МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДВНЗ «УЖГОРОДСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМ

МЕТОДИЧНІ ВКАЗІВКИ

для виконання лабораторних робіт

з дисципліни

ТЕХНОЛОГІЯ ПРОГРАМУВАННЯ ТА СТВОРЕННЯ ПРОГРАМНИХ ПРОДУКТІВ

для студентів спеціальності

F2 Інженерія програмного забезпечення

Ужгород
2025

УДК 004.4:004.43:378.147
М 54

Технологія програмування та створення програмних продуктів: **методичні вказівки до виконання лабораторних робіт для здобувачів першого (бакалаврського) рівня вищої освіти, спеціальності F2 Інженерія програмного забезпечення факультету інформаційних технологій ДВНЗ «Ужгородський національний університет»** / Укладачі: П.П. Федорка, І.М. Лях, Р.Ю. Бучук. Ужгород: ДВНЗ «УжНУ», 2025. 31 с.

У методичних вказівках до виконання лабораторних робіт з курсу «Технологія програмування та створення програмних продуктів» сформульовано теми та мети завдань до виконання лабораторних робіт, вимоги до порядку виконання та змісту звіту по проробленій роботі. Також наведено теоретичні відомості, завдання до виконання лабораторних робіт та перелік контрольних запитань на підсумковий контроль.

Укладачі:
Федорка Павло Павлович – доктор філософії, доцент кафедри програмного забезпечення систем факультету інформаційних технологій ДВНЗ «Ужгородський національний університет»;
Лях Ігор Михайлович – доцент, доктор технічних наук, професор кафедри інформатики та фізико-математичних дисциплін факультету інформаційних технологій ДВНЗ «Ужгородський національний університет»;
Бучук Роман Юрійович – кандидат фізико-математичних наук, доцент кафедри програмного забезпечення систем факультету інформаційних технологій ДВНЗ «Ужгородський національний університет»

Рецензенти:
Кут Василь Іванович – доцент, кандидат технічних наук, завідувач кафедри інформатики та фізико-математичних дисциплін факультету інформаційних технологій ДВНЗ «Ужгородський національний університет»;
Млавець Юрійович Юрійович – доцент, кандидат фізико-математичних наук, доцент кафедри кібернетики і прикладної математики факультету математики та цифрових технологій ДВНЗ «Ужгородський національний університет»

Методичні рекомендації розглянуто та затверджено
на засіданні кафедри програмного забезпечення систем
факультету інформаційних технологій
(протокол №13 від «12» травня 2025 р.)

Схвалено та рекомендовано до друку
науково-методичною комісією
факультету інформаційних технологій
(протокол №7 від «12» червня 2025 р.)

ЗМІСТ

ВСТУП..... 4
ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ..... 6
САМОСТІЙНА РОБОТА..... 7
ЛАБОРАТОРНА РОБОТА № 1 «ПОПЕРЕДНЄ ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ»..... 8
ЛАБОРАТОРНА РОБОТА № 2 «РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ»..... 12
ЛАБОРАТОРНА РОБОТА № 3 «ПОБУДОВА ФУНКЦІОНАЛЬНОЇ СХЕМИ СИСТЕМИ ПЗ»..... 17
ЛАБОРАТОРНА РОБОТА № 4 «ЗОВНІШНЄ ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ»..... 22
ЛАБОРАТОРНА РОБОТА № 5 «РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ»..... 27
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ..... 31

ВСТУП

Актуальність вивчення дисципліни зумовлена тим, що сучасна індустрія програмного забезпечення стрімко розвивається і вимагає від фахівців високого рівня технічних знань та практичних навичок у розробці програмних продуктів. Вміння ефективно створювати, впроваджувати й підтримувати програмні рішення є ключовим фактором професійної конкурентоспроможності. Теоретичні знання та практичні навички, здобуті під час вивчення дисципліни, є базовими компонентами для подальшої спеціалізації та успішної роботи в командних та індивідуальних ІТ-проєктах.

Метою вивчення навчальної дисципліни «Технологія програмування та створення програмних продуктів» є формування компетентностей щодо розробки, тестування, документування та супроводу програмного забезпечення з урахуванням сучасних технологій програмування, принципів проєктування, вимог до якості та етапів життєвого циклу програмного продукту. Це передбачає опанування методами аналізу вимог, побудови архітектури програмних систем, використання інструментальних засобів та стандартів розробки ПЗ.

Дисципліна сприяє розвитку практичних навичок командної роботи над програмними проєктами, управління версіями, забезпечення якості та впровадження програмних рішень у професійній діяльності; охоплює повний цикл створення програмного забезпечення – від постановки завдання до супроводу готового продукту. Значна увага приділяється практичній реалізації теоретичних знань шляхом виконання лабораторних робіт. Опанування дисципліни дозволяє краще зрозуміти особливості організації роботи над ПЗ в реальному середовищі. Програмне забезпечення, яке створюється в рамках навчального процесу, орієнтоване на вирішення реальних задач і може бути основою для стартапів чи кваліфікаційних робіт бакалавра або магістра. Завдяки системному підходу до викладання курсу студенти навчаються не лише програмувати, а й аналізувати вимоги, проєктувати структуру та архітектури системи та забезпечувати її відповідність стандартам. Вивчення дисципліни також сприяє розвитку критичного мислення, аналітичних здібностей та навичок розв’язання проблем у галузі інженерії програмного забезпечення. Вона формує розуміння

важливості безперервного професійного розвитку у даній сфері, що постійно змінюється.

Відповідно до освітньої програми, вивчення дисципліни сприяє формуванню у здобувачів вищої освіти таких компетентностей:

ІК. Здатність розв’язувати складні спеціалізовані завдання або практичні проблеми інженерії програмного забезпечення, що характеризуються комплексністю та невизначеністю умов, із застосуванням теорій та методів інформаційних технологій.

ЗК 2. Здатність застосовувати знання у практичних ситуаціях.

ФК 1. Здатність ідентифікувати, класифікувати та формулювати вимоги до програмного забезпечення.

ФК 2. Здатність брати участь у проектуванні програмного забезпечення, включаючи проведення моделювання (формальний опис) його структури, поведінки та процесів функціонування.

ФК 3. Здатність розробляти архітектури, модулі та компоненти програмних систем.

ФК 4. Здатність формулювати та забезпечувати вимоги щодо якості програмного забезпечення у відповідності з вимогами замовника, технічним завданням та стандартами.

ФК 5. Здатність дотримуватися специфікацій, стандартів, правил і рекомендацій в професійній галузі при реалізації процесів життєвого циклу

ФК 13. Здатність обґрунтовано обирати та освоювати інструментарій з розробки та супроводження програмного забезпечення.

ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Змістовий модуль 1.

Тема 1. Вступ до предмета, базові поняття.

Тема 2. Види програмного забезпечення.

Тема 3. Технологія розробки програмного забезпечення

Тема 4. Життєвий цикл розробки програмного забезпечення.

Тема 5. Моделі життєвого циклу розробки програмного забезпечення

Тема 6. Стандартизація розробки програмного забезпечення.

Змістовий модуль 2

Тема 7. Сучасні методології розроблення програмних систем.

Тема 8. Якість програмного забезпечення.

Тема 9. Тестування програмного забезпечення.

Тема 10. Захист інформації та безпека програмного забезпечення.

Тема 11. Маркетинг програмних продуктів.

Тема 12. Перспективи розвитку індустрії програмного забезпечення.

САМОСТІЙНА РОБОТА

№ п/п	Назва теми
1.	Вступ до предмета, базові поняття
2.	Види програмного забезпечення
3.	Технологія розробки програмного забезпечення
4.	Життєвий цикл розробки програмного забезпечення
5.	Моделі життєвого циклу розробки програмного забезпечення
6.	Стандартизація розробки програмного забезпечення
7.	Сучасні методології розроблення програмних систем
8.	Якість програмного забезпечення
9.	Тестування програмного забезпечення
10.	Захист інформації та безпека програмного забезпечення
11.	Маркетинг програмних продуктів
12.	Перспективи розвитку індустрії програмного забезпечення

ЛАБОРАТОРНА РОБОТА № 1 (4 години)
«ПОПЕРЕДНЄ ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ»

Тема роботи: складання переліку вимог і функціональних характеристик розроблюваної програми.

Мета роботи: проведення попереднього проєктування конкретної програми; розробка документу «Постановка завдання».

1.1 Порядок виконання роботи і звітність

Під час виконання лабораторної роботи необхідно визначити потребу в програмному виробі, його призначення та основні функціональних характеристик; скласти перелік вимог до нього. Звіт до лабораторної роботи повинен бути оформлений згідно вимог з назвою документа «Постановка завдання».

Постановка завдання: обрати певний вид ПЗ і зробити опис технічного завдання на майбутній проєкт, який планується розробити.

1.2 Теоретичні відомості

Визначення повного комплексу вимог до програмний продукт є первинним завданням його розробки. Неякісне визначення вимог призводить до створення програмного продукту, яке буде правильно вирішувати невірно сформульоване завдання, а програмний продукт не буде відповідати справжнім потребам замовника. Тому при визначенні вимог до програмного виробу потрібно дотримуватися максимально можливу акуратність і точність, щоб потім ці вимоги можна було транслювати в проєкт, що розробляється з мінімальним числом помилок. Вимоги задаються природною (для цієї задачі) мовою і повинні бути чітко сформульовані.

Попереднє проєктування програмного забезпечення є одним із ключових етапів життєвого циклу розробки ПЗ. Воно передбачає глибоке осмислення проблемної області, виявлення потреб користувачів і аналіз існуючих рішень або аналогів. Основна мета цього етапу – закласти чітке розуміння того, що саме потрібно створити, перш ніж перейти до безпосереднього проєктування архітектури та написання коду.

Одним із важливих інструментів попереднього проєктування є системний аналіз, який дозволяє оцінити зв’язки між елементами майбутньої системи, виявити функціональні, нефункціональні та бізнес-вимоги. У процесі аналізу також враховуються зовнішні обмеження, які можуть впливати на вибір технологій або методів реалізації.

Особливу роль відіграє чітка постановка цілей проєкту. Розробник має розуміти як ціль програмного продукту, так і ціль самого проєкту. Це дозволяє уникнути типових помилок, коли програмне забезпечення технічно працює правильно, але не виконує реальних завдань замовника.

Процес попереднього проєктування часто завершується створенням документа «Постановка завдання», який вважається своєрідною угодою між замовником і розробником. У ньому описуються всі аспекти майбутньої програми: її функціональність, вхідні та вихідні дані, обмеження, критерії ефективності, вимоги до безпеки тощо. Якість цього документа значною мірою визначає успішність усієї подальшої розробки.

Ще одним важливим аспектом є трасування вимог – здатність відстежувати зв’язок кожної вимоги з конкретними елементами системи, які її реалізують. Це дозволяє контролювати повноту реалізації та полегшує тестування. Крім того, трасування є основою для майбутньої підтримки і розвитку програмного виробу.

Для уникнення подібних ризиків, вимоги до програмного виробу формалізуються у вигляді окремого документа, який чітко визначає як функціональність, що підлягає реалізації, так і межі відповідальності розробника – тобто те, що не входитиме до складу готового програмного продукту.

1.3 Завдання

Постановка завдання пишеться на природній мові в термінах зрозумілих і користувачеві і розробнику програмного забезпечення і може містити наступні розділи:

1. *Заголовок програми*. Повинна бути вказана повна назва програмного продукту.

2. *Умови завдання*. Формулюється умова завдання, короткий опис програми, що розробляється, її призначення та необхідні уточнення.

3. *Початок/закінчення роботи*. Вказується місяць і рік початку/закінчення розробки програми.

4. *Підстава для розробки* програмного продукту. Підставою для розробки програмного продукту може бути замовлення користувача, завдання адміністрації навчального закладу, контракт навчального закладу з іншою організацією та інше.

5. *Коротка характеристика об'єкта розробки*. Описується об'єкт розробки: як вирішується поставлене завдання в даний час без програми, що розробляється і яка частина буде замінена програмою.

6. *Користувач*. Вказуються користувачі програми.

7. *Мета і призначення розробки*. Створення програмного виробу, що дозволяє автоматизувати сфери застосування. Призначення програми полягає в наданні користувачеві зручного інструменту для виконання щоденних операцій.

8. *Основні вимоги*. Описуються вимоги користувача до розроблюваної програми. Тут же з точки зору користувача слід детально перерахувати функції програми.

9. *Вхідна інформація*. Перераховуються всі вхідні дані програми з точки зору їх змісту і призначення - звіти, файли, записи, поля даних, таблиці ... Їх можливі носії і засоби відображення інформації і т.д.

10. *Вихідна інформація*. Результати обробки вхідних даних у вигляді звітів, таблиць, графіків або повідомлень; службова інформація для користувача: сповіщення про помилки, підказки; підтвердження виконаних дій, можливість експорту результатів у зручні формати для користувача (PDF, CSV тощо).

11. *Вимоги до апаратного та програмного забезпечення*. Описується конфігурація апаратури і програмного забезпечення, в яких розробляється програма може працювати, інші програмні продукти, від яких вона залежить.

12. *Зовнішні обмеження*. Специфіка середовища експлуатації; обмеження на обсяг даних, які може зберігати або обробляти програма одночасно; необхідність дотримання вимог законодавства щодо обробки персональних даних ; залежність від сторонніх бібліотек, форматів або інтерфейсів, що не можуть бути змінені на рівні користувача.

13. *Ефективність*. Цілі продуктивності, такі, як часові і об'ємні характеристики, пропускна здатність, використання ресурсів тощо.

14. *Безпека даних від несанкціонованого доступу*. Контроль доступу на основі ролей (авторизація); шифрування конфіденційної інформації при збереженні та передачі; ведення

журналу дій користувача для виявлення потенційних порушень; можливість резервного копіювання даних.

15. *Ергономічні характеристики*. Ергономічними характеристиками виробу є такі властивості, які забезпечують надійність, комфорт і продуктивність роботи користувачів і операторів. Ергономіка (грец.) - праця + закон - галузь знання, що вивчає трудові процеси з метою створення кращих умов праці.

16. *Мобільність*. Описуються вимоги і цілі забезпечення перенесення програмного продукту з одних робочих умов в інші.

17. *Окупність капіталовкладень*. Визначається прибуток, яку дасть створення програмного продукту в поняттях, відповідних цільовим призначенням організації.

18. *Інші угоди сторін*. Програмний виріб створюється в рамках навчального процесу та не передбачає комерційного використання; усі функціональні зміни, не передбачені початковим технічним завданням, можуть бути внесені лише за погодженням обох сторін; у разі необхідності використання стороннього ПЗ, його джерела повинні бути відкритими або офіційно дозволеними для безкоштовного навчального використання; авторські права на програмний виріб належать розробнику до моменту передачі або публічного розповсюдження за окремою домовленістю.

19. *Термінологія*. Чітко визначається вся термінологія, яка може виявитися специфічної для даної розробки.

1.4 Контрольні запитання

1. Що вивчає навчальна дисципліна «Технологія програмування»? Які ключові поняття вона охоплює?
2. Що таке попереднє проєктування програмного забезпечення? Яку роль у ньому відіграє системний аналіз?
3. Як формулюються цілі проєкту та програмного продукту? Яка між ними різниця?
4. Що таке вимоги до програмного забезпечення? Як їх правильно визначати та класифікувати?
5. Яке призначення документа «Постановка завдання»? Чому його ще називають угодою про вимоги?

ЛАБОРАТОРНА РОБОТА № 2 (4 години)
«РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ»

Тема роботи: визначення етапів життєвого циклу розробки програмного забезпечення та планування процесу створення конкретної програми.

Мета роботи: визначення етапів розробки ПЗ та розробка календарного плану створення програми.

2.1 Порядок виконання роботи і звітність

Під час виконання лабораторної роботи необхідно визначити тип моделі життєвого циклу програмного продукту, обґрунтувати її вибір, описати етапи розробки обраної програми відповідно до цієї моделі, а також представити схематичне зображення життєвого циклу з урахуванням особливостей власного проєкту. Звіт до лабораторної роботи повинен бути оформлений згідно з вимогами та із назвою документа «Календарний план ПЗ».

Постановка завдання: обрати програмне забезпечення та здійснити опис моделі його життєвого циклу з деталізацією етапів розробки, які будуть реалізовані під час реалізації даного ПЗ.

2.2 Теоретичні відомості

Життєвий цикл програмного забезпечення охоплює сукупність етапів, які проходить програмний продукт від моменту зародження ідеї до повного припинення його використання. Правильне розуміння структури життєвого циклу дає змогу не лише планувати й контролювати процес розробки, а й гарантувати відповідність кінцевого результату потребам користувачів.

Першим етапом життєвого циклу виступає системний аналіз, або попереднє проєктування. У його межах здійснюється дослідження проблемної області, визначення основних вимог до програмного виробу та аналіз доцільності створення ПЗ. Здійснюється оцінка розробки ПЗ з точки зору експлуатаційних, економічних і комерційних чинників. Цей

етап закладає основу для наступних дій, формує стратегічне бачення проєкту й дозволяє уникнути критичних помилок на подальших етапах.

Наступним етапом є проєктування програми, яке включає як зовнішній, так і внутрішній рівень опрацювання архітектури ПЗ. На цьому етапі виконуються наступні кроки: функціональна декомпозиція завдання, розробляється структура системи, моделюються її підсистеми, визначаються компоненти, інтерфейси та структура бази даних. Програмна архітектура повинна враховувати принципи масштабованості, повторного використання та підтримуваності.

У процесі конструювання програми виконується внутрішнє проєктування форм, модулів і об'єктів. Після цього здійснюється програмування, що включає кодування, налагодження окремих компонентів і їх компонування у функціональну систему. Важливою частиною цього етапу є перевірка правильності взаємодії між окремими модулями та дотримання вимог, сформульованих раніше.

Після реалізації основного функціоналу виконується налагодження програми в цілому, що передбачає тестування на відповідність технічним і функціональним вимогам. У цьому контексті особливу роль відіграє оцінка або випробування програмного забезпечення, які дозволяють виявити потенційні помилки, неточності або суперечності.

Завершальним етапом є експлуатація програмного забезпечення. Цей етап включає такі кроки: впровадження ПЗ в робоче середовище, його обслуговування, оновлення, а також поступове виведення з експлуатації. На цьому етапі важливо забезпечити підтримку користувачів і за необхідності – міграцію на нові рішення.

Узагальнена модель життєвого циклу (ЖЦ) допомагає формалізувати процес розробки, забезпечує прозорість і контрольованість. Її використання дозволяє уникнути невиправданих витрат, підвищити якість кінцевого продукту та мінімізувати ризики на всіх стадіях життєвого циклу.

Таким чином, розробка програмного забезпечення не є лише технічним процесом кодування, а структурованим набором послідовних етапів, які охоплюють аналіз, проєктування, реалізацію, тестування та експлуатацію. Знання структури ЖЦ ПЗ є обов'язковим для кожного розробника, оскільки саме воно формує основу ефективної і керованої стадії життєвого циклу.

2.3 Завдання

Постановка завдання пишеться на природній мові в термінах, зрозумілих і користувачеві, і розробнику програмного забезпечення, і може містити наступні розділи:

1. *Заголовок програми*. Програмне забезпечення має назву «Система управління *» (доповнюється індивідуально за вибором студента).

2. *Умови завдання*. Розробляється ПЗ, яке забезпечує можливість додавання, редагування, перегляду та завантаження даних для програмного забезпечення, яке розробляється.

3. *Початок/закінчення роботи*. Розробка програмного забезпечення розпочинається у ... та завершується у ... (вказується дата початку та завершення розробки ПЗ).

4. *Підстава для розробки програмного продукту*. Підставою для створення програмного забезпечення є завдання викладача в межах навчального курсу «Технологія програмування та створення програмних продуктів», з метою оволодіння навичками застосування моделі життєвого циклу ПЗ на практиці.

5. *Коротка характеристика об'єкта розробки*. На поточному етапі поставлене завдання виконується вручну або за допомогою розрізнених інструментів, що ускладнює керування процесом; програмне забезпечення автоматизує ці дії та забезпечує їх централізоване виконання.

6. *Користувач*. Користувачем є особа, яка має потребу в автоматизації конкретного процесу чи задачі, що реалізується за допомогою розроблюваного програмного забезпечення.

7. *Мета і призначення розробки*. Створення програмного виробу, що дозволяє автоматизувати сфери застосування. Призначення програми полягає в наданні користувачеві зручного інструменту для виконання щоденних операцій.

8. *Основні вимоги*. Програмне забезпечення повинно мати зручний інтерфейс, забезпечувати виконання ключових функцій, пов'язаних з обробкою даних, взаємодією з користувачем і підтримкою специфіки обраної предметної області.

9. *Вхідна інформація*. Вхідними даними є інформація, що вводиться користувачем або надходить із зовнішніх джерел, необхідна для виконання функцій

програмного забезпечення. Дані можуть вводитися вручну або імпортуватися через інтерфейс завантаження.

10. *Вихідна інформація*. Вихідною інформацією є результати обробки даних, які відображаються або зберігаються у зручному для користувача форматі.

11. *Вимоги до апаратного та програмного забезпечення*. Програма повинна працювати на операційних системах Windows, Linux та macOS та підтримка сучасних браузерів обов'язкова.

12. *Зовнішні обмеження*. Програма має відповідати вимогам законодавства щодо захисту персональних даних, не перевищувати допустиме навантаження на систему зберігання даних, а також працювати в умовах обмеженого Інтернет-з'єднання.

13. *Ефективність*. Програма повинна обробляти запити не довше ніж за 2 секунди, підтримувати до 50 одночасних сесій без втрати продуктивності, ефективно використовувати доступні ресурси системи.

14. *Безпека даних від несанкціонованого доступу*. Забезпечується авторизація за ролями, шифрування конфіденційних даних, журналювання активності користувачів, а також можливість резервного копіювання збереженої інформації.

15. *Ергономічні характеристики*. Інтерфейс повинен бути адаптований під користувачів з різним рівнем підготовки, використовувати контрастну кольорову гаму, підтримувати адаптивну верстку та мінімалізм у візуальних елементах.

16. *Мобільність*. Програма має працювати як у десктопному, так і у мобільному форматі без втрати основного функціоналу.

17. *Окупність капіталовкладень*. Окупність ПЗ може оцінюватися через підвищення ефективності процесів, які автоматизує програмне забезпечення, та зменшення витрат на виконання рутинних завдань.

18. *Інші угоди сторін*. ПЗ створюється в рамках навчального процесу та не передбачає комерційного використання; усі функціональні зміни, не передбачені початковим технічним завданням, можуть бути внесені лише за погодженням обох сторін; у разі необхідності використання стороннього ПЗ, його джерела повинні бути відкритими або офіційно дозволеними для безкоштовного навчального використання; авторські права на програмний виріб належать розробнику до моменту передачі або публічного розповсюдження за окремою домовленістю.

19. *Термінологія*. Терміни, що використовуються в ПЗ (наприклад, «користувач», «інтерфейс», «дані», «результат»), тлумачаться у межах контексту розроблюваного програмного забезпечення відповідно до його функціонального призначення.

2.4 Контрольні питання

1. Що таке технологія розробки програмного забезпечення? Які етапи вона охоплює та яку роль відіграє в створенні якісного програмного продукту?
2. Які вимоги висуваються до ідеальної технології розробки ПЗ? Чому ці вимоги вважаються критично важливими в контексті ефективності й надійності ПЗ?
3. У чому полягають особливості об’єктно-орієнтованого підходу до розробки програмного забезпечення? Які переваги надає цей підхід?
4. Що означає розглядати програмне забезпечення як продукт (виріб)? Які наслідки це має для процесу проєктування, тестування та супроводу?
5. Які основні етапи життєвого циклу програмного забезпечення виділяються під час проєктування? У чому полягає їхня взаємозалежність?
6. Яким чином правильне визначення вимог до програмного забезпечення впливає на подальші етапи розробки та якість кінцевого продукту?
7. Які існують моделі процесу розробки програмного забезпечення (наприклад, каскадна, спіральна, Agile) та у чому полягають їхні переваги й недоліки?
8. Чому документація є важливою складовою розробки програмного забезпечення? Які типи документації використовуються?
9. Які чинники впливають на вибір мови програмування та середовища розробки під час реалізації програмного продукту?
10. Яким чином реалізується підтримка, супровід і модернізація програмного забезпечення після його впровадження в експлуатацію?

ЛАБОРАТОРНА РОБОТА № 3 (4 години)
«ПОБУДОВА ФУНКЦІОНАЛЬНОЇ СХЕМИ СИСТЕМИ ПЗ»

Тема роботи: побудова функціональної схеми системи програмного забезпечення.

Мета роботи: проведення функціональної декомпозиції задачі, що вирішується та побудова функціональної схеми розроблюваної системи ПЗ.

3.1 Порядок виконання роботи і звітність

Під час виконання лабораторної роботи необхідно здійснити функціональну декомпозицію задачі, що вирішується у програмному забезпеченні. Для цього слід визначити основні функції системи, встановити взаємозв'язки між ними та згрупувати їх у відповідні рівні абстракції. На основі отриманих даних будується функціональна схема системи ПЗ. Звіт до лабораторної роботи повинен містити специфікацію, що включає ієрархічну функціональну структуру майбутньої програми, а також графічну функціональну схему, побудовану за обраною методикою декомпозиції.

Постановка завдання: обрати приклад програмного продукту, провести функціональну декомпозицію задачі, визначити основні функції та їх взаємозв'язки. Побудувати функціональну схему системи, яка відображає структуру і взаємодію елементів, що реалізують задану функціональність.

3.2 Теоретичні відомості

Побудова програмного забезпечення, що відповідає вимогам користувача, починається із чіткого розуміння функціональності системи, яка розробляється. Одним із найбільш ефективних методів опису функціональної структури майбутньої програми є функціональна декомпозиція – процес розбиття загальної задачі на підзадачі, що є простішими для реалізації, тестування та подальшої підтримки.

Функціональна декомпозиція відображає підхід до вирішення складної проблеми через її поетапне спрощення. Кожна функція на вищому рівні деталізації розбивається на підфункції, які реалізують її частинами, аж до того рівня, коли функціональні блоки стають достатньо простими для прямого проєктування і програмування. Цей процес

дозволяє структурувати логіку системи, встановити межі відповідальності окремих модулів і забезпечити гнучкість у розподілі завдань між командами розробників.

Функціональна декомпозиція тісно пов’язана з побудовою ієрархічної функціональної схеми системи ПЗ. Така схема графічно відображає залежності між основною функцією та підфункціями, дає змогу побачити загальну картину функціональної архітектури ПЗ. У ній чітко простежується послідовність виконання функцій, логічна підпорядкованість і зв’язки між ними.

Побудова функціональної схеми має підпорядковуватись певним принципам. По-перше, кожен елемент ієрархії повинен бути завершеним на своєму рівні деталізації. По-друге, розподіл задачі на підзадачі має бути повним, щоб жоден функціональний компонент не був втрачений. Також важливо зберігати логічну цілісність – кожна підфункція повинна бути логічним продовженням батьківської функції, а не автономним елементом.

Побудова функціональної схеми полегшує як планування, так і управління системою ПЗ. Завдяки цьому підходу можна виділити критичні функції, оцінити трудомісткість їх реалізації, встановити пріоритети та визначити послідовність виконання робіт. Це особливо корисно при створенні календарних або ресурсних планів у подальших етапах розробки.

Крім того, ієрархічна декомпозиція функцій сприяє зменшенню кількості помилок, оскільки дозволяє розглядати кожен функціональний блок окремо, приділяючи увагу його коректності ще до початку реалізації. Це також спрощує тестування: кожну підфункцію можна перевірити незалежно, таким чином забезпечуючи якісну верифікацію програми.

У сучасній розробці програмного забезпечення функціональні схеми часто створюються за допомогою спеціалізованих інструментів моделювання, таких як UML-діаграми або діаграми потоків даних (DFD). Вони дозволяють формалізувати опис системи та підготувати її до подальшого етапу проєктування – структурного або об'єктно-орієнтованого.

Отже, функціональна декомпозиція є ключовим етапом формування архітектури майбутнього програмного забезпечення. Вона створює міцний логічний фундамент, на

якому базується уся система ПЗ, та служить містком між аналітичним етапом і безпосередньою реалізацією програмного забезпечення.

3.3 Завдання

Постановка завдання пишеться на природній мові в термінах зрозумілих і користувачеві і розробнику програмного забезпечення і може містити наступні розділи:

1. *Заголовок програми*. Автоматизована система управління процесами програмного забезпечення.
2. *Умови завдання*. Розробити програмний продукт, який реалізує ключові функції управління інформаційними потоками та взаємодією компонентів у рамках заданої предметної області.
3. *Початок/закінчення роботи*. Розробка програмного забезпечення розпочинається у ... та завершується у ... (вказується дата початку та завершення розробки ПЗ).
4. *Підстава для розробки програмного продукту*. Необхідність автоматизації бізнес-процесів чи інформаційних систем у відповідній сфері діяльності.
5. *Коротка характеристика об'єкта розробки*. На теперішній час більшість операцій виконується вручну або за допомогою декількох розрізнених електронних таблиць. Програмне забезпечення повинне об'єднати всі функції в єдину централізовану систему.
6. *Користувач*. Особи, які задіяні у роботі з розробки ПЗ, що потребують автоматизації діяльності.
7. *Мета і призначення розробки*. Створити програмне забезпечення для ефективного управління даними та процесами у межах визначеної предметної області.
8. *Основні вимоги*. Забезпечення інтуїтивного інтерфейсу, підтримка основних функціональних можливостей, надійність та масштабованість системи.
9. *Вхідна інформація*. Дані, що надходять з різних джерел відповідно до функціональних вимог системи ПЗ.
10. *Вихідна інформація*. Результати обробки даних у вигляді звітів, статусів, повідомлень тощо.

11. *Вимоги до апаратного та програмного забезпечення*. Програма повинна працювати на операційних системах Windows, Linux та macOS, необхідна наявність сучасного браузера; серверна частина – сумісність з PostgreSQL або аналогічною СУБД.

12. *Зовнішні обмеження*. Необхідність забезпечення конфіденційності персональних даних згідно із законодавством, врахування нормативних, технічних або організаційних обмежень.

13. *Ефективність*. Максимальний час відгуку інтерфейсу – до 2 секунд; забезпечення одночасної роботи щонайменше 30 користувачів без зниження продуктивності.

14. *Безпека даних від несанкціонованого доступу*. Механізм авторизації з поділом ролей; журналювання дій; автоматичне блокування після кількох невдалих спроб входу; шифрування даних.

15. *Ергономічні характеристики*. Інтерфейс із підтримкою адаптивного дизайну; інтуїтивна навігація; темна/світла тема; мінімізація кількості дій для виконання типових задач.

16. *Мобільність*. Можливість доступу до системи з різних пристроїв та платформ.

17. *Окупність капіталовкладень*. Зниження витрат за рахунок автоматизації та підвищення ефективності.

18. *Інші угоди сторін*. ПЗ створюється в рамках навчального процесу та не передбачає комерційного використання; усі функціональні зміни, не передбачені початковим технічним завданням, можуть бути внесені лише за погодженням обох сторін; у разі необхідності використання стороннього ПЗ, його джерела повинні бути відкритими або офіційно дозволеними для безкоштовного навчального використання; авторські права на програмний виріб належать розробнику до моменту передачі або публічного розповсюдження за окремою домовленістю.

19. *Термінологія*. Основні терміни, які використовуються у контексті системи ПЗ.

3.4 Контрольні запитання

1. Що таке функціональна декомпозиція системи? Яку роль вона відіграє в проєктуванні програмного забезпечення?
2. У чому полягає відмінність між висхідним і нисхідним підходами до проєктування ПЗ?
3. Які методи проєктування програмного забезпечення найчастіше використовуються в сучасній практиці?
4. Як відбувається побудова функціональної схеми системи? Які основні етапи цього процесу?
5. Яким чином ієрархічна структура функціональної декомпозиції впливає на надійність та підтримуваність програмного продукту?
6. Що таке модуль в контексті функціональної схеми, і як визначити його межі?
7. Як пов’язана об’єктно-орієнтована модель із функціональною декомпозицією? Чи можуть вони бути використані разом?
8. Які типові помилки виникають під час побудови функціональної схеми ПЗ і як їх уникнути?
9. Які критерії вибору рівнів деталізації при побудові функціональної схеми системи?
10. Як функціональна схема сприяє оптимізації процесу розробки програмного забезпечення?

ЛАБОРАТОРНА РОБОТА № 4 (4 години)
«ЗОВНІШНЄ ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ»

Тема роботи: побудова сценаріїв взаємодії користувача з ПЗ, проєктування інтерфейсу, створення зовнішньої специфікації.

Мета роботи: проведення зовнішнього проєктування конкретної програми; розробка взаємодії користувача з програмою: сценарій, екранні форми, набір підказок, повідомлення тощо.

4.1 Порядок виконання роботи і звітність

Під час виконання лабораторної роботи необхідно описати зовнішню поведінку розроблюваного ПЗ з точки зору користувача. Це передбачає створення сценаріїв взаємодії, розробку зовнішнього вигляду інтерфейсів, побудову послідовностей дій користувача та формування повідомлень системи. Звіт до лабораторної роботи повинен бути оформлений у вигляді зовнішньої специфікації.

Постановка завдання: обрати певний вид ПЗ і здійснити його зовнішнє проєктування шляхом побудови сценаріїв взаємодії з користувачем, опису екранних форм і підказок, а також інших зовнішніх елементів, що впливають на зручність та ефективність роботи з ПЗ.

4.2 Теоретичні відомості

Зовнішнє проєктування програмного забезпечення – це процес, що зосереджений не на технічній реалізації, а на взаємодії користувача із системою. Його основною метою є створення зручного, зрозумілого та ефективного інтерфейсу, який забезпечує комфортне використання майбутнього ПЗ. На цьому етапі враховуються особливості поведінки користувачів, їх рівень підготовки, типові сценарії взаємодії, а також загальні принципи користувацького досвіду.

Результатом зовнішнього проєктування є зовнішня специфікація – документ, у якому описується очікувана поведінка програми з погляду користувача. Вона включає структуру діалогів, організацію меню, екранні форми, повідомлення про помилки,

підказки, функціональні клавіші, вхідні й вихідні форми даних, а також інші компоненти зовнішньої взаємодії.

Одним із ключових елементів зовнішньої специфікації є сценарії користувача – опис типових дій, які виконує користувач у системі. Це допомагає виявити потенційні помилки, визначити логіку переходів між етапами, а також адаптувати інтерфейс під очікування й потреби цільової аудиторії. Кожен сценарій супроводжується відповідними формами виводу даних та можливими повідомленнями.

Правильна організація діалогу з користувачем потребує врахування кількох правил. По-перше, інформація повинна бути представлена у зручному вигляді: зрозумілі мітки, коментарі до чисел, уникаючи “сирих” даних. По-друге, інтерфейс має бути послідовним: однакові формати, скорочення, ієрархія меню сприяють кращому засвоєнню та навігації.

Також варто уникати перевантаження користувача ручним введенням даних – краще надавати можливість вибору із запропонованих опцій. Це знижує кількість помилок, пришвидшує роботу та підвищує зручність. Підказки, контекстна допомога й елементи довідки є обов’язковими складовими сучасного ПЗ, особливо для нових або непрофесійних користувачів.

Важливим аспектом є емоційний комфорт користувача. Інтерфейс не повинен викликати негативні враження до ПЗ. Тексти повідомлень повинні бути дружніми, без жаргону чи технічного сленгу. При цьому не слід залишати користувача в невизначеності – система має чітко повідомляти про успішність дій або помилки.

Слідкувати також треба за естетикою: оформлення екранних форм має бути візуально приємним, логічно структурованим і відповідати загальному стилю ПЗ. Зовнішній вигляд впливає на враження користувача не менше, ніж функціональність, а іноді навіть сильніше.

Програмне забезпечення має бути спроєктоване так, щоб користувач міг у будь-який момент завершити роботу або повернутись на попередній етап. При цьому дані не повинні губитися, а всі відкриті ресурси – коректно закриватися. Така поведінка свідчить про надійність та стабільність ПЗ.

Окрему увагу варто приділити обробці помилок: система має миттєво реагувати на некоректні дії користувача, не допускаючи системної помилки або неконтрольованих

станів. При цьому не варто намагатися автоматично виправляти введення – краще надати чітке пояснення помилки й запропонувати варіанти виправлення.

4.3 Завдання

Постановка завдання пишеться на природній мові в термінах зрозумілих і користувачеві, і розробнику програмного забезпечення і може містити наступні розділи:

1. *Заголовок програми*. Повинна бути вказана повна назва програмного продукту – наприклад, «Інтерактивна система зовнішнього проєктування програмного забезпечення».

2. *Умови завдання*. Розробити зовнішню специфікацію програмного продукту, що включає опис діалогів із користувачем, структуру меню, екранні форми, повідомлення про помилки, підказки та інші зовнішні взаємодії. Програма повинна бути орієнтована на користувача з різним рівнем підготовки і забезпечувати комфортний інтерфейс.

3. *Початок/закінчення роботи*. Розробка програмного продукту планується розпочати у ..., завершити – у ... (потрібно вказати початок та завершення).

4. *Підстава для розробки програмного продукту*. Розробка виконується в межах освітнього процесу як навчальний проєкт із дисципліни «Технологія програмування та створення програмних продуктів», що передбачає практичне застосування принципів зовнішнього проєктування програмного забезпечення.

5. *Коротка характеристика об'єкта розробки*. Об'єктом розробки є програмне забезпечення, що забезпечує зовнішнє проєктування ПЗ – описує зовнішню поведінку і взаємодію з користувачем. На даний час такі описи часто ведуться вручну, що ускладнює їхні узгодження та перевірку. Програма автоматизує створення зовнішньої специфікації та уніфікує формат документації.

6. *Користувач*. Основними користувачами програми є розробники програмного забезпечення, тестувальники, а також викладачі та студенти, які вивчають процес проєктування ПЗ.

7. *Мета і призначення розробки*. Метою є створення програмного забезпечення, який полегшує і стандартизує процес зовнішнього проєктування ПЗ, роблячи його більш зрозумілим і доступним для користувачів із різним рівнем знань.

8. *Основні вимоги*. Програма повинна надавати можливість опису діалогів з користувачем, структуру меню, екранні форми, повідомлення та підказки, а також забезпечувати зручний інтерфейс вводу та виводу інформації. Необхідна підтримка перевірки коректності специфікації до початку розробки ПЗ.

9. *Вхідна інформація*. Вхідними даними є вимоги користувача, сценарії використання, приклади діалогів і макети екранних форм, які можуть надходити у вигляді текстових файлів або описів у форматі, що приймається програмою.

10. *Вихідна інформація*. Вихідними даними є зовнішня специфікація програми у структурованому, зрозумілому форматі, яка містить опис діалогів, меню, екранних форм, повідомлень і допоміжних підказок.

11. *Вимоги до апаратного та програмного забезпечення*. Програма повинна працювати на стандартних комп'ютерах із операційними системами Windows, Linux або macOS, не вимагає спеціального обладнання, і може бути реалізована у вигляді настільного додатку або вебсервісу.

12. *Зовнішні обмеження*. Програма повинна відповідати вимогам доступності для користувачів із різним рівнем підготовки; дотримуватися стандартів оформлення документації; забезпечувати відповідність законодавству щодо обробки персональних даних, якщо такі використовуються.

13. *Ефективність*. Програма має швидко обробляти введені дані, забезпечувати інтуїтивно зрозумілий інтерфейс та мінімізувати кількість помилок користувача.

14. *Безпека даних від несанкціонованого доступу*. Впроваджувати авторизацію користувачів із ролями, зберігати конфіденційні дані у зашифрованому вигляді, вести журнал дій для відстеження змін у специфікації.

15. *Ергономічні характеристики*. Інтерфейс має бути інтуїтивно зрозумілим, мати логічну структуру, кольорову схему, що не втомлює очі, а також швидку навігацію між основними частинами програми.

16. *Мобільність*. За можливості, програма має бути доступною для запуску на різних платформах і пристроях, забезпечуючи однаковий користувацький досвід.

17. *Окупність капіталовкладень*. Використання програми підвищить якість навчального процесу і знизить час на підготовку зовнішньої специфікації, що призведе до економії ресурсів навчального закладу.

18. *Інші угоди сторін*. ПЗ створюється в рамках навчального процесу та не передбачає комерційного використання; усі функціональні зміни, не передбачені початковим технічним завданням, можуть бути внесені лише за погодженням обох сторін; у разі необхідності використання стороннього ПЗ, його джерела повинні бути відкритими або офіційно дозволеними для безкоштовного навчального використання; авторські права на програмний виріб належать розробнику до моменту передачі або публічного розповсюдження за окремою домовленістю.

19. *Термінологія*. Використовуються терміни «зовнішнє проєктування», «зовнішня специфікація», «діалог з користувачем», «екранна форма», «підказка», «сценарій використання», які чітко визначені у відповідних розділах документації.

4.4 Контрольні запитання

1. Що таке зовнішнє проєктування програмного забезпечення і яка його роль у життєвому циклі розробки?
2. Які основні компоненти входять до зовнішньої специфікації програмного продукту?
3. Які правила слід дотримуватися при розробці інтерфейсу користувача, щоб уникнути помилок і непорозумінь?
4. Як зовнішнє проєктування впливає на безпеку і ергономіку програмного продукту?
5. Які типові помилки допускають розробники при створенні зовнішніх взаємодій і як їх уникнути?

ЛАБОРАТОРНА РОБОТА № 5 (4 години)
«РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ»

Тема роботи: розробка архітектури програмного продукту.

Мета роботи: розробити архітектуру програмного забезпечення, спроєктувати структуру всіх його компонент, забезпечити об’єктно-модульно-ієрархічну побудову.

5.1 Порядок виконання роботи і звітність

Під час виконання лабораторної роботи необхідно розробити архітектуру програмного забезпечення, спроєктувати структуру всіх його компонент, забезпечити об'єктно-модульно-ієрархічну побудову. Важливо продумати логічну взаємодію між модулями і описати їхні функції. Звіт до лабораторної роботи повинен бути оформлений у вигляді специфікації, що містить детальний опис архітектури розроблюваної програми.

Постановка завдання: обрати певний вид програмного забезпечення і зробити опис технічного завдання на майбутній проєкт, який планується розробити, з акцентом на розробку його архітектури.

5.2 Теоретичні відомості

Розробка архітектури програмного забезпечення є ключовим етапом у створенні будь-якого ПЗ, адже саме архітектура визначає структуру системи, її компоненти та взаємодію між ними. Типова архітектура програми поділяється на рівні, такі як: центральний диспетчер, місцеві диспетчери, функціональні програми та стандартні бібліотеки. Ця ієрархія дозволяє розробляти модульні і масштабовані системи .

Кожен програмний модуль має бути автономним, функціонально завершеним і незалежним від інших. Висока міцність модулів досягається через посилення внутрішніх зв’язків всередині модуля та ослаблення зв’язків між різними модулями – так званий принцип “слабкого зчеплення”. Такий підхід спрощує підтримку, тестування і подальший розвиток ПЗ.

Організація модулів і їх зв’язків повинна бути ієрархічною. Кожен модуль містить оптимальну кількість коду, достатню для виконання своїх функцій, що сприяє зрозумілості та

легкості в роботі з ним. Модулі, як правило, не повинні зберігати стан між викликами, тобто бути передбачуваними і незалежними від історії використання.

Структура управління у системі передбачає, що модулі нижчого рівня можуть викликатися лише з модулів вищого рівня, а взаємодія між модулями одного рівня не дозволена. Передача управління здійснюється через початок модуля, а повернення – після його завершення. Такий порядок забезпечує чіткий і логічний контроль виконання.

Інформаційні зв’язки між модулями реалізуються через локальні та глобальні змінні, тому доступ до глобальних змінних обмежений відповідними зонами дії, що мінімізує небажані побічні ефекти. Локальні змінні залишаються у межах свого модуля, що підвищує безпеку і передбачуваність.

Кожен модуль повинен мати можливість підключення налагоджувальних засобів, які допомагають розробникам контролювати роботу ПЗ, знаходити і усувати помилки. Оператори, що реалізують ці засоби, зазвичай розташовуються в кінці модуля для зручності.

Процедури, що є частиною модуля, повинні обмінюватися даними через чітко визначені параметри. Функції, які повертають одне значення, є більш зручними для використання в обчисленнях, ніж процедури, що не мають явного результату. Це сприяє гнучкості та читабельності коду.

Врахування і дотримання цих правил при розробці архітектури програмного забезпечення дозволяє створювати надійні, масштабовані та зручні для підтримки системи, які ефективно відповідають вимогам користувачів і замовників.

5.3 Завдання

Постановка завдання пишеться на природній мові в термінах зрозумілих і користувачеві, і розробнику програмного забезпечення і може містити наступні розділи:

1. *Заголовок програми*. Розробка архітектури програмного забезпечення для обраного програмного забезпечення.
2. *Умови завдання*. Розробити структурну архітектуру програмного забезпечення, що включає опис компонентів, їх взаємодії, ієрархію та принципи зв’язку модулів. Архітектура повинна забезпечувати функціональну завершеність, автономність і незалежність модулів, а також сприяти зручності підтримки та подальшого розвитку ПЗ.

3. *Початок/закінчення роботи*. Розробка програмного продукту планується розпочати у ..., завершити – у ... (потрібно вказати початок та завершення).

4. *Підстава для розробки програмного продукту*. Розробка виконується в межах освітнього процесу як навчальний проєкт із дисципліни «Технологія програмування та створення програмних продуктів» для формування компетенцій студентів у сфері розробки архітектури ПЗ.

5. *Коротка характеристика об'єкта розробки*. Програмне забезпечення призначене для автоматизації певної сфери діяльності. На даному етапі потрібно створити архітектуру, яка замінить частину або весь поточний спосіб організації функцій у програмі.

6. *Користувач*. Користувачами є розробники, тестувальники та кінцеві користувачі програмного забезпечення, які будуть взаємодіяти із системою.

7. *Мета і призначення розробки*. Мета – створення архітектури, яка забезпечить модульність, гнучкість і масштабованість програмного забезпечення, полегшить розробку і підтримку, а також підвищить надійність і ефективність системи.

8. *Основні вимоги*. Архітектура повинна включати опис рівнів системи (центральний диспетчер, місцеві диспетчери, функціональні модулі, стандартні бібліотеки), правила організації зв'язків між модулями за управлінням та інформацією, а також механізми забезпечення автономності модулів.

9. *Вхідна інформація*. Вхідними даними є технічне завдання, вимоги користувача, існуючі документи про функціональність та структуру програмного забезпечення.

10. *Вихідна інформація*. Результатом є документ зі специфікацією архітектури ПЗ, що містить опис модулів, їх взаємодії, схеми зв'язків і рекомендації по реалізації.

11. *Вимоги до апаратного та програмного забезпечення*. Розробка виконується на персональних комп'ютерах зі стандартним ПЗ для документування і моделювання .

12. *Зовнішні обмеження*. Архітектура повинна відповідати стандартам програмування, бути сумісною з обраними технологіями і забезпечувати інтеграцію з існуючими системами.

13. *Ефективність*. Архітектура має сприяти оптимальному використанню ресурсів, забезпечувати швидкий доступ до даних і ефективну обробку операцій.

14. *Безпека даних від несанкціонованого доступу*. Архітектура повинна враховувати механізми контролю доступу, захисту даних і забезпечення цілісності інформації.

15. *Ергономічні характеристики*. Описати зручність підтримки і розширення системи, а також зрозумілість структури для розробників.

16. *Мобільність*. Забезпечити можливість адаптації архітектури до різних платформ і умов експлуатації.

17. *Окупність капіталовкладень*. Оцінити вигоди від впровадження модульної архітектури у вигляді зменшення часу розробки і витрат на підтримку.

18. *Інші угоди сторін*. Усі зміни в архітектурі повинні погоджуватися між розробниками та замовником, а права на архітектурний документ належать навчальному закладу до моменту передачі.

19. *Термінологія*. Визначити всі спеціальні терміни, які використовуються в документі для уникнення неоднозначностей.

5.4 Контрольні запитання

1. У чому полягає суть конструювання архітектури програмного забезпечення та які основні підходи використовуються?

2. Які рівні входять до типової архітектури програмного забезпечення, та яку функцію виконує кожен із них?

3. Які основні правила структурної побудови програмних модулів та як вони впливають на підтримуваність і масштабованість системи?

4. Які принципи реалізації зв’язку між модулями програмного забезпечення за управлінням та як забезпечити коректну ієрархію викликів?

5. Які особливості має передача інформації між модулями програмного забезпечення та як досягається безпечне використання змінних у межах різних модулів?

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Кожевніков Г. К. Методичні вказівки до виконання практичних робіт з курсу «Технологія створення програмних продуктів». Харків, 2023. 45 с.
2. Бульба С. С., Молчанов Г. І., Савченко В. М. Програмування: навч.-метод. посіб. Харків: НТУ «ХПІ», 2024. 121 с.
3. Невлюдов І. Ш., Новоселов С. П., Сичова О. В. Технологія програмування промислових контролерів в інтегрованому середовищі CODESYS : навч. посіб. Харків: ХНУРЕ, 2020. 132 с.
4. Басьюк Т. М., Думанський Н. О., Пасічник О. В. Основи інформаційних технологій : навч. посіб. Львів: Новий Світ–2000, 2021. 390 с.
5. Мартін Р. Чистий код: створення і рефакторинг за допомогою Agile = Clean Code: A Handbook of Agile Software Craftsmanship. Харків: Фабула, 2021. 448 с.
6. Цибульник С. О., Барандич К. С. Технології розроблення програмного забезпечення. Частина 1. Життєвий цикл програмного забезпечення: підруч. Київ: КПІ ім. Ігоря Сікорського, 2022. 270 с.
7. Васильєв О. М. Програмування в Python. Теорія і практика: навч. посіб. Київ: Ліра-К, 2023. 462 с.
8. Васильєв О. М. Програмування мовою Java : навч. посіб. Тернопіль: Навчальна книга – Богдан, 2020. 696 с.
9. Амелін Р. В., Колесников С. А., Бутенко А. О. Agile-підходи в управлінні ІТ-проєктами : навч. посіб. Київ: КНЕУ, 2021. 158 с.
10. Дьяків А. І., Марченко Р. В. Основи DevOps та CI/CD : навч. посіб. Львів: ЛНУ ім. Івана Франка, 2022. 112 с.
11. Бондаренко Н. І., Кириченко О. А. UML-моделювання інформаційних систем : навч. посіб. Суми: СумДУ, 2020. 214 с.