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 та інформатики Львівського національного університету імені Івана Франка

(протокол № 1 від 29 серпня 2023 р.)

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

Силабус навчальної дисципліни

“Програмування” ,

що викладається в межах ОПП Кібербезпека

першого (бакалаврського) рівня вищої освіти

для здобувачів зі спеціальності 125 – Кібербезпека та захист інформації

Львів – 2023

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	12 – інформаційні технології 125 – кібербезпека та захист інформації
Викладачі дисципліни	Музичук Анатолій Омелянович, к. ф.-м. н., доц., доцент кафедри програмування Клакович Леся Миронівна, к. ф.-м. н., доцент, доцент кафедри програмування
Контактна інформація викладачів	Електронна пошта: lesya.klakovych@lnu.edu.ua веб-сторінка: https://ami.lnu.edu.ua/employee/klakovych Електронна пошта: anatoliy.muzychuk@lnu.edu.ua Веб-сторінка: https://ami.lnu.edu.ua/employee/muzychuk
Консультації з питань навчання по дисципліні відбуваються	Один раз на тиждень згідно з оприлюдненим розкладом консультацій викладача. Можливі онлайн-консультації в середовищі Microsoft Teams. Для погодження часу онлайн-консультацій 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 125 – кібербезпека для освітньої програми Кібербезпека, яка викладається в 1-3-іх семестрах в обсязі 12-ти кредитів (за Європейською Кредитно-Трансферною Системою ECTS).
Коротка анотація дисципліни	Фокус уваги курсу спрямовано на розробку складних та високопродуктивних комп’ютерних програм. Одним з предметів вивчення і, водночас, основним засобом навчання є мова програмування С++. Перший семестр навчання закладає базу: дає основи алгоритмізації та вступ до об’єктно-орієнтованого програмування. У другому семестрі вивчають тонкощі ООП та узагальнене

	програмування. Третій семестр стосується вивчення ООП засобами мови Python, зокрема виконання ефективних матрично-векторних обчислень (б-ка NumPy), візуалізації даних (б-ка matplotlib), опрацювання табличних даних (б-ка Pandas), тестування (б-ка unittest) та інтерактивного комп'ютингу (б-ка widgets).
Мета та цілі дисципліни	Мета – формування компетенцій, необхідних для ефективного використання сучасних технологій розробки програмного забезпечення для вирішення прикладних та наукових задач. Цілі: ознайомлення з сучасними технологіями опрацювання та візуалізації даних засобами бібліотек мов C++ та Python, оволодіння навичками розробки та програмування алгоритмів розв'язання прикладних задач у імперативному, процедурному та об'єктно-орієнтованому стилях.
Література для вивчення дисципліни	*Основна:* 1. Stephen Prata. C++ Primer Plus (Developer's Library). Addison-Wesley Professional; 6th edition (October 18, 2021), – 1440 p. 2. Ярошко С.А. Методи розробки алгоритмів. Програмування мовою С++: Навчальний посібник / С.А. Ярошко, О.С. Ярошко – Львів: ЛНУ імені Івана Франка, 2022. – 248 с. [електронна версія: https://lnuittutor.github.io/] 3. Bruce Eckel. Thinking in C++, Vol. 1: Introduction to Standard C++, 2nd Edition. Prentice Hall; (March 25, 2020), 840 p. 4. Bjarne Stroustrup.The C++ Programming Language. Addison-Wesley Professional; 4th edition, 2019 – 1376 p. 5. Маттес Е. Пришвидшений курс Python. – Львів : ВСЛ, 2021. 6. Селіверстов Р., Мельничин А. Основи програмування мовою Python: навч. посібник. – Львів : ЛНУ імені Івана Франка, 2020. 7. The Python Tutorial. – https://docs.python.org/3/tutorial/index.html 8. Lutz M. Learning Python, 5th Edition. – O'Reilly Media, 2013. 9. Lambert K. A. Fundamentals of Python: First Programs, 2nd Edition. – Cengage, 2019. 10. NumPy. – http://numpy.org . 11. Pandas. – http://pandas.pydata.org . 12. Matplotlib. – http://matplotlib.org . *Додаткова:* 13. Дудзяний І.М. Програмування мовою С++. Частина 1: Парадигма процедурного програмування: навчальний посібник / І.М. Дудзяний. – Львів: ЛНУ імені Івана Франка, 2013. – 468 с. 14. https://www.learncpp.com/ 15. https://en.cppreference.com/w/ 16. https://cplusplus.com/ 17. Nicolai M. Josuttis. C++ Standard Library, The: A Tutorial and Reference. Addison-Wesley Professional; 2nd edition (March 30, 2019) – 1136 p.Wentworth P., Elkner J., Downey A., Meyers C. How to Think Like a Computer Scientist: Learning with Python 3. – Green Tea Press, 2018. 18. Python Tricks: The Book. — Dan Bader, 2017. 19. Sweigart A. Automate the Boring Stuff with Python: Practical Programming for Total Beginners. – No Starch Press, 2014. 20. McKinney W. Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython, 2nd Edition. – O'Reilly Media, 2018.

	21. Mark J. Price C# 10 and .NET 6 – Modern Cross-Platform Development – Packt Publishing, 2021 – 826 pp. 22. Joseph Albahari C# 10 in a Nutshell: The Definitive Reference – O'Reilly Media, 2022 – 1000 pp. 23. Ian Griffiths C# 10 in a Nutshell – O'Reilly Media, 2022 – 833 pp.
Обсяг курсу	12 кредитів ЄКТС – 360 годин. З них 32*3 годин лекцій, 32*3 годин лабораторних занять та 56*3 годин самостійної роботи (1, 2 і 3-й семестри).
Очікувані результати навчання	Після завершення цього курсу студент буде *знати:* – синтаксис, основні конструкції та типи даних мов C++ та Python; – принципи імперативного, процедурного, функціонального та об'єктно-орієнтованого програмування мовами C++ та Python; – сучасні технології розробки програм для ефективних обчислень на різних платформах; – сучасні програмні засоби графічного подання даних, організації та опрацювання табличних даних засобами ефективних колекцій. *вміти:* – застосовувати сучасні технології ООП для розробки програмного забезпечення у різних предметних областях; – працювати з потоками даних різної структури на різних пристроях з використанням різних бібліотек ; – реалізовувати фундаментальні патерни проектування програмного забезпечення; – створювати багатомодульні програми; – розробляти графічні інтерфейси для інтерактивного комп'ютингу; – забезпечувати відповідну якість програмного забезпечення за допомогою різних фреймворків модульного тестування.
Компетентності	*Інтегральна: ІК* *Загальні: КЗ 1, КЗ 3, КЗ 5* *Фахові компетентності спеціальності: КФ 3*
Програмні результати навчання	ПРН 1-4, ПРН 6, ПРН 14,15, ПРН 18, 20.
Ключові слова	алгоритм, тип даних, інструкція, функція, об'єкт, вказівник, посилання, клас, поліморфізм, наслідування, інкапсуляція, STL, NumPy, matplotlib, Pandas, widgets, unittest, TDD, Test fixture, Class fixture, FluentAssertions.
Формат курсу	Очний: проведення лекцій, лабораторних робіт та консультацій в приміщеннях університету (в умовах карантину – онлайновий на платформі Microsoft Teams)

Підсумковий контроль, форма	іспит у кінці кожного семестру
Пререквізити	Для вивчення курсу студенти потребують знань з попередніх семестрів навчання: основи ООП, використання винятків, взаємодія з потоками, використання контейнерів, модульне тестування. Одночасно з вивченням програмування студенти проходять навчальну обчислювальну практику, впродовж якої виконують завдання з програмування, поглиблюють набуті в курсі знання та удосконалюють навички в розробці ефективного програмного забезпечення на різних платформах.
Навчальні методи та техніки, які будуть використовуватися під час викладання курсу	Лекції з мультимедійними презентаціями; лабораторні заняття у вигляді виконання практичних завдань (у тому числі командних); самостійне опрацювання навчальних матеріалів, розміщених у хмарних сховищах (Moodle, Microsoft Teams); обговорення тем та консультації в середовищах Microsoft Teams, Slack, Skype тощо.
Необхідне обладнання	Для проведення лекцій: комп'ютер, проєктор, доступ до мережі Інтернет. Для проведення лабораторних занять та виконання завдань: комп'ютер, ОС Windows/Linux, доступ до інтернету.
Критерії оцінювання (окремо для кожного виду навчальної діяльності)	Оцінювання проводиться за 100-бальною шкалою у кожному семестрі. 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 3-5 балів залежно від складності. Оцінка за екзаменаційне завдання може бути поділена на дві частини: 20 балів за засвоєння теоретичного матеріалу, виставлені після опитувань впродовж семестру (у формі тестувань, колоквіумів тощо) та 30 за написання комп'ютерної програми (декількох програм). Відвідування занять є важливою складовою навчання. Очікується, що всі студенти відвідують 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 враховують 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 по завершенню курсу.
Питання на іспит	Семестр 1: 1. Тип char – для чого використовується, які операції з даними цього типу можна виконувати 2. Тип double - для чого використовується, які операції з даними цього типу можна виконувати? 3. Чим відрізняється тип float від типу double? 4. Тип string - для чого використовується, які операції з даними цього типу можна виконувати? 5. Чим відрізняється тип char від типу string? 6. Тип int - для чого використовується, які операції з даними цього типу можна виконувати? 7. Тип unsigned - для чого використовується, які операції з даними цього типу можна виконувати 8. Типи даних з long і short - для чого використовується 9. Масиви статичні і динамічні, відмінності 10. Цикл for, коли використовувати 11. Цикл while, коли використовувати 12. Цикл do while, коли використовувати 13. Оператор switch 14. Оператор new – функціональність 15. Оператор delete 16. Що таке вказівник? 17. Вказівник на void 18. Як можна передати дані у функцію? 19. Параметри-значення і параметри-посилання – відмінності 20. Глобальні та локальні змінні - відмінності 21. Як передати (повернути) масив у функцію? 22. Функція певного типу і функція типу void 23. Оператор return – для чого використовується і яка в нього дія 24. Адреса змінної – що це таке і який тип даних може її зберігати 25. Операції з посиланнями 26. Операції з вказівниками 27. Що таке перелічення, як його визначити 28. Що таке struct? Для чого використовується

	29. Що таке class? Для чого використовується 30. Що таке об’єкт класу, як його створити? 31. Що таке конструктори класу, для чого використовуються 32. Хто викликає конструктори класу? 33. Для чого потрібний дефолтний конструктор 34. Для чого потрібний конструктор копіювання 35. Що таке деструктори класу 36. Чим відрізняються private і public члени класу? 37. Чим відрізняються private і protected члени класу? 38. Що таке поля класу і як їх визначити? 39. Що таке методи класу, для чого вони використовуються 40. Як перевантажити оператори для класу 41. Перевантаження операторів через методи і глобальні функції, відмінності 42. Дружні функції 43. Для чого перевантажувати оператор присвоєння 44. Як перевантажувати арифметичні операції 45. Як перевантажувати інкремент та декремент 46. Як перевантажувати оператори порівняння **Семестр 2.** 1. Що таке перевантаження операторів в C++? 2. Які оператори можна перевантажувати в класах? 3. Які методи класу викликаються при перевантаженні операторів? 4. Які переваги має перевантаження операторів? 5. Що таке наслідування класів? 6. Які типи наслідування існують в C++? 7. Що таке базовий клас і похідний клас? 8. Які методи базового класу доступні у похідному класі? 9. Що таке композиція класів? 10. Яка різниця між композицією та наслідуванням? 11. Які переваги має використання композиції? 12. Що таке множинне наслідування? 13. Які проблеми можуть виникнути при використанні множинного наслідування? 14. Яким чином вирішується проблема "алмаза наслідування"? 15. Що таке шаблони в C++? 16. Які переваги мають шаблони в програмуванні? 17. Як створити шаблонний клас чи функцію? 18. Що таке виняткова ситуація? 19. Як використовувати блок try-catch для обробки виняткових ситуацій? 20. Як створити власний клас виняткової ситуації? 21. Які маніпулятори в C++ використовуються для форматного виведення? 22. Як використовувати маніпулятори для вирівнювання виводу? 23. Як створити власний маніпулятор? 24. Які основні компоненти входять до складу бібліотеки STL? 25. Які контейнери даних надає STL? 26. Яка різниця між контейнерами vector і list? 27. Які операції можна виконувати з векторами в STL?

	28. Які операції можна виконувати зі списками в STL? 29. Які контейнери STL використовуються для реалізації стеку та черги? 30. Які переваги має використання бібліотеки STL у програмуванні? 31. Які контейнери в STL використовуються для реалізації хеш-таблиці? 32. Які контейнери в STL можна використовувати для зберігання унікальних елементів без сортування? 33. Що таке адаптери контейнерів в STL і як вони використовуються? 34. Які контейнери в STL підтримують стекову (LIFO) структуру даних? 35. Які контейнери в STL підтримують чергову (FIFO) структуру даних? 36. Що таке ітератори в STL і яка їхня роль? 37. Які типи ітераторів існують в STL? 38. Яким чином можна перебрати всі елементи контейнера за допомогою ітератора? 39. Як можна отримати ітератор на початок або кінець контейнера? 40. Як можна перевірити, чи є ітератор дійсним (вказує на правильну позицію в контейнері)? 41. Що таке інкремент та декремент ітератора? 42. Які типи ітераторів підтримують зміну значень елементів контейнера? 43. Як можна пересунутись до певного віддаленого елемента за допомогою ітератора? 44. Як можна отримати значення, на яке вказує ітератор, або змінити значення елемента за допомогою ітератора? 45. Які алгоритми обробки даних надає STL? 46. Як виконати сортування вектору за допомогою алгоритму sort? 47. Як виконати знаходження елемента в контейнері за допомогою алгоритму find? 48. Що таке функціональні об'єкти (functors) в STL і як вони використовуються? 49. Як виконати видалення дублікатів з контейнера за допомогою алгоритму unique? 50. Які функції маніпулювання розміром контейнера є в STL? 51. Як створити власний функціональний об'єкт для використання в алгоритмах STL? 52. Які алгоритми STL підтримують умовне сортування? 53. Які алгоритми STL використовуються для сортування в порядку спадання? 54. Які алгоритми STL можна використовувати для об'єднання двох відсортованих контейнерів? 55. Які функції можна використовувати для зчитування та запису даних у файли за допомогою STL? 56. Які алгоритми STL можна використовувати для перетину двох відсортованих контейнерів? 57. Які алгоритми STL можна використовувати для обчислення суми або середнього значення елементів вектору 58. Яка основна відмінність між асоціативними контейнерами std::map і std::multimap? set та multiset? 59. Які методи можна використовувати для вставки елементів в асоціативний контейнер? 60. Які алгоритми STL можна використовувати для пошуку елемента в

	асоціативному контейнері? 61. Які контейнери в STL підтримують впорядкований доступ до елементів за допомогою ключа? 62. Які контейнери в STL підтримують швидкий доступ до елементів за допомогою хеш-функції? 63. Які методи можна використовувати для вилучення елементів з асоціативного контейнера? 64. Які алгоритми STL можна використовувати для сортування асоціативного контейнера за ключем? 65. Які методи можна використовувати для перевірки наявності елемента в асоціативному контейнері? 66. Які контейнери в STL підтримують збереження унікальних ключів без сортування? Семестр 3. 1. Базові типи Python. 2. Особливості та наслідки динамічної типізації 3. Операції над числовими типами. 4. Операції над рядками. 5. Взаємодія з файлами. 6. Колекції об’єктів. 7. Умовні інструкції. 8. Цикли. 9. Оголошення і виклик функцій. 10. Області видимості. 11. Режими співставлення аргументів. 12. Анонімні функції. 13. Оперування функціями як об’єктами. 14. Засоби функціонального програмування в Python. 15. Модулі та пакети, операції імпорту. 16. Класи і екземпляри. 17. Модифікатори доступу. 18. Перевантаження операторів. 19. Наслідування та композиція. 20. Ітератори та генератори. 21. Обробка винятків та тестування коду. 22. Декораторування функцій і класів. 23. Векторно-матричне числення в NumPy. 24. Первинний статистичний аналіз і візуалізація даних. 25. Віджети та обробка подій у графічних інтерфейсах.

Схема курсу

Тижд.	Тема, план, короткі тези	Форма заняття	Тривалість, год	Термін виконання

1.1	Що таке "комп'ютер"? Кодування інформації. Мови програмування. Структура програми мовою C++, 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
	Додаткові теми стосовно вісімкової та шістнадцяткової систем числення	Самостійна робота	3	
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	
	Файли стандартної бібліотеки	Самостійна робота	3	
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
	Додаткові теми щодо інструкцій галуження та їх використання	Самостійна робота	3	
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
	Додаткові теми щодо складених типів даних	Самостійна робота	3	
1.5	Вказівники, арифметика вказівників, застосування до масивів. Динамічні змінні: створення, використання, знищення. Введення-виведення масивів. Робота з двохвимірними	Лекція	2	

	масивами. Побудова лінійного списку за допомогою вказівників.			
	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
	Додаткові теми щодо динамічної пам’яті та роботи з нею	Самостійна робота	3	
1.6	Модульне програмування. Функції в C++: прототип, визначення, виклик. Параметри-значення, параметри-вказівники, Вказівники і специфікатор const. Тип посилання, використання в параметрах функцій.	Лекція	2	
	Поєднання цикла з галуженням. Вкладені цикли (у матричних задачах). Контрольна робота "Основи алгоритмізації"	Контрольна робота	2	
	Рекурсивні функції	Самостійна робота	3	
1.7	Функції і масиви. Впорядкування елементів масиву.	Лекція	2	
	Створення, використання, знищення динамічних одно- та двохвимірних масивів (у тому числі масивів літер). Оголошення функцій для обчислень виразів, для роботи з масивами.	Лабораторна робота	2	Наступне лабораторне заняття
	Вказівник на функцію, масиви функцій, функції вищих порядків. Використання лямбда-виразів.	Самостійна робота	3	
1.8	Структуровані типи даних C++: структури, бітові поля, об'єднання. Конструктори, деструктори структур, селектор імені. Перевантаження операторів: арифметичних, введення-виведення.	Лекція	2	
	Оголошення і використання функцій, що змінюють свої аргументи: параметри-вказівники та парамети-посилання. Використання вказівника на функцію: алгоритми числового інтегрування, наближене розв'язування рівнянь.	Лабораторна робота	2	Наступне лабораторне заняття
	Функції для опрацювання структур.	Самостійна робота	3	
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.	Лекція	2	
	Оголошення і використання типу struct: конструктори, оператори введення-виведення. Моделювання одно- та двохзв'язного списку. Функція побудови списку, виведення, пошуку значення тощо.	Лабораторна робота	2	Наступне лабораторне заняття
	Сортування списком. Алгоритм злиття впорядкованих послідовностей.	Самостійна робота	3	
1.10	Структура даних дерево. Поняття рекурсії. Рекурсивні розв’язки, рекурсивні функції.	Лекція	2	

	Моделювання двійкового дерева, дерева пошуку. Рекурсивні та нерекурсивні алгоритми обходу дерева.	Лабораторна робота	2	Наступне лабораторне заняття
	Алгоритми обходу двійкового дерева. Дерева пошуку. Сортування деревом. Збалансовані дерева, вставка, вилучення.	Самостійна робота	3	
1.11	Введення/виведення даних і використання файлів у C++. Потоки, буфери, файли.	Лекція	2	
	Впорядкування масиву. Впорядкування списку. Застосування різних алгоритмів. Перевірка правильності виконання.	Лабораторна робота	2	Наступне лабораторне заняття
	Робота з багатьма файлами. Текстові та двійкові файли.	Самостійна робота	3	
1.12	Аргументи функцій за замовчуванням. Статичні змінні функції. Поліморфізм і перевантаження функцій. Вбудовані функції. .	Лекція	2	
	Завантаження колекції значень з файла. Зберігання даних до файла. Функції з параметрами за замовчуванням.	Лабораторна робота	2	Наступне лабораторне заняття
	Оголошення та використання просторів імен. Класи пам'яті	Самостійна робота	3	
1.13	Клас - тип, оголошений користувачем. Термінологія. Прихована і доступна частини класу. Конструктори: за замовчуванням, з параметрами.	Лекція	2	
	Оголошення простору імен, розташування в ньому типів, функцій. Використання. Контрольна робота "Модульне програмування"	Контрольна робота	2	
	Клас "Пакет акцій". Розташування програмного коду у файлах.	Самостійна робота	3	
1.14	Оголошення класу і визначення методів класу. Використання дружніх функцій. Перевантаження операторів: різні способи. Перетворення типів: конструктори, оператори.	Лекція	2	
	Оголошення та використання класу. Конструктори з параметрами та за замовчуванням. Перевантаження операторів.	Лабораторна робота	2	Наступне лабораторне заняття
	Клас "Time"	Самостійна робота	3	
1.15	Обробка помилок за допомогою винятків: захищений блок, перехоплення, запуск, повторний запуск винятків. Розгортання стека.	Лекція	2	
	Об'єкти, що використовують рядки. Визначення конструктора копіювання, перевизначення оператора присвоєння.	Лабораторна робота	2	Наступне лабораторне заняття
	Стандартні класи винятків C++. Типи винятків, оголошені користувачем	Самостійна робота	3	
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	

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.	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
	Перевизначення методів, поліморфізм поведінки. Віртуальні методи та пізнє зв'язування.	Самостійна робота	3	
2.2	Деструктори підкласів. Оператори уведення-виведення для ієрархії класів. Класифікація об'єктів. Ієрархія класів плоских геометричних фігур. Абстрактний базовий клас, абстрактні методи.	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
	Класи "квадрат", "круг", "трикутник".	Самостійна робота	3	
2.3	Ієрархія об'ємних геометричних фігур. Відношення "has-a", композиція об'єктів.	Лекція	2	
	Оголошення розгалуженої ієрархії класів. Використання абстрактного базового класу. Віртуальні методи, віртуальні деструктори.	Лабораторна робота	2	Наступне лабораторне заняття
	Використання вкладених класів, підкласи для розширення можливостей надкласу.	Самостійна робота	3	
2.4	Агрегація об'єктів. Проектування динамічного масиву об'ємних фігур. Оператори індексування. Питання власності агрегованих об'єктів. .	Лекція	2	
	Побудова типів на основі композиції класів (стандартних і власних). Використання власних (вкладених) класів винятків.	Лабораторна робота	2	Наступне лабораторне заняття
	Глибоке копіювання контейнера. Бібліотека RTTI: перевірка типу об'єкта, приведення типу	Самостійна робота	3	
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
	Додаткові теми по наслідуванню	Самостійна робота	3	
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
	Приклади множинного наслідування	Самостійна робота	3	
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.	Лекція	2	
	Створення класів за допомогою множинного наслідування. Модульне тестування написаного коду.	Лабораторна робота	2	Наступне лабораторне заняття
	Шаблон з параметром не типом.	Самостійна робота	3	
2.8	Алгоритм пошуку значення в послідовності: узагальнення типу даних, структури послідовності, умови відшукання. Наслідування, включення, часткова спеціалізація.	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
	Шаблони - параметри шаблонів. Дружні конструкції.	Самостійна робота	3	
2.9	Стандартна бібліотека шаблонів (STL). Базові поняття: контейнер, ітератор, алгоритм, функтор. Класифікація ітераторів, моделі ітераторів.	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
	Класифікація контейнерів, спільні можливості.	Самостійна робота	3	
2.10	Послідовні контейнери: концепція, влаштування, базові операції. Приклади використання контейнерів vector, list, deque.	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
	Порівняння ефективності послідовних контейнерів. Ітератори вставляння. Потокові ітератори.	Самостійна робота	3	
2.11	Контейнерні адаптери stack, queue, priority_queue. Приклади використання: побудова польського запису, моделювання багатопотокового обслуговування черги клієнтів.	Лекція	2	
	Вивчення функціональності vector<T>: дописування, вставляння, вилучення, індексування,	Лабораторна робота	2	Наступне лабораторне заняття

	перебір, копіювання; дослідження типу. Використання у власних програмах.			
	Доступ до реалізації. Наслідування від стандартних контейнерів.	Самостійна робота	3	
2.12	Використання класів характеристик для побудови шаблонів Класи, що налаштовують функціональність шаблона.	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
	Зміна характеристик класу string. Метапрограмування.	Самостійна робота	3	
2.13	Класифікація функторів. Об'єкт-функція. Стандартні функтори та функціональні адаптери. Створення власних функторів. Огляд бібліотеки алгоритмів. .	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
	Приклади використання для написання "програм без циклів"	Самостійна робота	3	
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
	Додаткові теми по асоціативних конструкторах	Самостійна робота	3	
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
	Викристання адаптерів	Самостійна робота	3	
2.16	"Майже контейнери": valarray<Num>, auto_ptr<Pointer>, complex<Num>, pair<T1,T2>.	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	

Тиждень	Тема, план, короткі тези	Форма діяльності	Тривалість, год	Термін виконання
3.1	Модель даних, примітивні вбудовані типи, оператори та інструкції Python. Стандартна бібліотека. Введення-виведення даних. Імпорт.	лекція	2	
	Основні прийоми роботи в Jupyter Notebook. Програмна реалізація простих алгоритмів	лаборат. заняття	2	наступне лаб. заняття
	Додаткові теми щодо стандартної бібліотеки	Самостійна робота	3	
3.2	Вбудовані структури даних. Визначення функцій.	лекція	2	
	Умовні конструкції та цикли	лаборат. заняття	2	наступне лаб. заняття
	Додаткові теми щодо інструкцій та функцій	Самостійна робота	3	
3.3	Рядки та файли. Символи. Індекси та зрізи. Функції для роботи з рядками та методи рядків. Режими відкриття файлів. Операції з файлами. Інструкція with/as. Модуль datetime.	лекція	2	
	Програмування з використанням рядків і файлів	лаборат. заняття	2	наступне лаб. заняття
	Додаткові теми щодо роботи з файлами	Самостійна робота	3	
3.4	Функції як одиниці роботи. Декомпозиція. Створення та виклик функції. Позиційні та іменовані аргументи. Значення за замовчуванням. Області видимості. Вкладення функцій. Лямбда-функції. Передавання довільної кількості аргументів	лекція	2	
	Програмування з використанням функцій та колекцій	лаборат. заняття	2	наступне лаб. заняття
	Використання функцій для розв’язування алгоритмічних задач	Самостійна робота	3	
3.5-6	Особливості ООП в Python: інкапсуляція, наслідування, поліморфізм. Протокол класу. Об'єкт класу. Створення екземплярів класу, __init__(). Стандартні атрибути, __dir__(). Особливості керування доступом до атрибутів; property. Змінні та методи класу. Статичні методи. Перевантаження операторів. Наслідування класів, вирішення посилань на атрибути. Перевизначення методів. isinstance() та issubclass(). Копіювання екземплярів.	лекція	4	
	Об’єктно-орієнтоване програмування	лаборат. заняття	8	4 тижні

	Використання ООП для розв’язування задач та створення проектів	Самостійна робота	6	
3.7	Модуль csv: reader і writer, DictReader і DictWriter для роботи з csv-файлами.		2	
3.8-9	Механізм винятків. Ієрархія класів винятків. Поняття про якість програм. Типи тестів. Модульне тестування. Концепція unittest: test fixture,test case,test suite,test runner. Загальні assert-методи. Тестування класів.	лекція	4	
	Модульне тестування програм		4	
	Практичні завдання з використанням з csv-файлами.	Самостійна робота	6	
3.10	Декоратори функцій і класів. Ітератори: явне використання iter() і next(). Неявне ітерування в циклах. Генератори. Використання виразів з генераторами для заповнення колекцій.	лекція	2	
	Заповнення колекцій за допомогою генераторів.	лаборат. заняття	2	наступне лаб. заняття
	Додаткові теми щодо генераторів для заповнення колекцій	Самостійна робота	3	
3.11	Матрично-векторні обчислення: NumPy. ndarray: утворення масивів, атрибути. Індексування, зрізка масивів. Використання універсальних функцій, клас ufunc. Маски.	лекція	2	
	Операції над векторами і матрицями засобами NumPy	лаборат. заняття	2	наступне лаб. заняття
	Застосування NumPy. Ndarray для роз’язування задач	Самостійна робота	3	
3.12	Візуалізація даних засобами matplotlib. Використання графічних панелей в інтерактивному режимі. Керування стилями і кольорами. Легенди. Композиції зображень. Тривимірні візуалізація .	лекція	2	
	Візуалізація даних.	лаборат. заняття	2	наступне лаб. заняття
	Тривимірна візуалізація даних	Самостійна робота	3	
3.13	Аналіз гетерогенних даних засобами Pandas.	лекція	4	
-14	Робота з табличними даними, статистичний аналіз	лаборат. заняття	4	наступне лаб. заняття
	Додаткові теми використання Pandas	Самостійна робота	6	
3.15	Інтерактивний комп'ютинг, API.	лекція	4	
-16	Розробка графічного інтерфейсу користувача засобами widgets			
	Розробка графічного інтерфейсу користувача.	лаборат. заняття	4	наступне лаб. заняття

	Додаткові теми щодо розробки графічного інтерфейсу користувача засобами widgets	Самостійна робота	3	
	Підсумкове заняття	лаборат. заняття	2	