КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА КІБЕРНЕТИКИ

Кафедра інтелектуальних програмних систем

«ЗАТВЕРДЖУЮ»
Заступник декана
з навчальної роботи

_____ Кашпур О.Ф.

«____»_____2018 року

РОБОЧА ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ ОСНОВИ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ

для студентів

галузь знань **12 «Інформаційні технології»**
(шифр і назва)

спеціальність **121 «Інженерія програмного забезпечення»**
(шифр і назва спеціальності)

освітній рівень **бакалавр**
(молодший бакалавр, бакалавр, магістр)

освітня програма **«Програмна інженерія»**
(назва освітньої програми)

вид дисципліни обов'язкова

Форма навчання	**денна**
Навчальний рік	**2020/2021**
Семестр	**3, 4**
Кількість кредитів ECTS	**8**
Мова викладання, навчання та оцінювання	**українська**
Форма заключного контролю	**іспит**

Викладачі: **к.ф.-м.н., асистент Жереб К.А.** (лекції, лабораторні заняття),
к.ф.-м.н., асистент Терлецький Д.О. (лабораторні заняття)

Пролонговано: на 20__/20__ н.р. _____(_____) «__»___ 20__р.
(підпис, ПІБ, дата)

на 20__/20__ н.р. _____(_____) «__»___ 20__р.
(підпис, ПІБ, дата)

КИЇВ – 2020

Розробник: Жереб Костянтин Анатолійович, к.ф.-м.н., асистент кафедри «Інтелектуальних програмних систем»

ЗАТВЕРДЖЕНО
Зав. кафедри «Інтелектуальних програмних систем»

_____ (Провотар О.І.)
(підпис) (прізвище та ініціали)

Протокол № ___ від «____» _____ 20___ р.

Схвалено науково-методичною комісією факультету комп'ютерних наук та кібернетики _____

Протокол від «____» _____ 20___ року №___

Голова науково-методичної комісії _____ (Омельчук Л.Л.)
(підпис) (прізвище та ініціали)

«_____» ___ _____ 20___ року

1. Мета дисципліни – формування у студентів основ знань, необхідних для розуміння принципів організації, функціонування і проектування об'єктно-орієнтованих програмних систем; опанування навичками проектування об'єктно-орієнтованих систем, зокрема з використанням графічної нотації UML; практичне опанування навичками програмування на мові C/C++, зокрема робота з функціями та вказівниками, об'єктами та класами, шаблонами, використання можливостей стандартних бібліотек.

2. Попередні вимоги до опанування або вибору навчальної дисципліни (*за наявності*):

1. Знати: основні поняття програмування та принципи розробки програм; базові алгоритми та структури даних.

2. Вміти: проектувати, розробляти та тестувати програми на базовому рівні.

3. Володіти елементарними навичками: програмування мовами C, C++.

3. Анотація навчальної дисципліни:

Навчальна дисципліна "Основи об'єктно-орієнтованого програмування" є складовою освітньо-професійної програми підготовки фахівців за першим (*бакалаврським*) рівнем вищої освіти *галузі знань* 12 „Інформаційні технології" зі *спеціальності 121* „Інженерія програмного забезпечення", *освітньо-професійної програми* „Програмна інженерія".

Дана дисципліна є обов'язковою навчальною дисципліною за ***програмою "Програмна інженерія"***.

Викладається у **3 та 4** семестрах **2** курсу в **обсязі – 240 год.**

(8 кредитів ECTS) зокрема: *лекції – 58 год., лабораторні – 56 год., консультації – 4 год., самостійна робота – 122 год.* У курсі передбачено **4** *змістовних частини* та 4 *контрольні роботи*. Завершується дисципліна – **іспитом в 4 семестрі.**

В результаті вивчення навчальної дисципліни студент повинен:

знати принципи організації та функціонування об'єктно-орієнтованих програмних систем; основні етапи життєвого циклу програмного забезпечення; основні методи та підходи до проектування об'єктно-орієнтованих програмних систем, зокрема використання патернів проектування; основні принципи об'єктно-орієнтованого програмування, зокрема інкапсуляцію, успадкування та поліморфізм.

вміти створювати програми з використанням мови програмування C/C++, використовувати засоби стандартних бібліотек мови C/C++; використовувати сучасні інструментальні засоби розробки, зокрема компілятори, інтегровані середовища розробки (IDE), сучасні засоби модульного тестування (unit testing), сучасні засоби командної розробки, зокрема системи контролю версій (revision control); проектувати об'єктно-орієнтовані програмні системи з використанням графічної нотації UML; знаходити та усувати дефекти в програмному коді.

Для допуску до дисципліни „ Основи об'єктно-орієнтованого програмування " освітньо-професійної програми «Програмна інженерія» студент повинен опанувати компетентності та результати навчання, які надають дисципліни „Основи програмування" та „Програмування" програми «Програмна інженерія». Дисципліна „Основи об'єктно-орієнтованого програмування " є базовою для засвоєння дисциплін «Об'єктно-орієнтоване програмування», «Розробка WEB-орієнтованих систем», «Операційні системи», «Програмування мовою Ruby», «Розподілені обчислення», «Інформаційні системи» програми «Програмна інженерія».

4. Завдання (навчальні цілі):

набуття знань, умінь та навичок (компетентностей) на рівні новітніх досягнень у програмуванні, відповідно до кваліфікації фахівець з інформаційних технологій. Зокрема, розвивати:

- здатність до абстрактного мислення, аналізу та синтезу (ЗК01);
- здатність застосовувати знання у практичних ситуаціях (ЗК02);
- здатність спілкуватися державною мовою як усно, так і письмово (ЗК03);
- здатність вчитися і оволодівати сучасними знаннями (ЗК05);
- здатність до пошуку, оброблення та аналізу інформації з різних джерел (ЗК06);
- здатність розробляти архітектури, модулі та компоненти програмних систем (СК03);
- володіння знаннями про інформаційні моделі даних та системи, здатність створювати програмне забезпечення для зберігання, видобування та опрацювання даних (СК07);
- здатність обґрунтовано обирати та освоювати інструментарій з розробки та супроводження програмного забезпечення (СК13);
- здатність до алгоритмічного та логічного мислення (СК14).

5. Результати навчання за дисципліною:

Результат навчання (1. знати; 2. вміти; 3. комунікація; 4. автономність та відповідальність)		**Форми (та/або методи і технології) викладання і навчання**	**Методи оцінювання та пороговий критерій оцінювання (за необхідності)**	**Відсоток у підсумковій оцінці з дисципліни**
Код	**Результат навчання**			
РН1.1	*Знати принципи організації та функціонування об'єктно-орієнтованих програмних систем*	*Лекція, лабораторне заняття*	*Тест, 60% правильних відповідей, іспит*	8%
РН1.2	*Знати основні етапи життєвого циклу програмного забезпечення*	*Лекція, лабораторне заняття*	*Тест, 60% правильних відповідей, іспит*	8%
РН1.3	*Знати основні методи та підходи до проектування об'єктно-орієнтованих програмних систем, зокрема використання патернів проектування*	*Лекція, лабораторне заняття*	*Тест, 60% правильних відповідей, іспит*	16%
РН1.4	*Знати основні принципи об'єктно-орієнтованого програмування, зокрема інкапсуляцію, успадкування та поліморфізм*	*Лекція, лабораторне заняття*	*Тест, 60% правильних відповідей, іспит*	8%
РН2.1	*Вміти створювати програми з використанням мови програмування C/C++, використовувати засоби стандартних бібліотек мови C/C++.*	*Лабораторне заняття, самостійна робота*	*Захист лабораторної роботи, іспит*	10%
РН2.2	*Вміти використовувати сучасні інструментальні засоби розробки, зокрема компілятори, інтегровані середовища розробки (IDE), сучасні засоби модульного тестування (unit testing), сучасні засоби командної розробки, зокрема системи контролю версій (revision control)*	*Лабораторне заняття, самостійна робота*	*Захист лабораторної роботи, іспит*	10%
РН2.3	*Вміти проектувати об'єктно-*	*Лабораторне заняття, самостійна*	*Захист лабораторної*	10%

	орієнтовані програмні системи з використанням графічної нотації UML	*робота*	*роботи, іспит*	
РН2.4	*Вміти знаходити та усувати дефекти в програмному коді*	*Лабораторне заняття, самостійна робота*	*Захист лабораторної роботи, іспит*	10%
РН3.1	*Обґрунтовувати власний погляд на задачу, спілкуватися з колегами з питань проектування та розробки програм, писати технічну документацію програм*	*Лабораторне заняття*	*Поточне оцінювання, захист ЛР*	10%
РН4.1	*Організовувати самостійну роботу для вивчення нових технологій*	*Самостійна робота*	*Поточне оцінювання, Захист лабораторної роботи*	5%
РН4.2	*Відповідально ставитися до виконуваних робіт, нести відповідальність за їх якість*	*Лабораторна робота*	*Захист лабораторної роботи*	5%

6. Співвідношення результатів навчання дисципліни із програмними результатами навчання

Результати навчання дисципліни **Програмні результати навчання**	**РН1.1**	**РН1.2**	**РН1.3**	**РН1.4**	**РН2.1**	**РН2.2**	**РН2.3**	**РН2.4**	**РН3.1**	**РН4.1**	**РН4.2**
(з опису освітньої програми)											
ПРН01. Аналізувати, цілеспрямовано шукати і вибирати необхідні для вирішення професійних завдань інформаційно-довідникові ресурси і знання з урахуванням сучасних досягнень науки і техніки.	+	+									
ПРН04. Знати і застосовувати професійні стандарти і інші нормативно-правові документи в галузі інженерії програмного забезпечення.		+				+					
ПРН05. Знати і застосовувати відповідні математичні поняття, методи доменного, системного і об’єктно-орієнтованого аналізів та математичного моделювання для розробки програмного забезпечення.	+		+								+
ПРН07. Знати і застосовувати на практиці фундаментальні концепції, парадигми і основні принципи функціонування мовних, інструментальних і обчислювальних засобів інженерії програмного забезпечення.						+					+
ПРН08. Вміти розробляти людино-машинний інтерфейс				+	+			+			
ПРН10. Проводити передпроектне обстеження предметної області, системний аналіз об'єкта проектування.		+	+							+	
ПРН12. Застосовувати на практиці ефективні підходи щодо проектування програмного забезпечення.			+				+				

ПРН13. Знати і застосовувати методи розробки алгоритмів, конструювання програмного забезпечення та структур даних і знань.				+	+						
ПРН14. Застосовувати на практиці інструментальні програмні засоби доменного аналізу, проектування, тестування, візуалізації, вимірювань та документування програмного забезпечення.						+	+	+			
ПРН15. Мотивовано обирати мови програмування та технології розробки для розв’язання завдань створення і супроводження програмного забезпечення.									+	+	
ПРН16. Мати навички командної розробки, погодження, оформлення і випуску всіх видів програмної документації.		+							+		
ПРН17. Вміти застосовувати методи компонентної розробки програмного забезпечення.								+			

7. Схема формування оцінки.

7.1 Форми оцінювання студентів:

- семестрове оцінювання:

1. Контрольна робота (тест) 1: РН 1.1, РН 1.2 — 10 балів/6 балів.
2. Контрольна робота (тест) 2: РН1.3, РН 1.4 - 10 балів/6 балів.
3. Лабораторна робота 1 (проект): РН1.1, РН2.1, РН3.1 – 10 балів/6 балів.
4. Лабораторна робота 2 (проект): РН1.2, РН2.1, РН2.2, РН3.1 – 10 балів/6 балів.
5. Лабораторна робота 3 (проект): РН1.3, РН2.3, РН3.1 – 10 балів/6 балів.
6. Лабораторна робота 4 (проект): РН1.4, РН2.4, РН3.1 – 10 балів/6 балів.

- підсумкове оцінювання (у формі іспиту) *вказується:*
- максимальна кількість балів які можуть бути отримані студентом: 40 балів;
- результати навчання які будуть оцінюватись: РН1.1, РН1.2, РН1.3, РН2.3;
- форма проведення і види завдань: письмова.
Види завдань: 4 письмових завдання.

Критерії оцінювання на іспиті

Завдання	Тема завдання	Максимальний відсоток від 40 балів	Всього відсотків
Завдання 1	Письмове запитання з принципів ООП	25%	25%
Завдання 2	Письмове запитання з патернів	25%	25%
Завдання 3	Письмове запитання з проектування програм	25%	25%
Завдання 4	Письмове запитання з графічної нотації UML	25%	25%
			100%

Запитання для підготовки до іспиту

1. Визначення класу. Дані-члени класу.
2. Функції члени класу .Доступ до членів класу.
3. Функції члени класу з кваліфікаторами const та volatile. Визначення mutable.
4. Закриті та відкриті функції-члени.
5. Неявний вказівник this. Використання вказівника this.
6. Статичні члени класу. Статичні функції члени класу.
7. Вказівник на члени класу. Робота з вказівниками на члени класу.
8. Об’єднання. Бітове поле.
9. Поняття області видимості класу. Розв’язання імен в області видимості класу.
10. Вкладені класи. Розв’язання імен в області видимості вкладеного класу.
11. Ініціалізація класу. Конструктор класу.
12. Конструктор за замовченням. Обмеження прав на створення об’єкта.
13. Копіювальний конструктор. Деструктор класу. Явний виклик деструктора.
14. Масиви та вектори об’єктів Ініціалізація масиву в кучі. Почлена ініціалізація.
15. Перевантаження операторів.
16. Operator=, operator[], operator(), operator++, operator new, operator delete.
17. Перетворення визначені користувачем.

18. Конвертори.
19. Конвертер як конструктор.
20. Визначення шаблона класу.
21. Конкретизація шаблона класу. Аргументи шаблона для параметрів констант.
22. Функції – члени шаблонів класів. Друзі в шаблонах класів.
23. Статичні члени в шаблонах класів. Вкладені типи шаблонів класів. Шаблони – члени.
24. Наслідування. Визначення ієрархії класів.
25. Ідентифікація членів ієрархії. Визначення базового класу. Визначення похідних класів.
26. Доступ до членів базового класу. Конструювання базового та похідного класу. Конструктор базового класу. Конструктор похідного класу.
27. Деструктор. Віртуальні функції в базовому та похідному класі.
28. Чисто віртуальні функції. Статичний виклик віртуальної функції. Віртуальні деструктори.
29. Множинне успадкування. Відкрите, закрите та захищене успадкування.
30. Успадкування та композиція. Композиція об'єктів.
31. Віртуальне успадкування. Визначення віртуального базового класу.
32. Семантика ініціалізації. Порядок викликів конструкторів та деструкторів.
33. Видимість членів віртуального базового класу.
34. Ідентифікація типів під час виконання. Оператор dynamic_cast.
35. Оператор typeid. Клас type_info. Виключення.
36. Оператор try{} catch{}. Об'єкти виключення. Виключення та успадкування.
37. Розподіл вимог по суб'єктам та прецедентам. Діаграма прецедентів.
38. Діаграма класів. Асоціація, агрегація, композиція використання та наслідування.
39. Діаграма послідовностей. Діаграма взаємодії.
40. Діаграма компонентів. Діаграма розгортання.

Студент не допускається до іспиту, якщо під час семестру набрав менше ніж 20 балів. Студент допускається до іспиту за умови виконання 70% передбачених планом лабораторних робіт.

7.2 Організація оцінювання:

Терміни проведення форм оцінювання:

1. Контрольна робота (тест): до 7 тижня семестру.
2. Контрольна робота (тест): до 15 тижня семестру.
3. Лабораторна робота 1 (проект): до 4 тижня семестру.
4. Лабораторна робота 2 (проект): до 8 тижня семестру.
5. Лабораторна робота 3 (проект): до 12 тижня семестру.
6. Лабораторна робота 4 (проект): до 15 тижня семестру.

Студент має право на одне перескладання кожної контрольної роботи із можливістю отримання максимально 80% початково визначених за цю контрольну роботу балів. Термін перескладання визначається викладачем.

У випадку відсутності студента з поважних причин відпрацювання та перездачі контрольних робіт здійснюються у відповідності до „Положення про порядок оцінювання знань студентів при кредитно-модульній системі організації навчального процесу" від 1 жовтня 2010 року.

У разі неякісного виконання лабораторної роботи, викладач має право не зарахувати лабораторну роботу, або знизити за неї бали.

Студент має право здавати лабораторні роботи після закінчення визначеного для них терміну, але з втратою одного балу за кожен тиждень, який пройшов з моменту закінчення терміну її здачі.

7.3 Шкала відповідності оцінок

Відмінно / Excellent	90-100
Добре / Good	75-89
Задовільно / Satisfactory	60-74
Незадовільно / Fail	0-59
Зараховано / Passed	60-100
Не зараховано / Fail	0-59

8. Структура навчальної дисципліни. Тематичний план лекцій і лабораторних занять

III семестр

№ лекції	Назва лекції	Кількість годин		
		лекції	лаборат.	С/Р
Частина 1. *Основи програмування з використанням мови С/С++*				
1	**Тема 1** *Процедурне та об'єктно-орієнтоване програмування.*	2	2	4
2	**Тема 2.** *Програмне середовище С/С++*	2	2	4
3	**Тема 3.** *Базові поняття програмування*	2	2	4
4	**Тема 4.** *Використання вказівників (pointers)*	2	2	5
5	**Тема 5.** *Використання псевдонімів (reference)*	2	2	4
6	**Тема 6.** *Вирази та інструкції*	2	2	4
7	**Тема 7.** *Типи: масиви, ланцюжки символів, структури*	1	2	5
	Контрольна робота 1	1		
Частина 2 *Процедурне програмування*				
8	**Тема 8.** *Функції в мові С/С++*			
9	**Тема 9.** *Сигнатура функції*	2	2	5
10	**Тема 10.** *Передача параметрів*	2	2	5
11	**Тема 11.** *Обчислення значення функції*	2	2	4
12	**Тема 12.** *Довизначення (overloading) функцій*	2	2	5
13	**Тема 13.** *Непряме використання функцій*	2	2	4
14	**Тема 14.** *Області видимості*	2	1	5
	Контрольна робота 2	1		
	ВСЬОГО	**30**	**28**	**62**

Загальний обсяг ***120*** *год,* в тому числі:
Лекцій – ***28*** *год.*
Лабораторні роботи – ***28*** *год.*
Самостійна робота - ***62*** *год.*
Консультації - ***2*** *год.*

IV семестр

№ лекції	Назва лекції	Кількість годин		
		лекції	лаборат.	С/Р
Частина 3. *Об'єктно-орієнтоване програмування*				
1	**Тема 16** *Класи. Об'єкти класів.*	2	2	4
2	**Тема 17.** *Функції-члени класів*	2	2	4
3	**Тема 18.** *Статичні члени класів*	2	2	4
4	**Тема 19.** *Об'єднання. Бітове поле. Область видимості класа*	2	2	4
5	**Тема 20.** *Вкладені класи. Класи як простір імен*	2	2	4
6	**Тема 21.** *Ініціалізація, конструктори та деструктори класів*	2	2	4
7	**Тема 22.** *Перевантаження операторів та перетворення типів*	1	2	4
	Контрольна робота 3	1		
Частина 4. Проектування об'єктно-орієнтованих систем				
8	**Тема 23.** *Шаблони класів (templates)*			
9	**Тема 24.** *Наслідування, віртуальні функції*	2	2	4
10	**Тема 25.** *Множинне наслідування*	2	2	4
11	**Тема 26.** *Віртуальне наслідування*	2	2	4
12	**Тема 27.** *Механізм RTTI. Виключення*	2	2	4
13	**Тема 28.** *Мова UML. Діаграма прецедентів, класів та об'єктів*	2	2	4
14	**Тема 29.** *Діаграми послідовностей, станів та діяльності*	2	1	4
15	**Тема 30.** *Патерни проектування (design patterns)*	1	1	4
	Контрольна робота 4	1		
	ВСЬОГО	**30**	**28**	**60**

Загальний обсяг ***120*** *год,* в тому числі:
Лекцій – ***30*** *год.*
Лабораторні роботи – ***28*** *год.*
Самостійна робота - ***60*** *год.*
Консультації - ***2*** *год.*

Теми, винесені на самостійне вивчення:
Використання систем контролю версій.
Виконання лабораторних робіт 1-4.
Умови лабораторних робіт:
Лабораторна робота 1: Розробка програми для 3D моделювання.
Лабораторна робота 2: Розробка мережевого застосунку.
Лабораторна робота 3: Проектування програмної системи з використанням UML.
Лабораторна робота 4: Розробка застосунку для Android.
Деталізовані умови лабораторних робіт розміщено за посиланнями:
https://drive.google.com/drive/folders/1285vxEpjq5GEREq7IVXFjVHDg9G_OXtm

9. Рекомендовані джерела:

Основна

1. Bjarne Stroustrup: The C++ Programming Language, 4th Edition. // Addison-Wesley Professional, 2013.
2. Bjarne Stroustrup: Programming: Principles and Practice Using C++, 2nd Edition // Addison-Wesley Professional, 2014.
3. Bjarne Stroustrup: A Tour of C++, 2nd Edition. // Addison-Wesley Professional, 2018.
4. Scott Meyers: Effective C++ Digital Collection: 140 Ways to Improve Your Programming, 1st Edition // Addison-Wesley Professional, 2012.
5. Scott Meyers: Effective Modern C++: 42 Specific Ways to Improve Your Use of C++11 and C++14 // O'Reilly Media, Inc., 2015.
6. Вирт Н. Систематическое программирование. Введение. М. Мир, 1977
7. Липпман С. Лажойе Ж. Язык программирования С++. Вводный курс. М. ДМК. 2001
8. Пильщиков В. Сборник упражнений по языку Паскаль. М. Наука 1989.
9. Завьялов Ю., Квасов Б., Мирошниченко В. Методы сплайн-функций М. Наука 1980.
10. Саттер Г. Решение сложных задач на С++. М. Вильямс 2002.
11. Самарский А. Введение в теорию разностных схем М. Наука 1971.

Додаткова:

12. Jim Arlow, Ila Neustadt: *UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design, 2nd Edition* // Addison-Wesley Professional, 2005.
13. Russ Miles, Kim Hamilton: *Learning UML 2.0* // O'Reilly Media, Inc., 2006.
14. Grady Booch, Robert A. Maksimchuk, Michael W. Engle, Bobbi J. Young, Jim Conallen, Kelli A. Houston: *Object-Oriented Analysis and Design with Applications, 3rd Edition* // Addison-Wesley Professional, 2007.
15. Hassan Gomaa: Software Modelling and Design: *UML, Use Cases, Patterns, and Software Architectures* // Cambridge University Press, 2011.
16. Крэг Ларман: *Применение UML 2.0 и шаблонов проектирования: Введение в объектно-ориентированный анализ, проектирование и итеративную разработку, 3-е издание* // Издательский дом "Вильясм", 2013.

17. Raul Sidnei Wazlawick: *Object-Oriented Analysis and Design for Information Systems: Modeling with UML, OCL, and IFML* // Elsevier, 2014.
18. Martina Seidl, Marion Scholz, Christian Huemer, Gerti Kappel: *UML @ Classroom: An Introduction to Object-Oriented Modeling* // Springer, 2015.
19. Bhuvan Unhelkar: *Software Engineering with UML* // CRC Press, 2018.
20. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: Design Patterns: *Elements of Reusable Object-Oriented Software* // Addison-Wesley Professional, 1994.
21. Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad: *Pattern-Oriented Software Architecture, Volume 1: A System of Patterns* // Wiley, 1996.
22. Martin Fowler: *Patterns of Enterprise Application Architecture, 1st Edition* // Addison-Wesley Professional, 2003.
23. Эрих Гамма, Ричард Хелм, Ральф Джонсон, Джон Влиссидес: *Приемы объектно-ориентированного проектирования. Паттерны проектирования.* // Питер, 2015.
24. Stephan Roth: Clean C++: *Sustainable Software Development Patterns and Best Practices with C++ 17* // Apress, 2017.
25. Dmitri Nesteruk: *Design Patterns in Modern C++: Reusable Approaches for Object-Oriented Software Design* // Apress, 2018.
26. Макс Шлее: *Qt 5.10. Профессиональное программирование на C++* // БХВ-Петербург, 2018.
27. Lee Zhi Eng: *Hands-On GUI Programming with C++ and Qt5, 1st Edition* // Packt Publishing, 2018.
28. Guillaume Lazar, Robin Penea: *Mastering Qt 5, 2nd Edition* // Packt Publishing, 2018.
29. Lee Zhi Eng: *Qt5 C++ GUI Programming Cookbook, 2nd Edition* // Packt Publishing, 2019.
30. Kent Beck: Test Driven Development: By Example // Addison-Wesley Professional, 2002.
31. Gerard Meszaros: xUnit Test Patterns: Refactoring Test Code // Addison-Wesley, 2007
32. Steve Freeman, Nat Pryce: Growing Object-Oriented Software Guided by Tests // Addison-Wesley Professional, 2009.
33. Roy Osherove: The Art of Unit Testing, 2nd Edition // Manning Publications, 2013.
34. Martin Fowler: Refactoring: Improving the Design of Existing Code, 2nd Edition // Addison-Wesley Professional, 2018.
35. Евстигнеев В. Применение теории графов в программировании. М. Наука 1985.
36. Мацяшек Л. Анализ требований и проектирование систем. М. Вильямс 2002.
37. Черников С. Линейные неравенства. М.Наука 1968.
38. Акулич И. Математическое програмирование в примерах и задачах. М.Высш.шк.,1985.
39. https://msdn.microsoft.com/
40. http://www.cplusplus.com/
41. http://en.cppreference.com/
42. https://git-scm.com/book/.

10. Додаткові ресурси:

https://drive.google.com/drive/folders/1285vxEpjq5GEREq7IVXFjVHDg9G_OXtm
https://drive.google.com/drive/folders/1X_NdykxXuAX0c5oHoBlHpqHqt5NqNzgx
https://drive.google.com/drive/folders/11aZsa5bRJPUUOWnMKEHal4v_NyAO_jLV