділу і порядкового номера формули в розділі, між якими ставлять крапку. Номери формул пишуть біля правого поля аркуша на рівні відповідної формули в круглих дужках, наприклад: (2.3) (третя формула другого розділу) (**додаток Е**).

Посилання на використані джерела

При написанні курсової роботи студент повинен робити посилання на використані літературні джерела. Посилання робляться тоді, коли в тексті використовують цитати чи вислови, переповідається своїми словами думка того чи іншого автора, наводяться цифри і факти, використовуються таблиці або ілюстрації з певного джерела тощо. Вони дають необхідну інформацію щодо процитованого матеріалу, допомагають з’ясувати його зміст, мову тексту, а також дають змогу відшукати документи і перевірити їх достовірність.

Загальні вимоги до цитування:

– текст цитати береться в лапки і наводиться дослівно;

– цитування повинно бути повним (у випадку пропуску слів в цитаті ставляться три крапки), без перекручень думок автора;

– кожна цитата повинна мати посилання на джерело;

– при непрямому цитуванні слід також посилатись на джерело, а думка автора повинна передаватись максимально точно.

Посилання в тексті курсової роботи роблять згідно їх переліку у квадратних дужках, наприклад [2, с. 25], де перша цифра – номер джерела у списку використаних джерел, друга – сторінка, з якої взята цитата. Якщо посилання робиться лише на джерело (без використання цитати), – у квадратних дужках вказується тільки одна цифра, яка відображає порядковий номер джерела у списку використаних джерел, наприклад [25]. Посилатися слід на останні видання публікації.

Приклад.

Цитата в тексті: Економісти Е.С. Хандріксен та М.Ф. Ван Бреда зазначають, що «витрати є несприятливим рухом ресурсів, який зменшує

прибуток фірми. Проте не кожен несприятливий рух є витратами. Більше того, витрати – це вико ристання або споживання товарів і послуг у процесі отримання доходу» [26, с. 243].

Відповідне джерело у переліку посилань:

26. Хандриксен Е.С., Ван Бреда М.Ф. Теория бухгалтерского учета / гл. ред. Я.В. Соколов. М. : Финансы и статистика, 2009. 576 с.

На всі ілюстрації, формули та таблиці у роботі повинні бути посилання в тексті. Посилання на них вказують порядковим номером. Наприклад:

– ілюстрації – «рис. 1.1»;

– «... у формулі 2.5»;

– «...в табл. 2.5».

У повторних посиланнях на таблиці та ілюстрації пишуть скорочено слово «дивись», наприклад: «(табл. 3.2)». При цьому слово «таблиця» в тексті пишуть скорочено. Посилаючись на джерело, слід мати на увазі, що кількість посилань має бути оптимальною, тобто не надто надмірною чи мізерною.

Формування списку використаних джерел

Список використаних джерел розміщують після висновків. У списку повинні наводитись лише ті джерела, які дійсно використовувалися автором при написанні курсової роботи.

Список використаних джерел формується одним в алфавітному порядку прізвищ перших авторів або заголовків.

Слід наголосити, що назва джерел подається тією мовою, якою вони видані. Не потрібно перекладати їх на українську.

Приклад оформлення списку використаних джерел наведено в **додатку Ж.** Бібліографічний опис джерел складають відповідно до чинних стандартів з бібліотечної та видавничої справи, міжнародних і державного стандартів з обов’язковим наведенням назв праць.

Оформлення додатків

Додатки оформляються як продовження курсової роботи на наступних її сторінках або у вигляді окремої частини, розміщуючи їх у порядку появи посилань у тексті.

Додаток повинен мати заголовок. Він пишеться угорі малими літерами з першої великої симетрично відносно тексту сторінки. Посередині рядка над заголовком малими літерами з першої великої пишуть (друкують) слово «Додаток» і велика літера, що позначає додаток.

Позначаючи послідовно додатки великими літерами української абетки із нумерації слід виключити літери Ґ, Є, З, І, Ї, Й, О, Ч, Ь. Наприклад, Додаток А, Додаток Б.

При оформленні додатків окремою частиною на окремому аркуші пишуть (друкують) великими літерами слово «ДОДАТКИ»

ЗАХИСТ ТА КРИТЕРІЇ ОЦІНЮВАННЯ КУРСОВОЇ РОБОТИ

Захист курсових робіт відбувається у дні та години, вказані у графіку захисту курсових робіт або відповідно до розкладу занять. Під час захисту студент коротко висловлює зміст роботи, відповідає на запитання викладача та зауваження, висловлені у рецензії. Оцінка за курсову роботу залежить від повноти, викладу теми правильності висвітлення питань, рівня використання теоретичного, методологічного та фактологічного матеріалу; належного оформлення та рівня захисту роботи. Підсумкову, диференційовану оцінку курсової роботи визначає комісія, що призначається кафедрою.

Курсова робота не допускається до захисту і повертається на доопрацювання, якщо:

- роботу представлено на кафедру (на перевірку) для рецензування з порушенням термінів, установлених кафедрою (викладачем, який викладає дану дисципліну);
- роботу написано на тему, що не включена до переліку тем курсових робіт з даної дисципліни або не погоджена з викладачем;
- роботу виконано не самостійно;
- структура і логіка побудови плану роботи не відповідає вимогам та темі курсової роботи;
- курсову роботу не зброшуровано (тобто аркуші не скріплені).

У процесі оцінювання враховується низка важливих показників якості курсової роботи, а саме:

- чіткість формулювання мети і завдання курсової роботи, складність досліджуваних у роботі проблем, відповідність логічної побудови роботи поставленим цілям і завданням;
- якість і глибина теоретико-методичного аналізу проблеми; широта й адекватність методологічного апарату; якість критичного огляду літературних джерел, наявність наукової полеміки, посилань на літературні джерела та визначення власної думки студента-автора курсової роботи;
- системність і глибина аналізу статистичних та фактичних матеріалів, наявність і переконливість узагальнень і висновків з аналізу, наявність і якість ілюстративних матеріалів у тексті роботи, використання економіко-математичних методів, соціологічних та інших досліджень;
- наявність та логічний зв'язок заходів, що пропонуються для вирішення проблеми, що досліджується у роботі, з проведеним у роботі аналізом фактичних та статистичних матеріалів, їх актуальність, реальність та економіко-соціальна обґрунтованість, наявність альтернативних підходів до вирішення визначених проблем;
- володіння культурою презентації, вміння стисло (в межах регламенту), послідовно й чітко викласти сутність і результати дослідження, здатність аргументовано захищати свої пропозиції, думки, погляди; повнота і ґрунтовність відповідей на запитання членів комісії, що приймає захист курсових робіт, на зауваження і пропозиції, що містяться у рецензії на курсову роботу.

АКАДЕМІЧНА ДОБРОЧЕСНІСТЬ

Студенти при виконанні курсової роботи з дисципліни «Об'єктно-орієнтоване програмування» повинні дотримуватись академічної доброчесності, тобто самостійно виконувати курсову роботу, здійснювати посилання на джерела інформації у разі використання певних тверджень й відомостей, надавати достовірну інформацію про результати власних розробок та досліджень. В курсовій роботі доцільно уникати плагіату, фальсифікації та фабрикування матеріалів дослідження.

Академічний плагіат – це оприлюднення (частково або повністю) наукових результатів, отриманих іншими особами, як результатів власного дослідження та/або відтворення опублікованих текстів інших авторів без зазначення авторства.

Академічним плагіатом є:

1) відтворення в тексті роботи без змін, з незначними змінами, або в перекладі тексту іншого автора (інших авторів), обсягом від речення і більше, без посилання на автора (авторів) відтвореного тексту;

2) відтворення в тексті роботи повністю або частково, тексту іншого автора (інших авторів) через його перефразування чи довільний переказ без посилання на автора (авторів) відтвореного тексту;

3) відтворення в тексті роботи наведених в іншому джерелі цитат з третіх джерел без вказування, за яким саме безпосереднім джерелом наведена цитата;

4) відтворення в тексті роботи наведеної в іншому джерелі науково-технічної інформації (крім загальновідомої) без вказування на те, з якого джерела взята ця інформація.

Фальсифікація розуміється, як свідома зміна чи модифікація вже наявних даних, що стосуються власної діяльності або діяльності інших учасників процесу реалізації державної політики у сфері якості освіти, зокрема підробка підписів в офіційних документах.

Під **фабрикуванням** мається на увазі вигадування даних чи фактів, що використовуються у власній діяльності чи діяльності інших учасників в процесі реалізації державної політики в сфері якості освіти.

ТЕМИ КУРСОВИХ РОБОТ

Основна частина курсової роботи має містити виконання наступних завдань:

1. Створити класовий тип згідно із варіантом.
2. Для створення об'єкту динамічного типу передбачити відповідні конструктори (конструктор по замовчанню, конструктор із параметрами, конструктор копіювання).
3. Реалізувати компонентні методи відповідно до варіанту.
4. Розробити демонстраційно - тестуючу програму. Виконати тестування розробленого програмного коду.

1. Класовий тип для обробки текстової інформації.
2. Класові типи для роботи з чисельними матрицями та векторами.
3. Бібліотечні засоби для розв'язування задач лінійної алгебри.
4. Класовий тип для роботи з структурами типу «Вектор».
5. Класовий тип для роботи з структурами типу «Бітовий вектор».
6. Клас для роботи з структурами типу «Зв’язний список».
7. Клас для обслуговування структур типу «Черга».
8. Клас для обслуговування структур типу «Двобічна черга».
9. Клас для обслуговування структур типу «Черга з пріоритетами».
10. Клас для роботи з структурами типу «Множина».
11. Клас для роботи з структурами типу «Мультимножина».
12. Клас для роботи з структурами типу «Словник».
13. Клас для роботи з структурами типу «Словник з повтореннями».
14. Клас для роботи з структурами типу «Стек».
15. Клас для роботи з структурами типу «Дерево».
16. Клас для роботи з структурами типу «Врівноважене дерево».
17. Класи для роботи з неорієнтованими графами: таблиця зв’язків та матриця суміжності.
18. Класи для роботи з орієнтованими графами: таблиця зв’язків та матриця суміжності.
19. Класи для компактного подання неорієнтованих та орієнтованих графів.
20. Клас для роботи з дводольними графами.
21. Клас для роботи з мультиграфами.
22. Клас для роботи з гіперграфами.
23. Клас для роботи з табличними інваріантами графів.
24. Розробка та реалізація алгоритму упізнання ізоморфності графів.
25. Модель лінії затримки на основі структури типу «Черга».
28. Комп’ютерна модель системи терморегулювання.
29. Алгебра бінарних відношень. Програмна реалізація.

ВАРІАНТИ ЗАВДАНЬ ДО КУРСОВОЇ РОБОТИ

Варіант 1. Класовий тип для обробки текстової інформації

1. Створити класовий тип String - символьний масив динамічного типу із змінною кількістю символів. Максимальний розмір масиву - у межах вільного простору оперативної пам’яті.
2. Реалізувати компонентні методи:
   - довжина рядка;
   - пошук субрядку;
   - пошук субрядку та його заміна;
   - приведення усіх букв до верхнього регістру;
   - приведення усіх букв до нижнього регістру;
   - конкатенація;
   - обмін значеннями двох змінних типу String.

Література: [1-5, 7].

Варіант 2. Класові типи для роботи з чисельними матрицями та векторами

1. Створити класовий тип Matrix - двовимірний чисельний масив динамічного типу із змінними розмірами.
2. Створити класовий тип Vector - одновимірний чисельний масив динамічного типу із змінними розмірами.
3. Для конкретизації типів матриці та вектора використати шаблони.
4. Реалізувати компонентні методи:
   - транспонування матриці;
   - множення матриць;
   - складання та віднімання;
   - порівняння матриць;
   - множення матриці на вектор;
   - скалярний добуток векторів;
   - обчислення норми матриці;
   - пошук седлових точок матриці;
   - пошук мінімальних та максимальних елементів.

Література: [1-5, 7, 14].

Варіант 3. Бібліотечні засоби для розв'язування задач лінійної алгебри

1. Створити класові типи - чисельна квадратна матриця та одновимірний масив динамічного типу із змінними розмірами.
2. Створити перевантажені операції:
3. Реалізувати компонентні методи:
   - додавання та віднімання матриць;
   - додавання та віднімання векторів;
   - множення матриці на вектор;
   - обертання матриці;
   - порівняння матриць;

- розв'язування системи лінійних рівнянь;
- скалярне множення векторів.
- обчислення визначника матриці;
- обчислення довжини вектору.

Література: [1-5, 7, 14].

Варіант 4. Класовий тип для роботи з структурами типу «Вектор»

1. Створити параметризований клас для роботи з структурою типу Vector - одновимірний динамічний масив із змінною кількістю елементів. Тип елементу масиву надається параметром шаблону.
2. Передбачити конструктор для створення вектору та ініціалізації його звичайним одновимірним масивом відповідного типу.
3. Реалізувати компонентні методи:
   - get - доступу до елемента за індексом;
   - concat - конкатенації двох векторів;
   - equals - порівняння двох векторів.
   - size - кількість елементів вектора;
   - front - посилання на перший елемент;
   - back - посилання на останній елемент;
   - swap - обмін значеннями з іншим вектором;
   - insert - уставити елемент у надану позицію;
   - push_back - додати новий елемент у кінець вектора;
   - pop_back - вилучити останній елемент;
   - erase - вилучити елемент у наданій позиції;
   - find - знайти елемент у векторі і повернути його позицію;
   - accumulate - накопичення суми або добутку;
   - for_each - обробка елементів по наданій процедурі;
   - max , min - пошук максимального та мінімального елементів;
   - sort - упорядкування елементів у порядку зростання або зменшення.

Література: [1-6].

Варіант 5. Класовий тип для роботи з структурами типу «Бітовий вектор»

1. Створити параметризований клас для роботи з структурою типу BitVector, яка являє собою одновимірний динамічний масив із змінною кількістю елементів бітового типу.
2. Реалізувати компонентні методи:
   - get - доступу до бітового елемента за індексом;
   - concat - конкатенації двох бітових векторів;
   - equals - порівняння;
   - бітові операції AND, OR, XOR, NOT.
   - push_back - додання нового бітового елементу у кінець вектора;
   - pop_back - вилучення останнього бітового елементу;
   - size - кількість елементів вектора;
   - front - посилання на перший елемент;
   - back - посилання на останній елемент;
   - swap - обмін значеннями з іншим вектором;

- num_true - кількість елементів, що дорівнюють true;
- num_false - кількість елементів, що дорівнюють false.

Література: [1-6].

Варіант 6. Клас для роботи з структурами типу «Зв'язний список»

1. Створити клас List для роботи із структурами типу «Двонапрямлений зв'язний список». Тип елементу списку надається параметром шаблону. Елемент двонапрямленого зв'язного списку містить у собі покажчики на наступний та попередній елементи структури last і next. Значення last для першого елементу і значення next для останнього дорівнюють NULL.
2. Реалізувати компонентні методи:
   - concat - конкатенації;
   - equals - порівняння;
   - size - кількість елементів списку;
   - front - посилання на перший елемент;
   - back - посилання на останній елемент;
   - swap - обмін значеннями з іншим списком;
   - insert - уставити елемент у надану позицію;
   - push_back - додати новий елемент у кінець списку;
   - pop_back - вилучити останній елемент;
   - push_front - додати новий елемент у початок списку;
   - pop_front - вилучити останній елемент;
   - find - перевірити, чи є наданий елемент у списку;
   - accumulate - накопичення суми або добутку;
   - for_each - обробка елементів по наданій процедурі;
   - max , min - пошук максимального та мінімального елементів;
   - sort - упорядкування елементів по зростанню;
   - unique - вилучення повторень (якщо два сусідніх елементи співпадають, один з них вилучається);
   - remove - вилучення всіх елементів із списку з наданим значенням;
   - merge - об'єднання (конкатенація) з наданим списком;

Література: [1-5, 9].

Варіант 7. Клас для обслуговування структур типу «Черга»

1. Створити клас Queue для роботи із структурами типу «Черга». Тип елементу черги надається параметром шаблону. Черга є структурою даних типу «Перший увійшов - перший вийшов» (FIFO - first-in, first-out), вона нагадує ескалатор у метро.
2. Реалізувати компонентні методи:
   - equals - порівняння;
   - get - доступу по індексу;
   - size - кількість елементів;
   - empty - повертає значення true, якщо черга пуста;
   - front - посилання на перший елемент;
   - back - посилання на останній елемент;

- swap - обмін значеннями з іншою чергою;
- push - додати новий елемент у кінець черги;
- pop - вилучити перший елемент;
- find - перевірити, чи є наданий елемент у черзі;
- accumulate - накопичення суми або добутку;
- for_each - обробка елементів по наданій процедурі;
- max , min - пошук максимального та мінімального елементів.

Література: [1-5, 9].

Варіант 8. Клас для обслуговування структур типу «Дек»

1. Створити клас Deque для роботи із структурами типу «Дек» (Двобічна черга). Тип елементу дека надається параметром шаблону.

Дек є черга з двостороннім доступом (deque - «double ended queue»). Деки використовуються з алгоритмами, які потребують доступу до голови і хвоста структури. Він поєднує у собі можливості вектора і списку. Для нього визначається операція індексування. Додавати і вилучати елементи дека можна з обох сторін.

2. Реалізувати компонентні методи:

- порівняння,;
- доступу по індексу;
- size - кількість елементів;
- empty - повертає значення true, якщо дек пустий;
- front - посилання на перший елемент;
- back - посилання на останній елемент;
- swap - обмін значеннями з іншим деком;
- push_front - додати новий елемент у початок дека;
- push_back - додати новий елемент у кінець дека;
- pop_front - вилучити перший елемент;
- pop_back - вилучити останній елемент;
- find - перевірити, чи є наданий елемент у деку;
- accumulate - накопичення суми або добутку;
- for_each - обробка елементів по наданій процедурі;
- max , min - пошук максимального та мінімального елементів.

Література: [1-5, 9].

Варіант 9. Клас для обслуговування структур типу «Черга з пріоритетами»

1. Створити клас PriorQueue для роботи із структурами типу «Черга з пріоритетами». Тип елементу структури надається параметром шаблону.

Структура типу «Черга з пріоритетами» має такі ж властивості, як і проста черга, але елементи її завжди розташовані у певному порядку. Упорядкування структури виконується автоматично при додаванні нового елемента. Тому операція pop вилучає найбільший (чи найменший) елемент черги.

2. Реалізувати компонентні методи:

- equals - порівняння;

- size - кількість елементів;
- empty - повертає значення true, якщо черга пуста;
- back - посилання на останній елемент;
- swap - обмін значеннями з іншою чергою;
- push - додати новий елемент у кінець черги;
- pop - вилучити найбільший (найменший) елемент;
- find - перевірити, чи є наданий елемент у деку;
- accumulate - накопичення суми або добутку;
- for_each - обробка елементів по наданій процедурі.

Література: [1-5, 9].

Варіант 10. Клас для роботи з структурами типу «Множина»

1. Створити клас Set для роботи із структурами типу «Множина». Кожен елемент множини має унікальне значення. Тип елементу структури надається параметром шаблону.

Множина є набір елементів, кожен з яких має унікальне значення. При додаванні елементу (операція include), який вже є у множині, він не додається. Елементи множини називають ключами. Для цієї структури визначаються операції, звичайні для математичних множин - об'єднання, переріз, віднімання, доповнення. Елементи множини автоматично упорядковуються, це дає змогу використовувати швидкі операції пошуку елемента із наданим значенням.

2. Реалізувати компонентні методи:

- equals - порівняння;
- union - об'єднання множин;
- section - переріз множин;
- subtract - віднімання;
- size - кількість елементів;
- empty - повертає значення true, якщо множина пуста;
- swap - обмін значеннями з іншою множиною;
- include - додати новий елемент у множину;
- exclude - вилучити наданий елемент із множини;
- compl - отримати значення, яке є доповненням до наданої множини;
- find - перевірити, чи є наданий елемент у множині;
- accumulate - накопичення суми або добутку для всіх елементів множини;
- for_each - змінювання елементів згідно з наданою процедурою.

Література: [1-5, 9].

Варіант 11. Клас для роботи з структурами типу «Мультимножина»

1. Створити клас MultiSet для роботи із структурами типу «Мультимножина». Тип елементу структури надається параметром шаблону.

Ця структура має такі ж властивості, як і множина, але елемент мультимножини не обов'язково має унікальне значення (елементи можуть повторюватись).

2. Реалізувати компонентні методи:

- equals - порівняння;
- size - кількість елементів;

- empty - повертає значення true, якщо мультимножина пуста;
- swap - обмін значеннями з іншою структурою того ж типу;
- include - додати новий елемент у мультимножину;
- exclude - вилучити наданий елемент із мультимножини;
- find - перевірити, чи є наданий елемент у мультимножині;
- accumulate - накопичення суми або добутку для всіх елементів структури;
- for_each - змінювання елементів згідно з наданою процедурою.

Література: [1-5, 9].

Варіант 12. Клас для роботи з структурами типу «Словник»

1. Створити клас Map для роботи із структурами типу «Словник» (таку структуру називають ще асоціативний масив або відображення). Елементом цієї структури є пара: ключ і значення. Значення ключа є унікальним. Словник автоматично упорядковується по ключам. Типи компонентів пари - елемента словника надається параметрами шаблону. Для словника звичайно визначаються операція індексування для швидкого доступу до ключа і операція порівняння для ключів. Застосовуються такі структури у задачах із словниками і асоціативними базами даних.
2. Реалізувати компонентні методи:
   - equals - порівняння;
   - size - кількість елементів;
   - size - кількість елементів;
   - empty - повертає значення true, якщо словник пустий;
   - swap - обмін значеннями з іншим словником;
   - include - додати елемент з наданим ключем у словник;
   - exclude - вилучити елемент з наданим ключем із словника;
   - find - знайти значення, яке відповідає наданому ключу;
   - for_each - змінювання елементів згідно з наданою процедурою.

Література: [1-5, 9].

Варіант 13. Клас для роботи з структурами типу «Словник з повтореннями»

1. Створити клас MultiMap для роботи із структурами типу «Словник з повтореннями» (мультивідображення). Типи компонентів пари - елемента словника надається параметрами шаблону.

Ця структура має такі ж властивості, як і попередня (словник), але ключі елементів мультивідображення не обов’язково мають унікальне значення, структура може вміщувати елементи з однаковими значеннями ключів. Мультивідображення , також як і словник, автоматично упорядковується по ключам.

2. Реалізувати компонентні методи:
   - equals - порівняння;
   - size - кількість елементів;
   - empty - повертає значення true, якщо словник порожній;
   - swap - обмін значеннями з іншим словником;

- include - додати елемент з наданим ключем у словник;
- erase - вилучити всі елементи з наданим ключем із словника і повернути кількість вилучених елементів;
- find - знайти наступне значення, яке відповідає наданому ключу;
- for_each - змінювання елементів згідно з наданою процедурою.

Література: [1-5, 9].

Варіант 14. Клас для роботи з структурами типу «Стек»

1. Створити клас для роботи із структурами типу «Стек» (структура типу «останній ввійшов - перший вийшов», або LIFO - last-in, first-out). Тип компонентів структури надається параметрами шаблону.

Структура типу «Стек» може створюватись на основі однієї з таких структур: вектор, дек або список. В усіх випадках для елементів стека визначаються однаковий набір операцій. Стек повинен піддержувати операції порівняння.

Структури цього типу застосовуються у багатьох алгоритмах, особливо для збереження і здобуття інформації у певному порядку (наприклад, для результатів обчислень підвиразів у електронному калькуляторі).

2. Реалізувати компонентні методи:

- equals - порівняння;
- size - кількість елементів у поточний момент;
- empty - повертає значення true, якщо стек пустий;
- swap - обмін значеннями з іншим стеком;
- push - ввести значення у стек;
- pop - вилучити значення з вершини стека;
- top - доступ до вершини стека.

Література: [1-5, 9].

Варіант 15. Клас для роботи з структурами типу «Дерево»

1. Створити клас Tree для роботи із структурами типу «Дерево». Тип елементу структури надається параметром шаблону. Розробити метод адресації елемента дерева.

2. Для створення об'єкту динамічного типу і правильного його вилучення передбачити відповідні конструктори та деструктори. Для ініціалізації об'єктів передбачити конструктор копіювання та конструктори з параметрами.

2. Реалізувати компонентні методи:

- equals - порівняння;
- pop - отримати наступний елемент дерева (у порядку його послідовного обходу у глибину);
- numbel - загальна кількість вузлів дерева;
- numbl - кількість кінцевих вузлів дерева;
- height - висота дерева;
- empty - повертає значення true, якщо дерево пусте;
- swap - обмін значеннями двох змінних типу Tree;
- include - додати новий елемент у дерево;
- exclude - вилучити наданий елемент із дерева;

- find - перевірити, чи є наданий елемент у дереві;
- accumulate - накопичення суми або добутку для всіх елементів дерева;
- for_each - змінювання елементів згідно з наданою процедурою.

Література: [1-5, 9].

Варіант 16. Клас для роботи з структурами типу «Врівноважене дерево»

1. Створити клас AVLTree для роботи із структурами типу «Врівноважене дерево». Тип елементу структури надається параметром шаблону. Розробити метод адресації елемента дерева. Передбачити, щоб після виконання операції додавання нового вузла, дерево автоматично врівноважувалось.
2. Реалізувати компонентні методи:
   - equals - порівняння;
   - pop - отримати наступний елемент дерева (у порядку його послідовного обходу у глибину);
   - numbel - загальна кількість вузлів дерева;
   - numbl - кількість кінцевих вузлів дерева;
   - height - висота дерева;
   - empty - повертає значення true, якщо дерево пусте;
   - swap - обмін значеннями двох змінних типу Tree;
   - include - додати новий елемент у дерево;
   - exclude - вилучити наданий елемент із дерева;
   - find - перевірити, чи є наданий елемент у дереві;
   - accumulate - накопичення суми або добутку для всіх елементів дерева.

Література: [1-5, 9].

Варіант 17. Класи для роботи з неорієнтованими графами: таблиця зв'язків та матриця суміжності

1. Створити класові типи - таблиця зв'язків та матриця суміжності неорієнтованого графа.
2. Для створення об'єктів динамічного типу і правильного їх знищення передбачити відповідні конструктори та деструктори. Для ініціалізації об'єктів передбачити конструктори копіювання та відповідні конструктори з параметрами і конструктори перетворення.
2. Реалізувати компонентні методи:
   - вилучення даної вершини із графа;
   - добавлення вершини у граф;
   - вилучення даного ребра із графа;
   - добавлення даного ребра у граф;
   - функцію, що повертає впорядкований вектор степенів вершин.

Література: [1-5, 7, 10-12].

Варіант 18. Класи для роботи з орієнтованими графами: таблиця зв'язків та матриця суміжності

1. Створити класові типи - таблиця зв'язків та матриця суміжності орієнтованого графа.

2. Реалізувати компонентні методи:
- процедуру вилучення даної вершини із графа;
- процедуру добавлення вершини у граф;
- процедуру вилучення даного ребра із графа;
- процедуру добавлення даного ребра у граф;
- функцію, що повертає впорядкований вектор степенів вершин.
- Література: [1-5, 7, 10-12].

Варіант 19. Класи для компактного подання неорієнтованих та орієнтованих графів

1. Створити класові типи, що відображають бінарні коди неорієнтованого та орієнтованого графа.
2. Реалізувати компонентні методи:
- процедуру вилучення даної вершини із графа;
- процедуру додавання вершини у граф;
- процедуру вилучення даного ребра із графа;
- процедуру добавлення даного ребра у граф;
- функцію, що повертає впорядкований вектор степенів вершин;
- процедуру, що формує таку множину графів, що утворюється при додаванні одного ребра у даний граф.

Література: [1-5, 7, 10-12].

Варіант 20. Клас для роботи з дводольними графами

1. Розробити класовий тип для подання та виконання операцій над дводольними графами.
2. Розробити алгоритм упізнання дводольності графа. Застосувати метод, що базується на хвильовому процесі.
3. Реалізувати компонентні методи:
- процедуру вилучення даної вершини із графа;
- процедуру додавання вершини у граф;
- процедуру вилучення даного ребра із графа;
- процедуру добавлення даного ребра у граф;
- функцію, що повертає впорядковані вектори степенів вершин першої та другої долі.

Література: [1-5, 7, 10-12].

Варіант 21. Клас для роботи з мультіграфами

1. Створити класовий тип для виконання операцій з мультіграфами.
2. Реалізувати компонентні методи:
- процедуру вилучення даної вершини із графа;
- процедуру добавлення вершини у граф;
- процедуру вилучення даного ребра із графа;
- процедуру добавлення даного ребра у граф;
- функцію, що повертає впорядкований вектор степенів вершин.

Література: [1-5, 10-12].

Варіант 22. Клас для роботи з гіперграфами

1. Створити класовий тип для виконання операцій з гіперграфами.
Застосувати уявлення гіперграфу як напрямлений дводольний граф.
2. Реалізувати компонентні методи:
   - процедуру вилучення даної вершини із графа;
   - процедуру добавлення вершини у граф;
   - процедуру вилучення даного ребра із графа;
   - процедуру добавлення даного ребра у граф;
   - функцію, що повертає впорядкований вектор степенів вершин;
   - функцію, що повертає впорядкований вектор потужностей ребер.

Література: [1-5, 10-12].

Варіант 23. Клас для роботи з табличними інваріантами графів

1. Створити класовий тип для табличних інваріантів графа першого порядку.
2. Реалізувати компонентні методи:
   - функцію, що повертає скалярне інтегроване значення інваріанту;
   - функцію, що повертає впорядкований вектор степенів вершин;
   - функції, що повертають діаметр та радіус графа;
   - функцію, що упізнає дводольність графа.

Література: [1-5, 10-12].

Варіант 24. Розробка та реалізація алгоритму упізнання ізоморфності графів

1. Створити алгоритм та програму встановлення ізоморфності двох даних графів переборним методом.
2. Побудувати демонстраційно-тестуючу програму, що розв'язує такі задачі:
   - задачу класифікації для даної множини графів;
   - задачу класифікації для вершин даного графа;
   - задачу класифікації для ребер даного графа.

3. Виконати дослідження обчислювальної ефективності створеного алгоритму упізнання ізоморфності графів.
Література: [1-5, 10-12].

Варіант 25. Модель лінії затримки на основі структури типу «Черга»

1. Створити комп’ютерну модель узагальненої лінії затримки на основі структури типу «Черга», застосовуючи Java Collections Framework.
2. Модель повинна відображати такі ефекти:
затримку сигналу, що пов’язана з скінченою швидкістю розповсюдження сигналу;
зниження величини сигналу завдяки поглинанню енергії у матеріалі лінії задерки;
модифікацію форми сигналу як результат дії ефекту дисперсії.
3. Передбачити у програмі графічне виведення для сигналів, що діють на вході і на виході лінії затримки.
4. Виконати розрахунки для сигналів прямокутної та трикутної форми.

Література: [1-5].

Варіант 26. Побудова засобів обробки даних на основі Java Collections Framework

1. Розробити бібліотеку процедур та функцій для побудови програм, де застосовуються операції обробки чисельних даних методом найменших квадратів на основі Java Collections Framework.
2. Передбачити процедури та функції для розв'язування таких задач:
   - апроксимація даних лінійною функцією;
   - апроксимація даних поліномом даної степені;
   - апроксимація даних кубічними сплайнами.
3. Для побудови процедур створити і застосувати класові типи «Динамічна матриця» і «Динамічний вектор».

Література: [1-5, 6].

Варіант 27. Побудова засобів для обчислень з довільною точністю на основі Java Collections Framework

1. Створити класовий тип Large для подання значення дійсного типу з наданим числом двійкових розрядів мантиси. Застосувати засоби стандартної бібліотеки Java Collections Framework.
2. Реалізувати компонентні методи:
   - присвоєння;
   - додавання;
   - віднімання;
   - множення;
   - ділення;
3. На основі створеного класу написати програму обчислення константи π з наданим числом десяткових розрядів.

Література: [1-6].

Варіант 28. Комп'ютерна модель системи терморегулювання

1. Розробити засоби комп'ютерного моделювання систем автоматичного регулювання температурою об'єктів. Створити класи:
   - об'єкт;
   - зовнішнє середовище;
   - регулятор;
   - функція цілі.

Передбачити можливість регулювання за законами пропорційного, інтегрального та диференціального регулювання.

2. Створити компонентні функції (процедури) для кожного класу для отримання нового стану відповідного об'єкту у наступний момент часу. Для передачі повідомлень від одного об'єкту до другого застосувати дружні функції.
3. Усі класи створити з урахуванням правил об'єктно-орієнтованого програмування.

4. На базі створених класів написати програму моделювання та дослідження систем терморегулювання з різними законами регулювання.
5. Дослідити можливості створених засобів, сформулювати задачі проектування, які можуть бути розв’язані за допомогою створених засобів.
Література: [1-7].

Варіант 29. Алгебра бінарних відношень. Програмна реалізація
1. Створити класовий тип **binrel** для роботи з бінарними відношеннями.
2. Для створення об'єкту динамічного типу і правильного його вилучення передбачити відповідні конструктори та деструктори. Для ініціалізації об'єктів передбачити конструктор копіювання.
2. Реалізувати компонентні методи:
- знаходження зворотнього відношення по наданому відношенню;
- отримання композиції двох відношень.
- функцію, яка визначає, чи є надане відношення рефлексивним;
- функцію, яка визначає, чи є надане відношення симетричним;
- функцію, яка визначає, чи є надане відношення антисиметричним;
- функцію, яка визначає, чи є надане відношення транзитивним;
- функцію, яка визначає, чи є надане відношення функцією;
- функцію, яка визначає, чи є надане відношення ін’єктивною функцією;
- функцію, яка визначає, чи є надане відношення сюр’єктивною функцією;
- функцію, яка визначає, чи є надане відношення бієктивною функцією;
- функцію, яка визначає, чи є надане відношення відношенням частинного порядку;
- функцію, яка визначає, чи є надане відношення відношенням лінійного порядку;
- функцію, яка визначає, чи є надане відношення відношенням еквівалентності.

Література: [1-5, 15].

ЛІТЕРАТУРА

1. Эккель Б. Философия Java.: 4-е изд. СПб.: Питер, 2009. 640 с.
2. Хорстманн К. С., Корнелл Г. Библиотека профессионала. Java 2. Том 1. Основы, 7-е изд.: Пер. с англ. М.: Издательский дом "Вильямс", 2007. 896 с.
3. Хорстманн, К., С., Корнелл, Г. Библиотека профессионала. Java 2. Том 2. Тонкости программирования, 7-е изд.: Пер. с англ. М.: Издательский дом "Вильямс", 2007. 1168 с.
4. Шилдт Г. Полный справочнык по Java. Java SE 6 Edition, 7-е изд.: Пер. с англ. – М.: Издательский дом "Вильямс", 2007. 1034 с.
5. Абельсон Х. Структура и интерпретация компьютерных программ. Абельсон Х., Джеральд Дж. С., Сассман Дж. М.: Добросвет, 2006. 608 с.
6. Хорстманн К. С., Корнелл Г. Библиотека профессионала. Java 2. Том 1. Основы, 7-е изд.: Пер. с англ. М.: Издательский дом "Вильямс", 2007. 896 с.
7. Хорстманн, К., С., Корнелл, Г. Библиотека профессионала. Java 2. Том 2. Тонкости программирования, 7-е изд.: Пер. с англ. М.: Издательский дом "Вильямс", 2007. 1168 с.
8. Зубов В.С. Справочник программиста. Базовые методы решения графовых задач и сортировки. – М.: "Филинъ", 1999. 256 с.
9. В.В.Белов, Е.М.Воробьев, В.Е.Шаталов. Теория графов. М.: Высш. шк., 1976. 392 с.
10. Оре О. Теория графов. М.: 1980.
11. А.А.Зыков. Основы теории графов.- М.: Наука, 1987. 384 с.
12. Ю.Н.Тюрин, А.А.Макаров. Анализ данных на компьютере. М.: ИНФРА-М, Финансы и стат., 1995. 384 с.
13. Л.И.Турчак. Основы численных методов. М.: Наука, 1987. 320 с.
14. В.Н.Нефедов, В.А.Осипова. Курс дискретной математики: Учеб. пос. М.: Изд-во МАИ, 1992. 264 с.
15. Bloch J. Effective Java: Programming Language Guide. Publisher: Addison Wesley, 2001. – 180 p.

Додаток А

Приклад оформлення титульної сторінки курсової роботи

ЗВО «УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА»

Факультет суспільних і прикладних наук

Кафедра інформаційних технологій

КУРСОВА РОБОТА

з Об’єктно-орієнтованого програмування

на тему:

«_____

_____»

Студента (ки) _____ курсу ___ _____ групи

галузі знань 12 «Інформаційні технології»

спеціальності 121 «Інженерія програмного забезпечення»

_____

(прізвище та ініціали)

Керівник _____к.т.н., Пашкевич О.П.

Національна шкала _____

Кількість балів __ ___ __ Оцінка: ECTS _____

м. Івано-Франківськ

2020

Додаток Б

Бланк завдання на курсовий проект (роботу)

_____

назва вищого навчального закладу

Кафедра _____

Дисципліна _____

Спеціальність _____

Курс_____ Група _____ Семестр ____ _____

ЗАВДАННЯ

на курсовий проект (роботу) студента

_____

Прізвище, ім’я, по-батькові

1. Тема проекту (роботи) _____

_____

_____

2. Строк здачі студентом закінченого проекту (роботи) _____

_____

_____

3. Вихідні дані до проекту (роботи) _____

_____

_____

4. Зміст роботи (перелік питань по розділах, які потрібно розробити) _____

_____

_____

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень) ____

_____

_____

6. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів курсового проекту (роботи)	Строк виконання	Примітки

Студент _____ _____
Підпис Прізвище, ініціали

Керівник _____ _____
Підпис Прізвище, ініціали

«____» _____ 202___ р.

Додаток В

Приклад оформлення змісту курсової роботи

ЗМІСТ

Додаток Г

Приклад оформлення таблиць у курсовій роботі

Таблиця 2.1 – Показники отримані підчас першого експерименту у несприятливих для GPS навігатора умовах

№	довгота	широта	№	довгота	широта
1	24.9218540	48.9218540	12	24.7003985	48.9218805
2	24.7003580	48.9218699	13	24.7003982	48.9218851
3	24.7003644	48.9218615	14	24.7003961	48.9218894
4	24.7003615	48.9218710	15	24.7003970	48.9218939
5	24.7003694	48.9218783	16	24.7004020	48.9218970
6	24.7003807	48.9218781	17	24.7004060	48.9219006

Додаток Д
Приклад оформлення рисунків у курсовій роботі

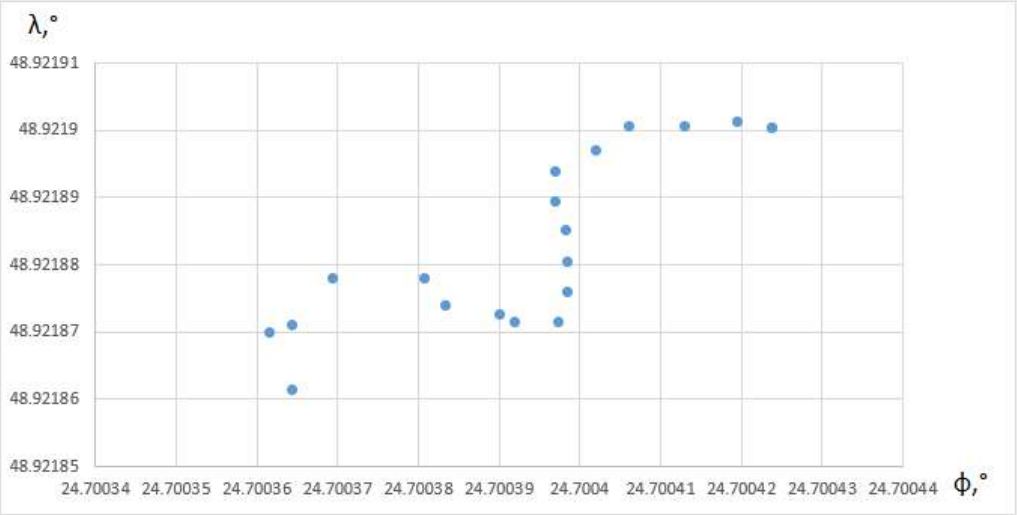

Рисунок 2.7 – Діаграма множин значень отриманих від GPS модуля на одній позиції у несприятливих умовах

Додаток Е
Взірець подання формул у курсовій роботі

Коефіцієнти передавальної функції цифрового фільтру нижніх частот першого порядку повинні відповідати формулі 2.2

$$b_0 = b_1 = (1 + a)/2. \quad (2.2)$$

Щоб забезпечити адекватну роботу фільтра нижніх частот необхідно щоб значення коефіцієнта а знаходилось у межах |a| < 1.

Додаток Ж
Взірець оформлення списку використаних джерел

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Differential GPS [Електронний ресурс] – Режим доступу до ресурсу: https://www.trimble.com/gps_tutorial/dgps-how.aspx.

2. Яценков В. С. Основы спутниковой навигации. Системы GPS NAVSTAR и ГЛОНАСС / В. С. Яценков. – Москва: Горячая Линия - Телеком, 2005. 272 с. ISBN 93517-218-6.

3. Герасименко К. В. Аналіз способів подавлення сигналів супутникових радіонавігаційних систем навмисними перешкодами / К.В. Герасименко // Системи озброєння і військова техніка. 2015. № 4(44). С. 61-63.

4. Анучин О.Н Интегрированные системы ориентации для морских подвижных объектов / О.Н. Анучин, Г.И Емельянцев. Вид. 2-ге, стер. Спб. : ГНЦ РФ-ЦНИИ «Электроприбор», 2003. 390 с. ISBN 5-900780-47-3.

5. Гакаев Рустам Точность и погрешность измерений на картах при выполнении практических работ по топографии. Педагогика высшей школы. 2016. № 1 (4). С. 48-53. ISSN 2410-7352.