МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Львівський національний університет імені Івана Франка
Факультет прикладної математики та інформатики
Кафедра програмування

Затверджено
На засіданні кафедри програмування
факультету прикладної математики
Львівського національного університету
імені Івана Франка
(протокол № 1 від 29 серпня 2024 р.)

Зав. кафедри к. ф.-м. н., доц. Ярошко С. А.

Силабус навчальної дисципліни
«Програмування» (I-II семестри),
що викладається в межах ОПП Інформатика
першого (бакалаврського) рівня вищої освіти
для здобувачів за спеціальністю 122 Комп'ютерні науки

Львів 2024 р.

Назва дисципліни	Програмування
Адреса викладання дисципліни	Львівський національний університет імені Івана Франка, вул. Університетська 1, м. Львів, Україна, 79000
Факультет та кафедра, за якою закріплена дисципліна	Факультет прикладної математики та інформатики, кафедра програмування
Галузь знань, шифр та назва спеціальності	Галузь знань: 12 Інформаційні технології Спеціальність: 122 Комп'ютерні науки
Викладачі дисципліни	Ярошко Сергій Адамович, к. ф.-м. н., доцент, завідувач кафедри програмування; Клакович Леся Миронівна, к. ф.-м. н., доцент, доцент кафедри програмування
Контактна інформація викладачів	Електронна пошта: serhiy.yaroshko@lnu.edu.ua, lesya.klakovych@lnu.edu.ua веб-сторінки: https://ami.lnu.edu.ua/employee/yaroshko https://ami.lnu.edu.ua/employee/klakovych
Консультації з питань навчання по дисципліні відбуваються	Консультації проводять раз на тиждень згідно з оприлюдненим розкладом консультацій викладача. Можливі он-лайн консультації через Zoom чи Microsoft Teams. Для погодження часу он-лайн консультацій слід писати на електронну пошту викладача.
Сторінка курсу	https://ami.lnu.edu.ua/course/programming-csit1
Інформація про дисципліну	Курс "Програмування" є нормативною дисципліною зі спеціальності 122 Комп'ютерні науки для освітньо-професійної програми «Інформатика», яку викладають у першому і другому семестрах в обсязі 8 кредитів (за Європейською кредитно-трансферною системою ECTS)
Коротка анотація дисципліни	Фокус уваги курсу спрямовано на те, щоб навчити вчорашнього випускника писати досить складні ефективні комп'ютерні програми. Одним з предметів вивчення і, водночас, головним засобом навчання є мова програмування C++. Перший семестр навчання закладає базу: дає основи алгоритмізації та вступ до об'єктно-орієнтованого програмування. У другому семестрі вивчають тонкощі ООП та узагальнене програмування.
Мета та цілі дисципліни	Метою нормативної дисципліни «Програмування» є навчити студента: • складати алгоритми, оцінювати їхню обчислювальну ефективність, перевіряти правильність за допомогою сукупності тестів; • вибирати відповідні засоби мови програмування і застосовувати їх належним чином для запису своїх алгоритмів розв'язування прикладних задач; • застосовувати процедурну методологію побудови комп'ютерних програм; • застосовувати об'єктно-орієнтований підхід до проектування і написання програм, розуміти принципи інкапсуляції, наслідування структури і поведінки об'єктів, суть і способи реалізації поліморфізму поведінки та структур даних; • застосовувати принципи узагальненого програмування для побудови та використання шаблонів функцій та класів, для метапрограмування; • використовувати узагальнені алгоритми та контейнери стандартної бібліотеки STL для написання власних програм, доповнювати можливості бібліотеки власними шаблонами.
Література для вивчення дисципліни	*Основна література* 1. Ярошко С. А. Методи розробки алгоритмів. Програмування мовою C++: навч. посібник / С.А. Ярошко, О.С. Ярошко – Львів: ЛНУ імені Івана Франка, 2022. – 248 с. – https://lnuittutor.github.io/ 2. Бублик В.В. Об'єктно-орієнтоване програмування: [Підручник] / В.В. Бублик. – К.: ІТ-книга, 2015. – 624 с. 3. Stephen Prata C++ Primer Plus 6th Edition (Developer's Library) In 2 Volumes – Adisson-Wesley Professional, 2011. – 1440 p. 4. Дудзяний І.М. Програмування мовою С++. Частина 1: Парадигма процедурного програмування: навчальний посібник / І.М. Дудзяний. – Львів: ЛНУ імені Івана Франка, 2013. – 468 с.

	5. Microsoft Learn: C++ language documentation. – Електронний ресурс. Режим доступу: https://learn.microsoft.com/en-us/cpp/cpp/?view=msvc-170 *Додаткова література* 6. Bruce Eckel Thinking in C++, Vol. 1: Introduction to Standard C++, 2nd Edition – Prentice Hall, 2000. – 814 p. 7. Bruce Eckel Thinking in C++, Vol. 2: Practical Programming, 2nd Edition – Prentice Hall, 2003. – 832 p. 8. Scott Meyers Effective Modern C++ – O'Reilly Media, 2015. – 316 p. 9. Nicolai M. Josuttis The C++ Standard Library. A tutorial and Reference. Second Eddition – Addison-Wesley, 2012. – 1162 p.
Обсяг курсу	8 кредитів ЄКТС – 240 годин. З них 32 + 32 годин лекцій, 32 + 32 години лабораторних занять та 56 + 56 годин самостійної роботи (1-й + 2-й семестр відповідно)
Очікувані результати навчання	Після завершення цього курсу студент буде: *знати* • синтаксис мови програмування C++, склад і правила використання бібліотеки STL; • засоби та можливості середовища програмування, зокрема, щодо тестування та налагодження програм; • засади модульного, об'єктно-орієнтованого та узагальненого програмування; • призначення та влаштування базових структур даних і алгоритмів для їхнього опрацювання. *вміти* • писати та налагоджувати ефективні комп'ютерні програми мовою C++; • проектувати, оголошувати та використовувати ієрархії класів, шаблони класів; • використовувати та доповнювати засоби бібліотеки STL; • реалізовувати базові патерни проектування програмного забезпечення; • використовувати хмарні сховища репозиторіїв програмного коду для зберігання та поширення власних програм, для роботи в команді.
Компетентності	*Інтегральна:* Здатність розв'язувати складні спеціалізовані задачі та практичні проблеми у галузі комп'ютерних наук або у процесі навчання, що передбачають застосування теорій та методів інформаційних технологій і характеризується комплексністю та невизначеністю умов. *Загальні (ЗК):* ЗК1. Здатність до абстрактного мислення, аналізу та синтезу. ЗК2. Здатність застосовувати знання у практичних ситуаціях. ЗК3. Знання та розуміння предметної області та розуміння професійної діяльності. ЗК7. Здатність до пошуку, оброблення та аналізу інформації з різних джерел. ЗК8. Здатність генерувати нові ідеї (креативність). ЗК9. Здатність працювати в команді. ЗК12. Здатність оцінювати та забезпечувати якість виконуваних робіт. *Спеціальні (фахові, предметні) компетентності (СК):* СК3. Здатність до логічного мислення, побудови логічних висновків, використання формальних мов і моделей алгоритмічних обчислень, проектування, розроблення й аналізу алгоритмів, оцінювання їх ефективності та складності, розв'язності та нерозв'язності алгоритмічних проблем для адекватного моделювання предметних областей і створення програмних та інформаційних систем. СК7. Здатність застосовувати теоретичні та практичні основи методології та технології моделювання для дослідження характеристик і поведінки складних об'єктів і систем, проводити обчислювальні експерименти з обробкою й аналізом результатів. СК8. Здатність проектувати та розробляти програмне забезпечення із застосуванням різних парадигм програмування: узагальненого, об'єктно-орієнтованого, функціонального, логічного, з відповідними моделями, методами й алгоритмами обчислень, структурами даних і механізмами управління.

	СК10. Здатність застосовувати методології, технології та інструментальні засоби для управління процесами життєвого циклу інформаційних і програмних систем, продуктів і сервісів інформаційних технологій відповідно до вимог замовника.
Програмні результати навчання	ПР1. Застосовувати знання основних форм і законів абстрактно-логічного мислення, основ методології наукового пізнання, форм і методів вилучення, аналізу, обробки та синтезу інформації в предметній області комп'ютерних наук. ПР5. Проектувати, розробляти та аналізувати алгоритми розв'язання обчислювальних та логічних задач, оцінювати ефективність та складність алгоритмів на основі застосування формальних моделей алгоритмів та обчислюваних функцій. ПР9. Розробляти програмні моделі предметних середовищ, вибирати парадигму програмування з позицій зручності та якості застосування для реалізації методів та алгоритмів розв'язання задач в галузі комп'ютерних наук. ПР15. Застосовувати знання методології та CASE-засобів проектування складних систем, методів структурного аналізу систем, об'єктно-орієнтованої методології проектування при розробці і дослідженні функціональних моделей організаційно-економічних і виробничо-технічних систем.
Ключові слова	Алгоритм, програма мовою С++, тип даних, інструкція галуження, інструкція повторення, масив, вказівник, функція, клас, об'єкт, наслідування, поліморфізм, віртуальний метод, шаблон класу, контейнер, функтор.
Формат курсу	Очний: проведення лекцій, лабораторних робіт та консультацій в приміщеннях університету, а в умовах карантину – онлайновий на платформі Microsoft Teams

Теми

Тижд.	Тема, план, короткі тези	Форма заняття	Тривалість, год	Термін виконання
1.1	Що таке "комп'ютер"? Кодування інформації. Мови програмування. Структура програми мовою С++, include, main(). Стандартні заголовкові файли, порядок трансляції.	Лекція	2	
	Системи числення: позиційні, непозиційні. Основа системи числення. Двійкова система: алгоритми переведення, арифметика. Кодування від'ємних чисел.	Лабораторна робота	2	Наступне лабораторне заняття
1.2	Система типів. Логічний тип: зображення, оператори, вирази. Літерний тип. Зображення літер. Цілі типи: назви, граничні значення, зображення, оператори. Оператори присвоєння. Дійсні типи. Перетворення типів. Математичні функції. Задача про трикутник. Використання cin, cout.	Лекція	2	
	Вісімкова, шістнадцяткова системи числення: переведення, арифметика, зв'язок з двійковою. Контрольна робота "Системи числення".	Контрольна робота	2	
1.3	Послідовні та галужені алгоритми. Інструкції if, if else, switch, break. Задача max(a,b,c). Задача "Ми знайшли k грибів". Оператор ?:. Повторювані алгоритми. Інструкції циклу for, while, do while. Оголошення в інструкціях. Задача про кількість цифр числа. Введення послідовностей чисел (стан потоку введення). Інструкції break, continue, goto.	Лекція	2	
	Програмна реалізація простих послідовних алгоритмів: обчислення за математичними формулами, найпростіші задачі цілочислової арифметики. Правила зображення цілих і дійсних чисел. Запис арифметичних і логічних виразів, використання стандартних функцій. Оформлення введення, виведення даних.	Лабораторна робота	2	Наступне лабораторне заняття
1.4	Модифікатор const, тип перелік. Загальна структура оголошення, typedef. Структуровані типи даних: масиви, рядки. Введення-виведення рядків. Функції для роботи з рядками (cstring). Використання масивів для зберігання/побудови послідовності значень.	Лекція	2	
	Програмна реалізація галужених алгоритмів. Використання умовних інструкцій: повної, вкороченої, вкладеної. Використання інструкції вибору варіанту.	Лабораторна робота	2	Наступне лабораторне заняття
1.5	Вказівники, арифметика вказівників, застосування до масивів. Динамічні змінні: створення, використання, знищення. Введення-виведення масивів. Робота з двохвимірними масивами. Побудова лінійного списку за допомогою вказівників.	Лекція	2	
	Програмна реалізація повторюваних алгоритмів. Прості цикли. Взаємозамінність різних інструкцій циклу. Опрацювання послідовностей значень без зберігання у пам'яті та з використанням статичних масивів.	Лабораторна робота	2	Наступне лабораторне заняття

	1.6	Модульне програмування. Функції в С++: прототип, визначення, виклик. Параметри-значення, параметри-вказівники, Вказівники і специфікатор const. Тип посилання, використання в параметрах функцій.	Лекція	2	
		Поєднання циклу з галуженням. Вкладені цикли (у матричних задачах). Контрольна робота "Основи алгоритмізації"	Контрольна робота	2	
	1.7	Функції і масиви. Впорядкування елементів масиву. Вказівник на функцію, масиви функцій, функції вищих порядків. Використання лямбда-виразів.	Лекція	2	
		Створення, використання, знищення динамічних одно- та двовимірних масивів (у тому числі масивів літер). Оголошення функцій для обчислень виразів, для роботи з масивами.	Лабораторна робота	2	Наступне лабораторне заняття
	1.8	Структуровані типи даних С++: структури, бітові поля, об'єднання. Конструктори, деструктори структур, селектор імені. Перевантаження операторів: арифметичних, введення-виведення. Функції для опрацювання структур.	Лекція	2	
		Оголошення і використання функцій, що змінюють свої аргументи: параметри-вказівники та параметри-посилання. Використання вказівника на функцію: алгоритми числового інтегрування, наближене розв'язування рівнянь.	Лабораторна робота	2	Наступне лабораторне заняття
	1.9	Поняття зв’язної структури даних. Тип для моделювання ланок лінійного однозв’язного списку. Побудова списку з послідовності значень. Перебір елементів списку. Вставляння значення у впорядкований список. Вилучення списку з динамічної пам’яті. Сортування списком. Алгоритм злиття впорядкованих послідовностей.	Лекція	2	
		Оголошення і використання типу struct: конструктори, оператори введення-виведення. Моделювання одно- та двозв’язного списку. Функція побудови списку, виведення, пошуку значення тощо.	Лабораторна робота	2	Наступне лабораторне заняття
	1.10	Структура даних дерево. Поняття рекурсії. Рекурсивні розв’язки, рекурсивні функції. Алгоритми обходу двійкового дерева. Дерева пошуку. Сортування деревом. Збалансовані дерева, вставка, вилучення.	Лекція	2	
		Моделювання двійкового дерева, дерева пошуку. Рекурсивні та нерекурсивні алгоритми обходу дерева.	Лабораторна робота	2	Наступне лабораторне заняття
	1.11	Введення/виведення даних і використання файлів у С++. Потоки, буфери, файли. Робота з багатьма файлами. Текстові та двійкові файли, формати *csv* та *json*.	Лекція	2	
		Впорядкування масиву. Впорядкування списку. Застосування різних алгоритмів. Перевірка правильності виконання.	Лабораторна робота	2	Наступне лабораторне заняття
	1.12	Аргументи функцій за замовчуванням. Статичні змінні функції. Поліморфізм і перевантаження функцій. Вбудовані функції. Оголошення та використання просторів імен. Класи пам'яті.	Лекція	2	
		Завантаження колекції значень з файла. Зберігання даних до файла. Функції з параметрами за замовчуванням.	Лабораторна робота	2	Наступне лабораторне заняття
	1.13	Клас – тип, оголошений користувачем. Термінологія. Прихована і доступна частини класу. Конструктори: за замовчуванням, з параметрами. Клас "Пакет акцій". Розташування програмного коду у файлах.	Лекція	2	
		Оголошення простору імен, розташування в ньому типів, функцій. Використання. Контрольна робота "Модульне програмування"	Контрольна робота	2	
	1.14	Оголошення класу і визначення методів класу. Використання дружніх функцій. Перевантаження операторів: різні способи. Перетворення типів: конструктори, оператори. Клас "Time"	Лекція	2	
		Оголошення та використання класу. Конструктори з параметрами та за замовчуванням. Перевантаження операторів.	Лабораторна робота	2	Наступне лабораторне заняття
	1.15	Обробка помилок за допомогою винятків: захищений блок, перехоплення, запуск, повторний запуск винятків. Розгортання стека. Стандартні класи винятків С++. Типи винятків, оголошені користувачем	Лекція	2	
		Об'єкти, що використовують рядки. Визначення конструктора копіювання, перевизначення оператора присвоєння.	Лабораторна робота	2	Наступне лабораторне заняття
	1.16	Модульне тестування. Як додати до рішення проект модульного тестування. Структура класу тестів. Методи-тести. Різновиди статичних методів класу Assert.	Лекція	2	

		Контрольна робота "Об’єктне програмування"	Контрольна робота	2	
	1.**	Система контролю версій git: технології, система команд, гілки. Сценарії використання: один програміст – віддалене сховище; співавторство, два програмісти – віддалене сховище, конфлікти версій, планування.	Лекція	онлайн	
	2.1	Ще раз про оголошення класу. Приватні методи. Клас "прямокутник". Наслідування типів, видимість членів типу. Відношення "is-a". Клас "квадрат". Конструктори підкласів. Розширені правила сумісності. Поліморфізм вказівників, посилань. Перевизначення методів, поліморфізм поведінки. Віртуальні методи та пізнє зв'язування.	Лекція	2	
		Оголошення класу, що містить залежні поля та приватні методи. Визначення всіх конструкторів та деструктора. Використання статичних полів і статичних методів. Модульне тестування написаного коду.	Лабораторна робота	2	Наступне лабораторне заняття
	2.2	Деструктори підкласів. Оператори уведення-виведення для ієрархії класів. Класифікація об'єктів. Ієрархія класів плоских геометричних фігур. Абстрактний базовий клас, абстрактні методи. Класи "квадрат", "круг", "трикутник".	Лекція	2	
		Оголошення простої ієрархії "надклас-підклас" з віртуальними методами. Побудова та опрацювання поліморфної колекції (масиву) об'єктів. Уведення об'єктів з консолі, з файла, виведення на консоль і до файла. Надсилання поліморфних повідомлень.	Лабораторна робота	2	Наступне лабораторне заняття
	2.3	Ієрархія об'ємних геометричних фігур. Відношення "has-a", композиція об'єктів. Використання вкладених класів, підкласи для розширення можливостей надкласу.	Лекція	2	
		Оголошення розгалуженої ієрархії класів. Використання абстрактного базового класу. Віртуальні методи, віртуальні деструктори.	Лабораторна робота	2	Наступне лабораторне заняття
	2.4	Агрегація об'єктів. Проектування динамічного масиву об'ємних фігур. Оператори індексування. Питання власності агрегованих об'єктів. Глибоке копіювання контейнера. Бібліотека RTTI: перевірка типу об'єкта, приведення типу.	Лекція	2	
		Побудова типів на основі композиції класів (стандартних і власних). Використання власних (вкладених) класів винятків.	Лабораторна робота	2	Наступне лабораторне заняття
	2.5	Закрите наслідування - ще один спосіб моделювання "has-a". Синтаксис. Порівняння можливостей з композицією. Захищене наслідування.	Лекція	2	
		Побудова типів за допомогою закритого та захищеного наслідування. Використання власних (вкладених) класів винятків.	Лабораторна робота	2	Наступне лабораторне заняття
	2.6	Множинне наслідування. Випадки безпроблемного використання. Випадок спільного наднадкласу: віртуальні базові класи, конструктори, пам'ять об'єкта.	Лекція	2	
		Створення та використання класів-контейнерів. Використання засобів RTTI. Модульне тестування написаного коду.	Лабораторна робота	2	Наступне лабораторне заняття
	2.7	Узагальнення в програмуванні (змінна, функція, шаблон). Шаблон функції, параметризований типом даних: оголошення, використання, явна спеціалізація, перевантаження. Шаблон класу з параметром типом (стек): оголошення, використання, явна спеціалізація. Шаблон з параметром не типом.	Лекція	2	
		Створення класів за допомогою множинного наслідування. Модульне тестування написаного коду.	Лабораторна робота	2	Наступне лабораторне заняття
	2.8	Алгоритм пошуку значення в послідовності: узагальнення типу даних, структури послідовності, умови відшукання. Наслідування, включення, часткова спеціалізація. Шаблони - параметри шаблонів. Дружні конструкції.	Лекція	2	
		Побудова та використання шаблонів функцій, наприклад, для опрацювання числових масивів. Перевантаження шаблонів. Явна спеціалізація шаблонів для окремих типів даних.	Лабораторна робота	2	Наступне лабораторне заняття
	2.9	Стандартна бібліотека шаблонів (STL). Базові поняття: контейнер, ітератор, алгоритм, функтор. Класифікація ітераторів, моделі ітераторів. Класифікація контейнерів, спільні можливості.	Лекція	2	
		Побудова шаблонів класів, наприклад, для реалізації динамічних масивів. Використання зі стандартними та власними типами даних. Використання числових параметрів шаблону.	Лабораторна робота	2	Наступне лабораторне заняття

	2.10	Послідовні контейнери: концепція, влаштування, базові операції. Приклади використання контейнерів vector, list, deque. Порівняння ефективності послідовних контейнерів. Ітератори вставляння. Потокові ітератори.	Лекція	2	
		Побудова шаблонів класів, наприклад, для реалізації зв'язних списків. Створення і використання шаблона стека, параметром якого є шаблон послідовного контейнера (динамічного масиву чи зв'язного списку).	Лабораторна робота	2	Наступне лабораторне заняття
	2.11	Контейнерні адаптери stack, queue, priority_queue. Приклади використання: побудова польського запису, моделювання багатопотокового обслуговування черги клієнтів. Доступ до реалізації. Наслідування від стандартних контейнерів.	Лекція	2	
		Вивчення функціональності vector<T>: дописування, вставляння, вилучення, індексування, перебір, копіювання; дослідження типу. Використання у власних програмах.	Лабораторна робота	2	Наступне лабораторне заняття
	2.12	Використання класів характеристик для побудови шаблонів Класи, що налаштовують функціональність шаблона. Зміна характеристик класу string. Метапрограмування.	Лекція	2	
		Вивчення функціональності deque<T>, list<T>. Використання у власних програмах. Побудова класів, що включають послідовні контейнери. Побудова класів, що наслідують послідовні контейнери.	Лабораторна робота	2	Наступне лабораторне заняття
	2.13	Класифікація функторів. Об'єкт-функція. Стандартні функтори та функціональні адаптери. Створення власних функторів. Огляд бібліотеки алгоритмів. Приклади використання для написання "програм без циклів".	Лекція	2	
		Оголошення об'єкт-функції. Використання стандартних і власних функторів і стандартних алгоритмів для опрацювання послідовних контейнерів, файлів.	Лабораторна робота	2	Наступне лабораторне заняття
	2.14	Асоціативні контейнери. Спільні можливості, особливості реалізації. Різні способи конструювання set. Особливості використання map.	Лекція	2	
		Вивчення функціональності stack<T>, queue<T>, priority_queue<T>. Використання у власних програмах. Застосування різних реалізацій, дослідження функціонування.	Лабораторна робота	2	Наступне лабораторне заняття
	2.15	Побудова власних шаблонів контейнерних класів (на основі векторної та зв'язної пам'яті), створення ітераторів, налаштування "співпраці" зі стандартними алгоритмами. Побудова класів-адаптерів.	Лекція	2	
		Вивчення функціональності set<T, Comparer, Allocator>, multiset<T>. Використання у власних програмах. Вивчення функціональності map<TKey, TValue, Comparer>, multimap <TKey,TValue>. Використання у власних програмах.	Лабораторна робота	2	Наступне лабораторне заняття
	2.16	"Майже контейнери": valarray<Num>, auto_ptr<Pointer>, complex<Num>, pair<T1,T2>. Об'єктно-орієнтоване проектування комп'ютерної програми (іграшки-стрілялки) та реалізація в графічному середовищі Windows.	Лекція	2	
		Створення власного рядкового типу на основі basic_string<Char,Traits,Allocator>, приклади використання. Оголошення класів характеристик і політик для власних шаблонних класів. Використання у програмах.	Лабораторна робота	2	
Підсумковий контроль, форма	іспит в кінці кожного семестру				
Пререквізити	Для вивчення курсу студенти потребують базових знань з дисциплін “Математика”, “Інформатика” за курс середньої школи. Одночасно з вивченням програмування студенти проходять навчальну обчислювальну практику, впродовж якої виконують завдання з програмування, поглиблюють набуті в курсі знання та удосконалюють навики. Дисципліна “Програмування” також тісно пов’язана з курсом “Алгоритми і структури даних”, який вивчають у другому семестрі.				
Навчальні методи та техніки, які використовують під час викладання курсу	Лекції з мультимедійними презентаціями та з демонстрацією прийомів практичного використання середовища програмування; лабораторні заняття у вигляді проектування алгоритмів і програм, виконання практичних завдань, у тому числі у команді з 3-4 осіб; самостійне опрацювання навчальних матеріалів: підручників, конспектів лекцій, готових програм мовою C++, додаткових навчальних посібників, розміщених у хмарному сховищі (Moodle, Microsoft				

	Teams, Google Classroom). Обговорення теоретичного та практичного матеріалу в онлайн сервісах, формулювання творчих завдань для студентів, виконання яких готує до вивчення нового теоретичного матеріалу.
Необхідне обладнання	Для проведення лекцій: комп'ютер, проектор, доступ до мережі інтернет. Для проведення лабораторних та виконання завдань: комп'ютер, ОС Windows/Linux, доступ до інтернету, середовище програмування мовою C++ (Microsoft Visual Studio, Code Blocks тощо). Уся література, яку студенти не зможуть знайти самостійно, буде надана викладачем виключно в освітніх цілях без права її передачі третім особам. Студенти заохочуються до використання також й іншої літератури та джерел, яких немає серед рекомендованих.
Критерії оцінювання (окремо для кожного виду навчальної діяльності)	**Оцінювання** проводиться за 100-бальною шкалою. 50 балів нараховують за виконання лабораторних завдань і контрольних робіт, ще 50 балів – за виконання екзаменаційного завдання. Лабораторні завдання можуть бути індивідуальні та командні. Упродовж семестру студент виконує не менше 10 лабораторних робіт, кожну з яких оцінюють 4-6 балів залежно від складності. Для кожного завдання визначено термін виконання: зазвичай до закінчення навчального тижня. Вчасно виконані завдання оцінюють так (у відсотках від максимальної оцінки): • 100% – умови завдання виконано повністю, алгоритми складено правильно, програма містить належні коментарі, роботу програми перевірено на достатньому наборі тестових даних, автор відповідає на всі запитання щодо використаних підходів, чітко інтерпретує отримані результати, немає ознак недоброчесності; • 80% – наведено логічно правильну послідовність розв'язування, алгоритми складено правильно, бракує окремих коментарів чи тестів, автор не досить повно пояснює використані підходи, немає ознак недоброчесності; • 60% – у правильній послідовності розв'язування допущено окремі помилки, які автор уміє виправити після зауваження викладача, бракує коментарів чи тестів, на запитання щодо використаних підходів автор відповідає з помилками, немає ознак недоброчесності; • 40% – у правильній послідовності розв'язування пропущено окремі етапи, завдання виконано частково, автор не розуміє недоліків поданої роботи, не вміє їх виправити, немає ознак недоброчесності; • 20% – завдання виконано частково, немає тестів, програма працює правильно для окремих наборів вхідних даних, автор не може самостійно інтерпретувати отримані результати, виправити помилки, немає ознак недоброчесності; • 0% – завдання не виконано, написана програма не відповідає умові, або ж виявлено ознаки недоброчесності: запозичення, фрагменти коду, дію яких автор пояснити не може, автор не володіє відповідним теоретичним матеріалом тощо; • можуть бути нараховані додаткові бали за повністю виконане завдання, яке містить кілька способів розв'язування, використовує особливо ефективний спосіб, демонструє креативність автора тощо. Запізнення зменшує максимальну оцінку за завдання: кожного наступного після терміну виконання тижня оцінка зменшується удвічі. Оцінка за екзаменаційне завдання може бути поділена на дві частини: 20 балів за засвоєння теоретичного матеріалу, виставлені після опитувань упродовж семестру (у формі тестувань, колоквіумів, проектів тощо) та 30 за написання комп'ютерної програми (декількох програм). Розподіл балів за комп'ютерну програму наведено в екзаменаційному білеті, зразок такого білета студенти отримають на консультації перед іспитом. Додаткові бали будуть зараховані учасникам Всеукраїнської студентської олімпіади з програмування (відповідно до кількості розв'язаних задач). **Відвідання занять** є важливою складовою навчання. Очікується, що всі студенти відвідають усі лекції і лабораторні заняття курсу. Активність під час проведення лекцій і лабораторних заохочується балами. У будь-якому випадку студенти зобов'язані дотримуватися усіх строків визначених для виконання усіх видів письмових робіт, передбачених курсом. Виконані роботи завантажують у відповідне хмарне сховище. Альтернативою відвідування лабораторних занять в університеті може бути дистанційна онлайнова робота за розкладом проведення занять. Активність на лекціях і лабораторних ураховують при оцінюванні відповідного лабораторного завдання.

	Академічна доброчесність: очікується, що роботи студентів будуть їхнім оригінальними дослідженнями чи міркуваннями. Відсутність посилань на використані джерела, фабрикування джерел, списування, втручання в роботу інших студентів, здавання чужих комп'ютерних програм як своїх становлять, але не обмежують, приклади можливої академічної недоброчесності. Виявлення ознак академічної недоброчесності в письмовій роботі студента є підставою для її незарахування викладачем, незалежно від масштабів плагіату чи обману.
Опитування	Анкету-оцінку з метою оцінювання якості курсу буде надано після завершення курсу.

Запитання до іспитів

1 семестр

1. Структура програми мовою С++, порядок її компіляції.
2. Які типи в С++ використовують для зберігання значень цілого типу? Наведіть всеможливі приклади зображення цілого.
3. Які типи в С++ використовують для зберігання значень дійсного типу? Наведіть всеможливі приклади зображення дійсного.
4. Які типи в С++ використовують для зберігання літер? Наведіть всеможливі приклади зображення літер.
5. Вкажіть оператори С++, які застосовують до значень цілих типів (у порядку спадання пріоритету).
6. Вкажіть оператори С++, які застосовують до значень дійсних типів (у порядку спадання пріоритету).
7. Вкажіть оператори присвоєння С++, поясніть алгоритм їхнього виконання.
8. Який тип С++ використовують для зберігання логічних значень? Як зображають логічні значення? Які оператори до них застосовують?
9. Перелічіть стандартні математичні функції (з файлу cmath).
10. Наведіть приклад вкороченої умовної інструкції мови С++. Поясніть алгоритм її виконання.
11. Наведіть приклад повної умовної інструкції мови С++. Поясніть алгоритм її виконання.
12. Наведіть приклад інструкції switch мови С++. Поясніть алгоритм її виконання.
13. Наведіть приклад інструкції for мови С++. Поясніть алгоритм її виконання.
14. Наведіть приклад інструкції while мови С++. Поясніть алгоритм її виконання.
15. Наведіть приклад інструкції do мови С++. Поясніть алгоритм її виконання.
16. Наведіть приклад використання оператора ? : мови С++. Поясніть алгоритм його виконання.
17. Інструкція переходу мови С++: синтаксис, алгоритм виконання, випадки використання.
18. Інструкція break мови С++: синтаксис, алгоритм виконання, випадки використання (перелічіть усі).
19. Інструкція return мови С++: синтаксис, алгоритм виконання, випадки використання (перелічіть усі).
20. Загальна структура оголошення імені в С++.
21. Як оголосити іменовану константу мовою С++?
22. Що таке typedef у мові С++? Як його можна використати? Як використати using з тією ж метою?
23. Оголошення одновимірного масиву в С++. Ініціалізація. Звертання до елементів масиву.
24. Оголошення двовимірного масиву в С++. Ініціалізація. Звертання до елементів масиву.
25. Опишіть функцію мовою С++ для виведення на друк одновимірного числового масиву.
26. Наведіть фрагмент коду для виведення на друк двовимірного числового масиву.
27. Оголошення структури в мові С++. Для чого використовують структури?
28. Оператори над структурами мови С++. Поясніть, як вони діють.
29. Конструктори структур мови С++: синтаксис, призначення.
30. Що таке список ініціалізації (в конструкторі)? Який його синтаксис? Як він діє?
31. Що таке вказівник? Як його оголосити?
32. Вкажіть всі способи ініціалізації вказівника.
33. Як створити динамічну змінну простого типу? Як задати їй значення? Як її знищити?
34. Як створити одновимірний динамічний масив? Як його знищити?
35. Як ввести (вивести) значення одновимірного динамічного масиву?
36. Як створити двовимірний динамічний масив?
37. Які оператори можна застосувати до вказівника? Поясніть.
38. Який тип мови С++ можна використати для побудови лінійного однозв'язного списку?
39. Що потрібно для того, аби використати функцію в С++програмі?
40. Що таке прототип функції? Навіщо він потрібен? Наведіть приклад.
41. Які є два основні різновиди функцій у С++? Яка між ними різниця?
42. Які є способи передавання аргументів функціям?
43. Поясніть синтаксис і семантику передавання аргументів за значенням.
44. Поясніть синтаксис і семантику передавання аргументів за адресою.
45. Поясніть синтаксис і семантику передавання аргументів через посилання.
46. Використання посилань у параметрах функції. Переваги та недоліки.
47. Як виконують виклик функції у С++? Наведіть приклади.
48. Що таке локальні змінні функції? (Оголошення, використання, видимість, час існування, приклади).
49. Що таке статичні локальні змінні функції? (Оголошення, використання, видимість, час існування, приклади).
50. Передавання одновимірного масиву функції (з указанням розміру). Пояснити на прикладі.

51. Передавання одновимірного масиву функції (з указанням діапазону). Пояснити на прикладі.
52. Опрацювання структур функціями (передавання всередину, повернення з, приклад).
53. Функції C++ та рекурсія. Навести приклад.
54. Тип «вказівник на функцію». Використання в тілі функції (наприклад, main).
55. Тип «вказівник на функцію». Використання в параметрах функції.
56. Лямбда вирази: синтаксис, призначення, приклад використання.
57. Аргументи за замовчуванням функцій C++.
58. Перевантаження функцій C++.
59. Заголовковий файл: призначення, використання, рекомендації щодо вмісту.
60. Що таке файловий потік? Як його використовують?
61. Як прочитати вхідні дані з текстового файлу?
62. Як зберегти результати до текстового файлу?
63. Як прочитати вхідні дані з двійкового файлу?
64. Як зберегти результати до двійкового файлу?
65. Що таке *клас*, *об'єкт*?
66. Що таке *поле даних*, *змінна класу*?
67. Що таке *метод (екземпляра)*, *метод класу*?
68. Що таке *конструктор*, *деструктор*?
69. Які різновиди конструкторів (у C++) Ви знаєте?
70. Які завдання виконує деструктор? Чи може він бути віртуальним?
71. У яких випадках використовується конструктор за замовчуванням? Наведіть його синтаксис.
72. У яких випадках використовується конструктор визначення? Наведіть його синтаксис.
73. У яких випадках використовується конструктор копіювання? Наведіть його синтаксис.
74. У яких випадках перевизначають конструктор копіювання? Наведіть його синтаксис.
75. У яких випадках перевизначають оператор присвоєння? Наведіть його синтаксис.
76. У яких випадках перевизначають деструктор? Наведіть його синтаксис.
77. Які члени класу генеруються автоматично? У яких випадках автоматичні члени класу не забезпечують потрібної функціональності?
78. Керування доступом до елементів класу C++. Правила видимості елементів класу.
79. Оголошення класу і визначення методів класу. Розташування програмного коду у файлах.
80. Діапазон видимості класу. Оголошення в класі. Доступ до членів класу.
81. Навіщо перевантажувати оператори? Синтаксис прототипу методу, що перевантажує оператор. Як компілятор опрацьовує вираз з оператором?
82. Вкажіть три головні способи перевантаження оператора.
83. Що таке дружня конструкція? Наведіть приклад використання.
84. У яких випадках оператор (для роботи з класом) можна перевантажити тільки дружньою функцією? Наведіть короткий приклад.
85. У яких випадках оператор (для роботи з класом) можна перевантажити зовнішньою функцією (без статусу дружньої)? Наведіть короткий приклад.
86. Що таке модульне тестування? Для чого його застосовують? Які інструменти використовують?
87. Як в одному рішенні об'єднати визначення класу (чи набору функцій для роботи з певним типом даних), програму, що його використовує, та модульні тести? Поясніть на прикладі, опишіть послідовні кроки конструювання такого рішення.
88. Які засоби модульного тестування фреймворку від Microsoft ви знаєте? Опишіть їхнє призначення, наведіть приклади використання.
89. Як за допомогою модульних тестів перевірити, чи генерує тестований код винятки? Опишіть хоча б два способи.
90. Алгоритм відшукання найбільшого з трьох заданих чисел.
91. Алгоритм обчислення суми значень числового масиву.
92. Алгоритм відшукання найбільшого значення числового масиву.
93. Алгоритм переведення цілого числа до нової системи числення.
94. Алгоритм перевірки впорядкованості числового масиву.
95. Алгоритм вставки числа у впорядкований масив.
96. Алгоритм злиття двох впорядкованих масивів.
97. Алгоритм впорядкування масиву методом обміну (бульбашки).
98. Алгоритм впорядкування масиву методом вибору (найбільшого чи найменшого).
99. Алгоритм впорядкування масиву методом простої вставки.
100. Алгоритм побудови лінійного однозв'язного списку за заданою послідовністю чисел.
101. Алгоритм вилучення з пам'яті лінійного однозв'язного списку.
102. Алгоритм вставки числа у впорядкований лінійний однозв'язний список.
103. Алгоритм злиття двох впорядкованих списків чисел.
104. Алгоритм розв'язування алгебричного рівняння методом дихотомії (ітеративна реалізація).
105. Алгоритм розв'язування алгебричного рівняння методом дихотомії (рекурсивна реалізація).

2 семестр

1. Які є способи передавання аргументів функціям?
2. Поясніть синтаксис і семантику передавання аргументів за значенням.

3. Поясніть синтаксис і семантику передавання аргументів за адресою.
4. Поясніть синтаксис і семантику передавання аргументів через посилання.
5. Що таке локальні змінні, статичні локальні змінні функції? (Оголошення, використання, видимість, час існування, приклади).
6. Передавання одновимірного масиву функції (з указанням розміру, з указанням діапазону). Пояснити на прикладі.
7. Що таке *клас*, *об'єкт*? *поле даних*, *змінна класу*? *метод (екземпляра)*, *метод класу*?
8. Що таке *конструктор*, *деструктор*?
9. Які різновиди конструкторів (у C++) Ви знаєте?
10. Які завдання виконує деструктор? Чи може він бути віртуальним?
11. У яких випадках використовується конструктор за замовчуванням? Наведіть його синтаксис.
12. У яких випадках використовується конструктор визначення? Наведіть його синтаксис.
13. У яких випадках використовується конструктор копіювання? Наведіть його синтаксис.
14. У яких випадках перевизначають конструктор копіювання? Наведіть його синтаксис.
15. У яких випадках перевизначають оператор присвоєння? Наведіть його синтаксис.
16. У яких випадках перевизначають деструктор? Наведіть його синтаксис.
17. Які члени класу генеруються автоматично? У яких випадках автоматичні члени класу не забезпечують потрібної функціональності?
18. Керування доступом до елементів класу C++. Правила видимості елементів класу.
19. Оголошення класу і визначення методів класу. Розташування програмного коду у файлах.
20. Діапазон видимості класу. Оголошення в класі. Доступ до членів класу.
21. Статичні поля класу. Оголошення, ініціалізація, використання.
22. Статичні методи. Оголошення, визначення, використання.
23. Навіщо перевантажувати оператори? Синтаксис прототипу методу, що перевантажує оператор. Як компілятор опрацьовує вираз з оператором?
24. Вкажіть три головні способи перевантаження оператора.
25. Що таке дружня конструкція? Наведіть приклад використання.
26. У яких випадках оператор (для роботи з класом) можна перевантажити тільки дружньою функцією? Наведіть короткий приклад.
27. У яких випадках оператор (для роботи з класом) можна перевантажити зовнішньою функцією (без статусу дружньої)? Наведіть короткий приклад.
28. Перетворення типів: «клас → вбудований тип» і «вбудований тип → клас».
29. Відкрите наслідування: синтаксис, призначення, видимість, що моделює.
30. Закрите наслідування: синтаксис, призначення, видимість, що моделює.
31. Захищене наслідування: синтаксис, призначення, видимість, що моделює.
32. Множинне наслідування: синтаксис, призначення, безпроблемне використання.
33. Множинне Наслідування зі споріднених класів, віртуальні батьківські класи.
34. Включення об'єкта (об'єктів) до складу іншого об'єкта: синтаксис, можливості моделювання, використання.
35. Синтаксис конструктора підкласу. Порядок виконання його коду.
36. Призначення абстрактного класу. Синтаксис абстрактного методу.
37. Призначення і синтаксис віртуального методу. Порядок виклику віртуального методу.
38. Сумісність за присвоєнням «надклас-підклас», «надклас*-підклас*», «надклас&-підклас&».
39. Перевірка та динамічне перетворення типу об'єкта.
40. Відмінності у синтаксисі та можливостях використання включення і закритого наслідування.
41. У яких випадках використовують шаблони функцій (у C++)? Наведіть приклад оголошення.
42. Явне та неявне створення екземплярів шаблону функції (у C++).
43. Явна спеціалізація шаблону функції (у C++).
44. Що таке шаблон класу (у C++)? У яких випадках його використовують?
45. Оголошення та використання шаблона класу з параметром типом.
46. Оголошення та використання шаблона класу з параметром «не типом».
47. У яких випадках компілятор за шаблоном класу генерує визначення класу? Наведіть приклади.
48. Використання шаблона класу в якості аргументу шаблона, рекурсивне використання.
49. Спеціалізації шаблона класу (у C++).
50. Використання шаблона класу в якості базового (у C++). Наведіть приклади.
51. Використання шаблона класу в якості компоненти (у C++). наведіть приклади.
52. Як узагальнюють алгоритми опрацювання контейнерів (у C++)?
53. Використання класів характеристик, політик для оголошення шаблона класу (у C++).
54. Головні (чотири) групи шаблонів, що складають бібліотеку STL.
55. Що таке ітератор? Для чого його використовують? Де оголошують? Класифікація ітераторів.
56. Що таке функтор? Для чого його використовують? Класифікація функторів.
57. Що таке об'єкт-функція? Наведіть приклад оголошення і використання.
58. Охарактеризуйте стандартні функтори.
59. Алгоритми заповнення інтервалів і генерування значень, алгоритми підрахунку.
60. Алгоритми копіювання та перестановок.
61. Алгоритми пошуку та заміни.
62. Алгоритми порівняння та вилучення.

63. Алгоритми впорядкування та споріднені дії.
64. Застосування дії до кожного елемента контейнера.
65. Спільні риси послідовних контейнерів.
66. Ітератори вставляння для послідовних контейнерів.
67. Потокові ітератори (оголошення, приклади використання).
68. Контейнер вектор (оголошення, функціональні можливості, приклади використання).
69. Контейнер список (оголошення, функціональні можливості, приклади використання).
70. Контейнер дек (оголошення, функціональні можливості, приклади використання).
71. Контейнер однозв'язний список (оголошення, можливості, приклади використання).
72. Контейнер стек (оголошення, можливості, приклади використання, доступ до реалізації).
73. Контейнер черга (оголошення, можливості, приклади використання, доступ до реалізації).
74. Контейнер пріоритетна черга (оголошення, можливості, приклади використання, доступ до реалізації, роль предиката в оголошенні та функціонування).
75. Спільні риси асоціативних контейнерів.
76. Контейнери множина, мультимножина (оголошення, можливості, приклади використання, роль предиката в оголошенні та функціонування).
77. Контейнери відображення, мультивідображення (оголошення, можливості, приклади використання, роль предиката в оголошенні та функціонування, особливості ітерування).
78. Клас string (влаштування, споріднені шаблони, можливості, приклади використання).
79. Числові масиви (оголошення, можливості, приклади використання).
80. Клас complex (оголошення, можливості, приклади використання).

Вміння писати програми:

1. Оголосити клас
   a. правильно використати режими видимості;
   b. правильно розташувати код у файлах;
   c. уміти оголосити конструктори (різні), деструктор, розуміти, коли вони працюють;
   d. уміти перевантажувати оператори (присвоєння, порівняння, арифметичні, введення/виведення) за допомогою методів, зовнішніх функцій, дружніх функцій;
   e. вміти створювати статичні та динамічні екземпляри класу, надсилати їм повідомлення;
   f. вміти створювати масиви об'єктів, завантажувати їх з файлу, зберігати до файлу.
2. Оголосити підклас (підкласи) за допомогою відкритого наслідування
   a. правильно оголошувати конструктори підкласу (з викликом конструкторів базового класу), деструктор;
   b. перевизначати методи (віртуальні), використовувати наявні, визначати нові;
   c. розуміти особливості доступу до членів базового класу з різною видимістю;
   d. перевизначати оператори для ієрархії класів, зокрема, оператори введення і виведення;
   e. розуміти та вміти використовувати розширені правила сумісності споріднених типів для передавання у функції (методи), для створення поліморфних колекцій;
   f. розуміти як працюють та вміти використовувати поліморфні повідомлення;
   g. розуміти призначення і оголошення абстрактних класів, уміти їх використовувати для побудови ієрархії типів.
3. Використовувати композицію та/або агрегацію об'єктів
   a. уміти включати об'єкт(и) до складу іншого;
   b. використовувати закрите та захищене наслідування для реалізації включення;
   c. проектувати та використовувати власні контейнери.
4. Використовувати узагальнені функції
   a. оголошення шаблона функції, використання, коли та як відбувається створення екземпляра функції;
   b. спеціалізації (повні та часткові шаблона функції) для певного типу;
   c. уміти оголосити та використати шаблон функції, оголосити його явну спеціалізацію.
5. Узагальнені типи
   a. оголосити та використати шаблон класу, параметризований типом та/або параметром-нетипом; наслідувати шаблони;
   b. вміти застосовувати явну спеціалізацію шаблонів;
   c. розуміти, коли та як відбувається створення оголошення класу за його шаблоном;
   d. використовувати характеристики класів для створення шаблонів класів.
6. Бібліотека STL
   a. розуміти, для яких задач які типи контейнерів підходять найкраще; уміти підібрати контейнер за його властивостями;
   b. уміти використовувати послідовні контейнери для зберігання колекцій значень, уміти їх перебирати, знаходити потрібне, змінювати, додавати/вилучати елементи, з'ясовувати загальні властивості контейнера;
   c. уміти використовувати асоціативні контейнери для зберігання колекцій унікальних значень, відображень, уміти їх перебирати, знаходити потрібне, змінювати, додавати/вилучати елементи, з'ясовувати загальні властивості контейнера;
   d. уміти виконувати перетворення контейнерів одного типу до іншого, завантажувати їх з файлу, зберігати до файлу;
   e. уміти використовувати стек, чергу, пріоритетну чергу за безпосереднім призначенням;

f. уміти опрацьовувати текстову інформацію за допомогою рядкових контейнерів;
g. уміти використовувати стандартні алгоритми та ітератори для опрацювання контейнерів, взаємодії з потоками тощо (копіювання, трансформації, впорядкування, дії з множинами, пошук, генерування та заповнення інтервалів, заміни, перестановки, вилучення, злиття, числові алгоритми; ітератори контейнерів, потокові, вставляння);
h. уміти оголошувати об’єкт-функції та лямбда-вирази і використовувати їх для налаштування алгоритмів на необхідні перетворення, чи умови, використовувати для оголошення контейнерів тощо.