ЗВО «УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА»

Факультет суспільних і прикладних наук

Кафедра інформаційних технологій

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ ЩОДО ВИКОНАННЯ КУРСОВОЇ РОБОТИ З НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

«ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ»

Галузь знань : ***12 «Інформаційні технології»***
Спеціальності: ***121 «Інженерія програмного забезпечення»***

Освітній рівень **(перший) бакалаврський**
Освітня програма **Розробка та тестування програмного забезпечення**

Вид дисципліни **базова**
Мова викладання, навчання та оцінювання **українська**

Івано-Франківськ
ЗВО «Університет Короля Данила»
2022/23

Методичні рекомендації щодо виконання курсової роботи з навчальної дисципліни «Об’єктно-орієнтоване програмування» для здобувачів освітнього рівня бакалавр зі спеціальності 121 «Інженерія програмного забезпечення» у ЗВО «Університет Короля Данила». – Івано-Франківськ: ЗВО «Університет Короля Данила», 2022. – 25 с.

ЗАТВЕРДЖЕНО на засіданні кафедри інформаційних технологій ЗВО «Університет Короля Данила» (протокол №2 від 29.09.2022 р.)

Розробник:

к.т.н., доцент кафедри ІТ  Олег ПАШКЕВИЧ

Узгоджено

Гарант ОП  Сергій ВАЩИШАК

ЗМІСТ

Загальні положення 4

Теми курсових робіт 6

Варіанти завдань до курсової роботи 7

Етапи виконання роботи 19

Структура курсової роботи 21

Загальні вимоги до оформлення пояснювальної записки 23

Критерії оцінювання 26

Дотримання академічної доброчесності 27

Література 28

Додатки 29

ЗАГАЛЬНІ ПОЛОЖЕННЯ

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.

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», грунтовному 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.

Вихідним документом для виконання курсової роботи є завдання, яке видається студенту не пізніше ніж через тиждень від початку семестру, в якому виконується курсова робота.

Завдання на курсову роботу видається керівником курсової роботи.

Керівник курсової роботи узгоджує із студентом календарний план роботи на весь період виконання курсової роботи із зазначенням конкретних термінів завершення окремих розділів.

Студент представляє керівнику курсової роботи свої підготовлені рішення з питань курсової роботи. Керівник курсової роботи розглядає подані матеріали і дає зауваження з обсягу і по суті виконаної роботи та виявлених помилок і недопрацювань.

До розробки наступного розділу курсової роботи студент приступає тільки після перевірки і погодження керівником виконаної роботи в попередньому розділі.

За прийняті в курсовій роботі рішення і достовірність усіх даних відповідає студент – автор курсової роботи.

Завершену курсову роботу студент підписує і подає керівнику, який після перевірки і схвалення роботи підписує його з відповідним висновком про допуск студента до захисту курсової роботи.

Професійні компетентності та результати навчання, яких набувають здобувачі внаслідок виконання курсової роботи з навчальної дисципліни «Об'єктно-орієнтоване програмування» (шифри та зміст компетентностей та програмних результатів вказані відповідно до освітньої програми «Розробка та тестування програмного забезпечення», введеної в дію ЗВО «Університет Короля Данила» 01.09.2022 року)

Шифр та назва компетентності	Шифр та назва результату 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. ПРН6. Уміння вибирати та використовувати відповідні задачі методологію створення програмного забезпечення. ПРН 10. Проводити передпроектне обстеження предметної області, системний аналіз об'єкта проектування. 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.	
ЗК5. Здатність вчитися і оволодівати сучасними знаннями.	
ЗК6. Здатність до пошуку, оброблення та аналізу інформації з різних джерел.	
ФК2. Здатність брати участь у проектуванні програмного забезпечення, включаючи проведення моделювання (формальний опис) його структури, поведінки та процесів функціонування.	
ФК3. Здатність розробляти архітектури, модулі та компоненти програмних систем.	
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.	

ТЕМИ КУРСОВИХ РОБІТ

Студент може обрати тему для виконання курсової роботи із запропонованих. Також студент може запропонувати власну тему.

Запропоновані теми курсової роботи:

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».
26. Комп’ютерна модель системи терморегулювання.
27. Алгебра бінарних відношень. Програмна реалізація.

Основна частина курсової роботи має містити виконання наступних завдань:

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

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 - вставити елемент у задану 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: [1-5].

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:
   - size - кількість елементів списку;
   - front - посилання на перший елемент;
   - back - посилання на останній елемент;
   - 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;
   - remove - вилучення всіх елементів із списку з наданим значенням;
   - 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);
   - filter - утворити новий список з елементів, що відповідають заданій умові (пердикату);
   - split - розбиття вихідного списку на два, залежно від умови (предикату);
   - swap - обмін значеннями з іншим списком;
   - equals - поелементне порівняння двох списків;
   - 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].

ЕТАПИ ВИКОНАННЯ

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.

Етап 1.

Основним змістом першого етапу є формулювання призначення програми та поставка цілей, а також виявлення оригінальних задач, які будуть вирішуватися в процесі виконанні 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.

Етап 2.

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

При формуванні концепції проектування здобувач освіти формує оригінальну авторську проектну ідею, а також засоби й методи її втілення. Проектна ідея ґрунтується на результатах допроектного аналізу аналогів й прототипів, детальному вивченні об'єкта проектування і його специфіки.

Часто ті або інші рішення, при побудові моделі предметної області визначаються специфікою завдань, що вирішуються в ній, і базуються на результатах, отриманих при вивченні предметної області. Для кожної з сутностей, які необхідні для вирішення конкретної задачі, наводяться специфікації обмежень цілісності та операційних правил, а саме:

1. Обмеження атрибутів сутностей
2. Обмеження унікальності
3. Динамічні обмеження
4. Інші обмеження
5. Операційні правила

Етап 3.

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.

На цьому етапі напрацьовуються конкретні програмні рішення а також, спільно з викладачем уточнюються можливі варіанти, основні ідеї та можливості їх реалізації. На цьому етапі, необхідно розробити відповідне демонстраційне програмне забезпечення, що дозволяє виконувати типові операції роботи з розробленим об’єктом (згідно варіанту завдання). Окрему увагу слід приділити тестуванню програми.

Етап 4.

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.

У результаті цього аналізу формується повний обсяг курсової роботи, достатній для найбільш оптимального розкриття проектної теми й реалізації оригінальної авторської концепції. Також на цьому етапі здобувач освіти виконує оформлення проектно-графічної частини.

СТРУКТУРА КУРСОВОЇ РОБОТИ

В загальному випадку курсова робота повинна включати:
– пояснювальну записку;
– цифровий носій з функціонуючою програмою.

Пояснювальна записка (далі ПЗ) курсової роботи – це документ, в якому приводяться необхідні описи, структури, таблиці, розрахунки та обґрунтування прийнятих у проекті конструкторських, технологічних, техніко-економічних та інших рішень.

В загальному випадку ПЗ повинна складатися з наступних частин (табл. 1)

Таблиця 1 – Структура ПЗ

№	н	Назва	Обсяг в аркушах
1	–	Титульний аркуш	1
2	–	Завдання на курсову роботу та календарний план	1
3	–	Анотація українською мовою	0,3…0,5
4	–	Анотація іноземною (англ., нім., тощо) мовою	0,3…0,5
5	+	Зміст	1
6	+	Основна частина	10…25
7	+	Висновки	1
8	+	Перелік використаної літератури	1
9	–	Додатки	0…20

(–)/(+) – не ставити / ставити порядковий номер на сторінці

Титульний аркуш

Титульний аркуш оформляти на відповідному бланку (додаток А).

Завдання на курсову роботу та календарний план

Завдання на курсову роботу оформляють на відповідному бланку (додаток Б).

Анотація українською мовою

Анотація призначена для ознайомлення з основним напрямком, ідеями та результатами курсової роботи і повинна містити стислу характеристику виконаної роботи. Після кожної анотації наводять ключові слова. Ключовим словом називають слово або стійке словосполучення із тексту анотації, яке з погляду інформаційного пошуку несе смислове навантаження. Сукупність ключових слів повинна відображати поза контекстом основний зміст курсової роботи.

Анотація іноземною мовою

Анотація іноземною мовою за змістом повинна відповідати українському варіанту (змістовний переклад).

Зміст

Зміст ПЗ оформляють на окремих аркушах. Слово «Зміст» розміщують посередині сторінки з великої літери. У змісті приводять порядкові номери і назви розділів, при необхідності – підрозділів, а також додатків із поданням їх позначення та заголовків із зазначенням номерів сторінок, на яких вони наведені. Зміст включають у загальну кількість аркушів пояснювальної записки.

Зміст **основної частини** ПЗ визначається специфікою курсової роботи і повинен включати розділи вказані в завданні.

У першому розділі дається аналітичний огляд можливостей побудови систем, що вирішують поставлену задачу. Розглядаються особливості об’єктно-орієнтованого підходу і мов, що підтримують ООП. Наводиться обґрунтування обраної мови реалізації.

В другому 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’язки 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

Перелік літературних джерел, на які є посилання в пояснювальній записці подають на окремих аркушах, крім того у відповідних місцях ПЗ повинні бути посилання на подані джерела інформації.

Бібліографічні джерела подають у порядку, за якими вони вперше згадуються в тексті. Порядкові номери у переліку повинні відповідати номерам у тексті пояснювальної записки.

Відомості про джерело інформації необхідно подати у відповідності до Національного стандарту України ДСТУ 8302:2015.

Додатки

Матеріали, що доповнюють курсову роботу. У додатки можна включати: графічний матеріал, таблиці великого формату, описи приладів, алгоритмів, блок-схем, текстів програм, специфікації тощо.

ЗАГАЛЬНІ ВИМОГИ ДО ОФОРМЛЕННЯ ПОЯСНЮВАЛЬНОЇ ЗАПИСКИ

Пояснювальну записку до курсової роботи оформляють у відповідності до вимог, що виносяться для робіт такого типу.

Електронну версію ПЗ подають на випускну кафедру записану на цифрових носіях у форматі pdf.

ПЗ повинна подана на кафедру з усіма заповненими пунктами в паперовому вигляді.

Текст ПЗ набирають у текстовому редакторі відповідно до таких вимог:

1. Формат сторінки А4 (210×297)

– відступи: зліва – 25мм, справа – 15мм, зверху – 20мм, знизу – 20мм;

– нумерація відповідно до основного напису для текстових документів.

2. Основний текст: гарнітура - Times, кегль – 14, абзац – 10мм, шрифт – звичайний, міжрядковий інтервал – 1.5, вирівнювання – по ширині.

3. Назви розділів: гарнітура – Times, кегль – 14, великими літерами, шрифт – напівжирний, вирівнювання – по центру.

4. Рисунки та графіки вставляють у текст ПЗ у одному з растрових форматів (bmp, tif) з роздільною здатністю не менше ніж 300dpi. Прості рисунки допускається виконувати засобами текстового редактора – обов’язково групувати в окремий об’єкт, складні багатокомпонентні рисунки формувати за допомогою програмних комплексів Visio, CorelDraw та інші. Написи на рисунках виконують шрифтом основного тексту, кегль – 12. Рисунки нумерують і підписують, під рисунком шрифтом основного тексту, кегль – 14, вирівнювання - по центру.

Рисунки необхідно подавати в роботі безпосередньо після тексту, де вони згадані вперше, або на наступній сторінці. Безпосередньо сама ілюстрація відокремлюється вільним рядком зверху від основного тексту та знизу від її підпису. Підпис ілюстрації відокремлюється знизу вільним рядком від основного тексту

Підпис рисунка складається зі слова «Рисунок», номера ілюстрації та її назви. Рисунки нумерують послідовно в межах розділу, за винятком рисунків, поданих у додатках. Номер ілюстрації повинен складатися з номера розділу і порядкового номера ілюстрації, між якими ставиться крапка. Між номером рисунка та його назвою ставиться тире. Якщо в роботі подано одну ілюстрацію, то її нумерують за загальними правилами

5. Таблиці подають як окремі об’єкти з розмірами приведеними до сторінки складання. Текст таблиці виконують шрифтом основного тексту, кегль – 12, заголовки колонок: кегль – 12, шрифт – напівжирний, вирівнювання – по центру. Заголовки (назви) таблиць: кегль – 14, шрифт – звичайний, вирівнювання – по центру. Нумерація таблиць: кегль – 14, шрифт – звичайний, вирівнювання – за правим краєм.

Таблиця обов’язково має містити "шапку" з назвою стовпчиків (іноді – рядків). Якщо це зробити неможливо (наприклад, таблиця становить собою матрицю), краще оформити таблицю як ілюстрацію. Шапка таблиці також виділяється жирним шрифтом.

Якщо цифрові або інші дані в якомусь полі таблиці не подають, то в ньому ставлять прочерк. Неприпустимо залишати поля таблиці незаповненими.

Таблицю подають після першого згадування про неї в тексті або, якщо це неможливо, на наступній сторінці. Таблицю розміщують таким чином, щоб її можна було читати без повороту переплетеного блоку роботи або з поворотом за годинниковою стрілкою.

На всі таблиці у пояснювальній записці курсового проекту повинні бути посилання у тексті. У повторних посиланнях на таблиці треба вказувати скорочено слово "дивись", наприклад: (див. табл. 1.3), (див. табл. А.3).

6. Формули подають у вигляді окремих об’єктів (MathType або інших доступних формульних редакторів), вирівнювання – по центру і нумерують в круглих дужках з правого краю. Шрифт – звичайний – 14 пт, великий індекс – 10 пт, маленький індекс – 8 пт, великий символ – 18 пт, маленький символ – 12 пт.

Елементи формули необхідно позначати відповідно до їх функціонального застосування (sin x: sin – функція, x – змінна). Позначення математичних, фізичних та інших величин в тексті та у формулах потрібно записувати курсивом, за винятком стандартних функцій: sin, cos, tg, ctg тощо, чисел (критеріїв) Re, Nu, Gr, Ar, Pr, Eu тощо; rot, div, grad, const тощо, а також позначень буквами грецького алфавіту чи цифр. Індекси в цих величинах записувати прямими буквами українського і грецького алфавітів та цифрами або курсивом – буквами латинського алфавіту. Якщо індекс складається з одного скорочення, то крапку після нього не ставлять, якщо ж з кількох скорочень, то крапку ставлять тільки у проміжних скороченнях, крім останнього. В розмірностях величин як букви, так і цифри записують прямим шрифтом.

7. Посилання. Здобувачі освіти обов’язково повинні робити посилання на джерела, матеріали або окремі результати з яких наводяться в роботі (теоретичні джерела, довідкові матеріали тощо), а також на таблиці, формули, ілюстрації та додатки роботи.

При використанні матеріалів, опублікованих іншими авторами, окреме посилання робиться для кожної наведеної закінченої думки, яка в тексті може бути виділена абзацом, окремим реченням або цитатою.

8. Література: гарнітура – Times, кегль – 14, шрифт – звичайний, вирівнювання – за лівим краєм. Розташування та нумерація в порядку посилань в тексті ПЗ (використання).

9. Додатки. Додатки оформлюють як продовження пояснювальної записки до основної частини курсового проекту безпосередньо після списку використаної літератури у вигляді окремої частини, і розміщуються у порядку появи посилань на них у тексті пояснювальної записки. Додатки повинні починатися з титульного аркуша, на якому великими літерами симетрично аркуша надруковано слово «ДОДАТКИ». Кожний додаток також повинен починатися з титульного аркуша, на якому симетрично до сторінки жирним шрифтом наведено слово «додаток» і відповідна літера.

Наприклад: «ДОДАТОК А». З іншого рядка – його назва малими літерами жирним шрифтом, наприклад: «Організаційна структура управління

підприємства». Слова «ДОДАТКИ», «ДОДАТОК», номер додатка та його назву в лапки не беруть.

Додатки слід позначати послідовно великими літерами української абетки, за винятком літер Г, Ґ, Є, І, Ї, Й, О, Ч, Ь, наприклад: «ДОДАТОК А», «ДОДАТОК Б» тощо.

Ілюстрації, таблиці та формули, які розміщені в додатках, нумерують у межах кожного додатка, наприклад: «Рисунок Д.2» – другий рисунок додатка Д; (А.1) – перша формула додатка А. В іншому на ілюстрації, таблиці та формули, розміщені в додатках, поширюються загальні вимоги щодо оформлення.

Після назви додатка перед текстом додатку (таблицями, рисунками) залишають один вільний рядок.

10. Загальне оформлення ПЗ. Зминання аркушів ПЗ, помарки та інші технічні пошкодження не допускаються. ПЗ курсової роботи повинна мати тверду або м'яку палітурку (з паперу більш щільного ніж аркуші ПЗ). ПЗ курсової роботи необхідно скріпити за допомогою швидкозшивача, тасьми тощо.

На кольорову обкладинку ПЗ потрібно наклеїти етикетку з білого паперу розміром 120х80мм, на якій чорним кольором вказують назву документу, його позначення, шифр групи, ім’я та прізвище студента, рік виконання роботи.

На білу обкладинку ПЗ вище згадані дані наносять безпосередньо в рамці, що відповідає розмірам етикетки (додаток В).

КРИТЕРІЇ ОЦІНЮВАННЯ

Для захисту роботи здобувач освіти повинен підготувати графічний матеріал, що відображає суть виконаної роботи

Розподіл балів наведено в таблиці 2

Таблиця 2 – Розподіл балів за виконання за захист курсової роботи

№	Етапи	Розподіл балів
	Виконання роботи	**60**
1	Аналіз теми, завдання, методичних вказівок і літератури з проектування	5
2	Етап 1	10
3	Етап 2	20
4	Етап 3	20
5	Етап 4	5
	Захист курсової роботи (доповідь)	**40**
1	Володіння культурою презентації (вільне володіння текстом доповіді, наявність в структурі доповіді всіх належних елементів: представлення, обґрунтування, мети, завдань курсової роботи, викладення особисто розроблених теоретичних, практичних, проблемних, рекомендаційних аспектів роботи. Вміння стисло (в межах регламенту), послідовно й чітко викласти сутність і результати.	20
2	Повнота і ґрунтовність відповідей на запитання викладачів, на зауваження і пропозиції, здатність аргументовано захищати свої пропозиції, думки, погляди.	10
3	Наявність графічного матеріалу для захисту курсової роботи (матеріалів, що відображають суть виконаної роботи повинен мати чітке, грамотне без будь-яких помилок оформлення; зв’язок доповіді з кожним листком презентації)	10

Таким чином, якість виконання даного курсової роботи оцінюється в діапазоні від 0 до 60 балів, а результати захисту курсової роботи оцінюються в діапазоні від 0 до 40 балів.

Загальна підсумкова оцінка при захисті курсової роботи складається з суми балів, отриманих за якість виконання курсової роботи, та кількості балів, отриманих при захисті.

До залікової відомості заносяться сумарні результати в балах, отримані при виконанні та при захисті курсової роботи.

Студент, що не виконав курсовий в термін, допрацьовує її самостійно. Консультації в цьому випадку організовуються з урахуванням причин відставання. Рішення про організації консультацій виносить кафедра на підставі

письмової заяви студента. При цьому кафедра визначає кількість годин і форму проведення консультацій, графік проектування і термін його виконання.

У випадку одержання незадовільної оцінки на захисті студент одержує нове завдання на курсову роботу для повторного проходження проектування.

ДОТРИМАННЯ АКАДЕМІЧНОЇ ДОБРОЧЕСНОСТІ

Здобувачі освіти при виконанні курсової роботи повинні дотримуватись принципів академічної доброчесності у відповідності до «Положення про академічну доброчесність у Приватному вищому навчальному закладі Університет Короля Данила» (https://cutt.ly/l2S2Pbv)

В курсовій роботі не повинно бути плагіату, фальсифікації та фабрикування матеріалів роботи.

Здобувач освіти несе відповідальність виявлені факти фабрикування чи фальшування результатів роботи у відповідності до чинного положення

ЛІТЕРАТУРА

1. Bruce Eckel. Thinking in Java: 4th Edition. Pearson, 2006. 1150 p.
2. Cay Horstmann. Core Java Volume 1 – Fundamentals: 11th Edition. Pearson, 2018. 928 p.
3. Cay Horstmann. Core Java Volume 2 – Advanced Features: 11th Edition. Pearson India, 2019. 888 p.
4. Herbert Schildt. Java: The Complete Reference, Eleventh Edition. McGraw Hill, 2018. 1248 p.
5. Harold Abelson. Structure and Interpretation of Computer Programs, 2nd Edition / Gerald Jay Sussman, Julie Sussman. – The MIT Press, 1996. 657 p.
6. Grady Booch. Object-Oriented Analysis and Design with Applications, 3rd Edition / Robert A. Maksimchuk, Michael W. Engle. – Addison-Wesley Professional, 2007. 720 p.
7. Андрій Будай. Дизайн-патерни просто, як двері. [Електронний ресурс]: URL: https://sites.google.com/site/designpatternseasy/
8. Олександр Швець. Занурення в патерни проектування. [Електронний ресурс]: URL: https://refactoring.guru/uk/design-patterns/book
9. Grady Booch. Unified Modeling Language User Guide, The (Addison-Wesley Object Technology Series), 2nd Edition. Addison-Wesley Professional, 2017. – 504 p.
10. Richard Helm. Design Patterns: Elements of reusable object-oriented software / Richard Helm, Erich Gamma, John M. Vlissides, Ralph Johnson. Addison-Wesley Professional, 1994. 416 p.
11. Martin Fowler. UML Distilled: A Brief Guide to the Standard Object Modeling Language, 3rd Edition. Addison-Wesley Professional, 2003. 208 p.
12. Allen Downey. Think Python: How to Think Like a Computer Scientist, 2nd Edition. O'Reilly Media, 2016. 289 p.
13. Scott Tilley. Systems Analysis and Design (MindTap Course List), 12th Edition. Cengage Learning, 2019. 576 p.
14. Stanley Lippman. C++ Primer, 5th Edition / Josée Lajoie, Barbara Moo. Addison-Wesley Professional, 2012. 976 p.
15. Tony Gaddis. Starting out with Visual C#, 5th Edition. Pearson, 2019. 960 p.

ДОДАТКИ

Додаток А

ФОРМА ТИТУЛЬНОГО АРКУША

ЗВО «УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА»

Факультет суспільних і прикладних наук

Кафедра інформаційних технологій

КУРСОВА РОБОТА

з дисципліни «ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ»

_____

(тема)

_____

ПОЯСНЮВАЛЬНА ЗАПИСКА

Студент гр._____

(шифр групи) (підпис) (розшифровка підпису)

Допускається до захисту:

Керівник курсової роботи

_____

(посада) (підпис) (дата) (розшифровка підпису)

м. Івано-Франківськ

2023

Додаток Б

ФОРМА ЗАВДАННЯ НА КУРСОВУ РОБОТУ

ЗВО «УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА»

Факультет суспільних і прикладних наук

Кафедра інформаційних технологій

Дисципліна Об’єктно-орієнтоване програмування

Спеціальність Інженерія програмного забезпечення

Курс _____ Група _____ Семестр_____

ЗАВДАННЯ
НА КУРСОВУ РОБОТУ

Студенту _____

1. Тема роботи _____

_____

_____

2. Термін здачі студентом закінченої роботи_____

3. Зміст пояснювальної записки_____

_____

_____

_____

_____

_____

_____

_____

4.Дата видачі завдання _____

Продовження додатку Б

Форма завдання на курсову роботу (на звороті першого аркуша)

КАЛЕНДАРНИЙ ПЛАН

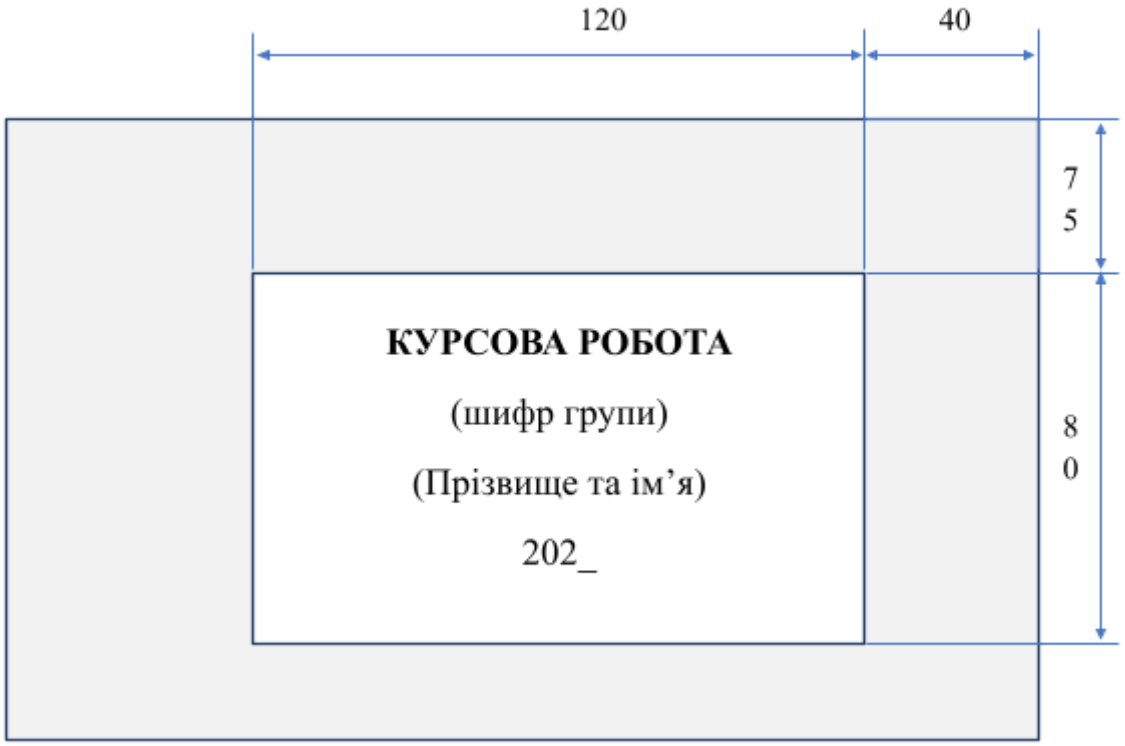
Номер і назва етапів курсової роботи	Термін виконання етапів роботи	Примітка

Студент_____
(підпис) (розшифровка підпису)

Керівник
роботи _____
(підпис) (розшифровка підпису)

Додаток В

Приклад форми обкладинки пояснювальної записки



Примітка: Розміри для довідок