КРИВОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
Кафедра моделювання та програмного забезпечення

ЗАТВЕРДЖУЮ
Перший проректор
_____ Владислав ЧУБАРОВ
“_____”_____2025 р.

РОБОЧА ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Основи програмування
(шифр і назва навчальної дисципліни)

спеціальність 121 – «Інженерія програмного забезпечення»
(шифр і назва напряму підготовки)

факультет Інформаційних технологій
(назва інституту, факультету, відділення)

Форма навчання	Курс	Семестр	Всього годин за планом	Кількість національних кредитів	Всього аудиторних годин	Аудиторних годин, (у тому числі КЗ)		Самостійна робота (год.)	Контрольно-модульні роботи	Залік (сем.)	Екзамен (сем.)
						Лекції	Лабораторні				
Денна	1	1	180	6	64	32	32	116	2	*	
	1	2	180	6	72	36	36	108	2		*
Заочна	1	1	180	6	16	8	8	164	-	*	
	1	2	180	6	16	8	8	164	-		*

Кривий Ріг – 2025 рік

Робочу програму навчальної дисципліни «Основи програмування» для здобувачів першого (бакалаврського) рівня вищої освіти за освітньою програмою «Інженерія програмного забезпечення» розроблено згідно з ОПП галузі знань 12 «Інформаційні технології» зі спеціальності 121 «Інженерія програмного забезпечення».

Розробник: старший викладач кафедри МПЗ Рибальченко О.Г.

Робоча програма затверджена на засіданні кафедри моделювання та програмного забезпечення

Протокол від “ 22 ” листопада 2024 року № 4

Завідувач кафедри МПЗ, доцент, к.п.н. _____ Андрій СТРЮК

Схвалено вченою радою факультету інформаційних технологій

Протокол від “26” грудня 2024 року № 5

Голова вченої ради _____ Іван МУЗИКА

Схвалено групою забезпечення ОПП

Протокол від “ 22 ” листопада 2024 року № 4

Гарант ОПП _____ Андрій СТРЮК

ЗМІСТ

1. ОПИС НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

<table>
<tr><td rowspan="2">Найменування показників</td><td rowspan="2">Галузь знань, спеціальність, ступінь вищої освіти</td><td colspan="2">Характеристика навчальної дисципліни</td></tr>
<tr><td>денна форма навчання</td><td>заочна форма навчання</td></tr>
<tr><td>Кількість кредитів – 12</td><td>Галузь знань
12 Інформаційні технології
(шифр і назва)</td><td colspan="2">Нормативна</td></tr>
<tr><td>Модулів – 2</td><td rowspan="5">Спеціальність
121 Інженерія програмного забезпечення
(код та найменування спеціальності)</td><td colspan="2">Рік підготовки:</td></tr>
<tr><td>Змістових модулів – 4</td><td>2024/2025 н.р.</td><td>2024/2025 н.р.</td></tr>
<tr><td rowspan="3">Загальна кількість годин - 360</td><td colspan="2">Семестр</td></tr>
<tr><td>1,2-й</td><td>1,2-й</td></tr>
<tr><td colspan="2">Лекції</td></tr>
<tr><td rowspan="11">Тижневих годин для денної форми навчання:
аудиторних – 4
самостійної роботи студента – 6,59</td><td rowspan="11">Ступінь вищої освіти:
бакалавр</td><td>68 год.</td><td>16 год.</td></tr>
<tr><td colspan="2">Практичні, семінарські</td></tr>
<tr><td>-</td><td>-</td></tr>
<tr><td colspan="2">Лабораторні</td></tr>
<tr><td>68 год.</td><td>16 год.</td></tr>
<tr><td colspan="2">Самостійна робота</td></tr>
<tr><td>224 год.</td><td>328 год.</td></tr>
<tr><td colspan="2">Вид контролю</td></tr>
<tr><td colspan="2">Залік – 1 семестр
Екзамен – 2 семестр</td></tr>
</table>

Примітка.

Співвідношення кількості годин аудиторних занять до самостійної і індивідуальної роботи становить:

для денної форми навчання – 0,607

для заочної форми навчання – 0,089

2. МЕТА ТА ЗАВДАННЯ НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

2.1. Метою дисципліни «Основи програмування» є вивчення принципів обробки інформації на комп'ютері, формування у студентів навичок розробки та відлагодження прикладних програм мовами програмування C++ та Python із застосуванням технологій структурного, модульного та об'єктно-орієнтованого програмування, отримання практичних навичок використання інтегрованих середовищ розробки програмного забезпечення.

2.2. Основними завданнями вивчення дисципліни «Основи програмування» є

- оволодіння навичками формалізації задач;
- оволодіння базовими методами проектування алгоритмів і програм;
- вивчення основних методологій розробки програмного забезпечення;
- отримання навичок розробки програм мовами програмування C++ та Python;
- вивчення основ управління проектами розробки програмного забезпечення;
- вивчення основ забезпечення та контролю якості програмного забезпечення.

2.3. Відповідно до освітньої програми дисципліна забезпечує наступні **компетентності**:

Загальні компетентності:

ЗК01 Здатність до абстрактного мислення, аналізу та синтезу;
ЗК02 Здатність застосовувати знання у практичних ситуаціях;
ЗК05 Здатність вчитися і оволодівати сучасними знаннями;

Фахові компетентності

СК10 Здатність накопичувати, обробляти та систематизувати професійні знання щодо створення і супроводження програмного забезпечення та визнання важливості навчання протягом всього життя;

СК13 Здатність обґрунтовано обирати та освоювати інструментарій з розробки та супроводження програмного забезпечення.

2.4. **Програмні результати навчання** освітньої програми, яким відповідає дисципліна:

ПР06 Уміння вибирати та використовувати відповідну задачі методологію створення програмного забезпечення.

ПР07 Знати і застосовувати на практиці фундаментальні концепції, парадигми і основні принципи функціонування мовних, інструментальних і обчислювальних засобів інженерії програмного забезпечення.

ПР15 Мотивовано обирати мови програмування та технології розробки для розв'язання завдань створення і супроводження програмного забезпечення.

ПР17 Вміти застосовувати методи компонентної розробки програмного забезпечення.

В результаті вивчення дисципліни студенти повинні **знати:**

- основні синтаксичні конструкції мов програмування C++ та Python;
- основні функції стандартних бібліотек мов програмування C++ та Python;
- основні методології розробки програмного забезпечення;
- основи управління проектами розробки програмного забезпечення;
- основи забезпечення та контролю якості програмного забезпечення.

вміти:

- розробляти структуровані алгоритми розв'язання задач, використовуючи типові алгоритмічні конструкції;
- створювати програми мовами програмування C++ та Python із застосуванням технологій структурного, модульного та об’єктно-орієнтованого програмування;
- відлагоджувати та тестувати програми для одержання коректних результатів;
- користуватися інтегрованими середовищами розробки програмного забезпечення.

2.5. Міждисциплінарні зв’язки

При вивченні дисципліни використовуються знання здобувачів з дисциплін шкільних курсів «Інформатика», «Математика», дисципліна «Алгоритмізація обчислювальних процесів».

Знання, одержані здобувачами при вивченні дисципліни, використовуються при вивченні дисциплін «Об’єктно-орієнтоване програмування» з курсовою роботою, «Розробка програм на платформі .NET».

Вимоги до знань та умінь визначаються галузевими стандартами вищої освіти України.

3. ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Заліковий модуль 1

Змістовий модуль 1. Основи структурного програмування мовою С++

Тема 1 – 4 год. Основні поняття мови С++

Історія розвитку мови С++. Структурне, модульне та об'єктно-орієнтоване програмування. Поняття компілятора, інтерпретатора. Структура програми на С++ та етапи її обробки. Препроцесор. Заголовні файли. Директива #include. Використання об'єктів cin та cout. Коментарі та їх використання. Призначення функції main(). Стандартні функції та їх використання.

Тема 2 – 4 год. Типи даних мови С++. Константи та змінні у С++.

Змінні. Виділення пам'яті. Розмір змінних і його визначення за допомогою sizeof. Знакові і беззнакові величини (signed /unsigned). Основні типи змінних (int, long, char, float, double, bool). Розмір в пам'яті і допустимий діапазон значень. Оголошення змінних. Чутливість до регістра букв. Одночасне оголошення більш ніж однієї змінної. Надання значення змінним. Ініціалізація.

Оператор typedef. Використання типів int, short та long. Переповнювання беззнакових і знакових цілих. Символи. Тип char і числа. Спеціальні символи. Константи. Константи-літерали. Символічні константи. Визначення констант за допомогою #define і const. Константи, що перераховуються. Оператор enum.

Тема 3 – 4 год. Вирази, математичні операції та операції відношення

Оператори. Порожні символи. Блоки і складені оператори. Вирази. Операція присвоювання. Математичні операції. Цілочисельне ділення і залишок від ділення. Поєднання присвоювання і математичних операцій. Інкремент і декремент. Префіксні і постфіксні операції. Пріоритет виконання операцій. Вкладені дужки. Операції відношення.

Тема 4 – 4 год. Структури управління та повторення у мові С++

Простий оператор if. Стилі виділення відступами. Повна форма оператора if з ключовим словом else. Вкладені оператори if. Використання дужок у вкладених операторах if. Логічні оператори І, АБО, НЕ (&&, ||, !). Пріоритет операцій відношення і логічних операцій. Умовна (тернарна) операція.

Оператор switch. Використання ключових слів case, default і break. Використання оператора switch в програмах, що містять меню.

Цикли. Цикл з ключовим слово goto. Цикли while. Цикли while із складними умовами. Оператори continue та break. Цикли do-while. Цикли for. Множинна ініціалізація та інкрементування в циклах for. Видимість імен в циклах. Безконечні цикли.

Змістовий модуль 2. Функції та управління пам'яттю у мові С++

Тема 5 – 4 год. Покажчики та посилання

Що таке покажчик? Адреси змінних. Використання операції & (адреса). Зберігання адреси в покажчику. Імена покажчиків. Непряма адресація. Операція *. Покажчики, адреси і змінні. Маніпулювання даними за допомогою покажчиків. Стек і вільна пам'ять (heap). Виділення, використання і видалення змінних у вільній пам'яті. Оператори new і delete. Null-покажчик. Покажчики const. Витік пам'яті. Створення і видалення об'єктів у вільній пам'яті. Покажчик this. Створення і використання посилань. Використання оператора адреси до посилань.

Тема 6 – 4 год. Масиви

Масиви як структурований тип даних. Визначення масивів. Синтаксис об'яви масивів. Поняття "елемент масиву" та "індекс масиву". Синтаксис використання масивів. Зв'язок

покажчиків з масивами. Вирази та арифметика з покажчиками. Динамічний розподіл пам'яті. Стандартні алгоритми роботи з масивами: визначення екстремальних значень, статистичних показників, упорядкування масивів, пошук визначеного елементу. Багатовимірні масиви. Приклади програм з багатовимірними масивами. Реалізація алгоритмів, пов'язаних з обробкою матриць.

Тема 7 – 4 год. Організація функцій користувача

Програмні модулі мови С++. Функції математичної бібліотеки. Функції бібліотеки стандартного введення/виведення. Використання функцій при розробці програм на мові С++. Визначення функцій. Прототипи функцій та файли заголовків. Розробка власних функцій. Виклик функції за значенням. Способи обміну інформацією між функціями. Локальні та глобальні змінні. Виклик функції за посиланням –передача параметрів за адресою. Приклад програми, що використовує виклик за посиланням. Класи пам'яті. Правила області дій. Параметри за умовчанням (default). Перевантаження (поліморфізм) функцій. Розробка програми у вигляді проекту. Зовнішні змінні. Рекурсія. Приклад використання рекурсії: числа Фібоначчі. Перевантаження функцій. Шаблони функцій.

Тема 8 – 4 год. Робота з рядками

Масиви символів (рядки С-типа). Використання cin, cin.get() для введення рядків. Використання бібліотеки string.h для роботи з рядками. Визначення довжини рядка за допомогою strlen(). Присвоєння (копіювання) рядків функціями strcpy() і strncpy(). Порівняння рядків за допомогою strcat() і strncat(). Основні функції для рядків С-стилю.

Заліковий модуль 2

Змістовий модуль 3. Об'єктно-орієнтоване програмування на С++

Тема 9 – 4 год. Структури та класи у мові С++

Створення нових типів даних. Класи і члени. Оголошення класу. Визначення об'єкту. Класи і об'єкти. Доступ до членів класу. Ключові слова private і public. Обґрунтування необхідності в приватних членах-даних. Приватність і безпека. Реалізація методів класу. Конструктори і деструктори. Конструктори і деструктори за умовчанням. Використання покажчика this. Функції-члени const. Інтерфейс і реалізація. Реалізація inline. Класи з іншими класами як дані члени. Структури.

Тема 10 – 4 год. Інкапсуляція, спадкування та поліморфізм.

Спадкування. Базові і похідні класи. Синтаксис похідних класів. Private та protected. Конструктори і деструктори в похідних об'єктах. Передача аргументів в базові конструктори. Перевантаження конструкторів в похідних класах. Перекривання методу базового класу в похідному. Утаєння методів базового класу. Віртуальні методи. Раннє та пізнє зв'язування. Віртуальні деструктори. Віртуальний конструктор копіювання. Поліморфізм. Ідентифікація типа під час виконання. Множинне спадкування. Віртуальне спадкування. Абстрактні типи даних. Повністю віртуальні функції. Складні ієрархії абстракцій.

Тема 11 – 8 год. Потокове введення та виведення, робота з файлами.

Потоки. Буферизація. Стандартні об'єкти введення/виводу. Введення з використанням cin. Використання cin.get(). Введення рядків із стандартного введення. Використання cin.getline(). Использование cin.ignore(). Використання cin.peek() і cin.putback(). Виведення з використанням cout. Очищення виводу. Метод cout.flush(). Функції cout.put() і cout.write(). Маніпулятори, прапори і інструкції форматування. Використання cout.width(), cout.fill(), cout.setf(). Потоки проти функції printf.

Файлове введення/виведення. Об'єкти типа ofstream і ifstream. Відкриття та закриття файлів для введення і виводу. Функції eof(), bad(), fail(), good(). Зміна поведінки ofstream при відкритті файлів. Використання ios::app, ios::ate, ios::trunc, ios::nocreate, ios::noreplace.

Змістовий модуль 4. Програмування мовою Python.

Тема 12 – 8 год. Структурне програмування мовою Python.

Історія створення мови Python. Інтерпретатор Python. Інтерактивний та сценарний режими. Інтегровані середовища розробки для Python. Введення та виведення даних, форматне виведення. Динамічна типізація в мові Python. Числа, рядки, послідовності, відображення. Арифметичні вирази. Складені об’єкти: списки, кортежі, словники, множини. Робота з файлами.

Структури управління та повторення у мові Python. Повне та неповне розгалуження. Реалізація багатоваріантного вибору через розгалуження. Параметричний цикл з варіативною та без варіативної частини. Цикл з передумовою. Оператори continue та break. Виконання команд всередині контексту (with). Обробка виключень в програмах, вбудовані типи виключень та визначення нових виключень.

Тема 13 – 4 год. Функції користувача та основи функціонального програмування в Python.

Визначення власних функцій. Передача параметрів і повернення результатів. Значення аргументів функції за замовчуванням. Довільний набір аргументів. Іменовані аргументи. Правила видимості. Функції, як об'єкти та замикання. Області видимості у мові Python. Декоратори. Ітератори. Генератори та со-програми. Генератори списків. Вирази-генератори. Основи декларативного програмування. Оператор Lambda. Атрибути функцій. Розробка рекурсивної функції на базі її процедурної форми. Послідовна, паралельна та псевдопаралельна рекурсії. Обробка списків рекурсивними функціями.

Тема 14 – 4 год. Модульне програмування. Стандартні та нестандартні модулі Python.

Створення і використання модуля. Пошук модулів і компільовані файли. Стандартні модулі: sys, os. Пакети. Коротка характеристика нестандартних модулів Python. Модуль чисел з плаваючою точкою Decimal. Модуль раціональних чисел Fractions. Модуль стандартних математичних функцій Math. Модуль абстрактних базових класів Numbers. Модуль псевдовипадкових чисел Random. Модуль для роботи з комплексними числами CMath. Модуль для роботи з масивами Array. Модуль сортування списків Bisect. Стандартні модулі обробки рядків String і Codecs. Модуль приблизного порівняння двох рядків DiffLib. Модуль для роботи з кодуванням і регулярними виразами Re. Модуль перетворення даних Struct. Модуль доступу до бази символів UnicodeData.

Тема 15 – 4 год. Об’єктно-орієнтоване програмування мовою Python.

Програмування класів. Методи класів. Екземпляри класів. Спадкування. Простір імен. Перевантаження операторів. Менеджери контексту. Обробка виключень в програмах, вбудовані типи виключень та визначення нових виключень.

4. СТРУКТУРА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Назви змістових модулів і тем	Кількість годин							
	Денна форма				Заочна форма			
	усього	у тому числі			усього	у тому числі		
		лекції	лабораторні	самостійна робота		лекції	лабораторні	самостійна робота
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Заліковий модуль 1								
Змістовий модуль 1. Основи структурного програмування мовою С++								
Тема 1. Основні поняття мови С++	20	4	4	12	20	1	1	18
Тема 2. Типи даних мови С++. Константи та змінні у С++	24	4	4	16	24	1	1	22
Тема 3. Вирази, математичні операції та операції відношення	23	4	4	15	23	1	1	21
Тема 4. Структури управління та повторення у мові С++	23	4	4	15	23	1	1	21
Разом за модуль 1	**90**	**16**	**16**	**58**	**90**	**4**	**4**	**82**
Змістовий модуль 2. Функції та управління пам'яттю у мові С++								
Тема 5. Покажчики та посилання	24	4	4	16	24	1	1	22
Тема 6. Масиви	23	4	4	15	23	1	1	21
Тема 7. Організація функцій користувача	23	4	4	15	23	1	1	21
Тема 8. Робота з рядками	20	4	4	12	20	1	1	18
Разом за модуль 2	**90**	**16**	**16**	**58**	**90**	**4**	**4**	**82**
Разом за семестр	**180**	**32**	**32**	**116**	**180**	**8**	**8**	**164**
Заліковий модуль 2								
Змістовий модуль 3. Об'єктно-орієнтоване програмування на С++								
Тема 9. Структури та класи у мові С++	25	4	4	17	25	1	1	23
Тема 10. Інкапсуляція, спадкування та поліморфізм.	25	4	4	17	25	1	1	23
Тема 11. Потокове введення та виведення, робота з файлами.	25	4	4	17	25	1	1	23
Разом за модуль 3	**75**	**12**	**12**	**51**	**75**	**3**	**3**	**69**
Змістовий модуль 4. Програмування мовою Python								
Тема 12. Структурне програмування мовою Python.	30	8	8	14	30	2	2	26
Тема 13. Функції користувача та основи функціонального програмування в Python.	35	8	8	19	35	1	1	33
Тема 14. Модульне програмування. Стандартні та нестандартні модулі Python.	20	4	4	12	20	1	1	18
Тема 15. Об'єктно-орієнтоване програмування мовою Python.	20	4	4	12	20	1	1	18
Разом за модуль 4	**105**	**24**	**24**	**57**	**105**	**5**	**5**	**95**
Разом за семестр	**180**	**36**	**36**	**108**	**180**	**8**	**8**	**164**
Усього годин	**360**	**68**	**68**	**224**	**360**	**16**	**16**	**328**

5. ТЕМИ ПРАКТИЧНИХ ЗАНЯТЬ

Не передбачено.

6. ТЕМИ ЛАБОРАТОРНИХ ЗАНЯТЬ

№ з/п	Назва теми	Кількість годин	
		Денна ф. н.	Заочна ф. н.
1.	Програмування лінійних алгоритмів мовою С++	2	
2.	Програмування задач з перевіркою кількох умов мовою С++	4	1
3.	Програмування багатоваріантного вибіру мовою С++	2	1
4.	Програмування простих циклів мовою С++	4	1
5	Програмування ітераційних циклів мовою С++	2	
6.	Програмування задач пошуку max / min в масиві мовою С++	2	1
7.	Програмування циклів обробки двовимірних масивів мовою С++	4	1
8.	Функції користувача у С++	4	1
9.	Покажчики у С++	4	1
10.	Рядки у С++	4	1
	Разом за семестр	**32**	**8**
16.	Перевантаження функцій. Шаблони	4	1
17.	Структури С++	2	
18.	Класи С++	4	1
19.	Файли С++	2	1
20.	Базові конструкції мови Python	4	
21.	Рядки та колекції Python	4	1
22.	Функції Python	4	1
23.	Декоратори, ітератори та генератори Python	4	1
24.	Класи Python	4	1
25.	Файли та менеджери контексту Python	4	1
	Разом за семестр	**36**	**8**
	Усього годин	**68**	**16**

7. САМОСТІЙНА РОБОТА

На самостійну роботу студентам денної форми навчання відведено 224 години, заочної - 328 годин.

Самостійна робота студентів при вивченні дисципліни «Основи програмування» залучає такі складові:

- опрацювання теоретичних основ прослуханого лекційного матеріалу;
- вивчення окремих тем або питань, що передбачені для самостійного опрацювання;
- підготовка до виконання, а також до захисту лабораторних робіт;
- підготовка до проведення контрольних заходів.

При виконанні самостійної роботи студент денної форми навчання повинен:

- опрацювати матеріал, що винесено на самостійне вивчення;
- скласти алгоритми та програми рішення індивідуальних задач;
- оформити звіт з лабораторних робіт;

При виконанні самостійної роботи студент заочної форми навчання повинен:

- опрацювати теоретичний матеріал;
- скласти алгоритми та програми рішення індивідуальних задач;
- оформити звіт з лабораторних робіт;

Питання для самостійного опрацювання

№	Назва теми	Кількість годин	
		денна	заочна
1.	Зовнішні (extern) і статичні (static) глобальні змінні і функції	6	8
2.	Складні декларації, визначення синонімів типів (typedef)	6	8
3.	Роздільна компіляція програмних модулів. Використання * .h файлів.	6	8
4.	Призначені для користувача типи даних (enum, struct, union)	6	8
5.	Перевантаження операторів і функцій. Особливості перевантаження і повернення значень в операторах «=» і «+». Перевантаження оператора «+ =»	6	8
6.	Перевантаження операторів введення і виведення, що працюють з потоками C ++	6	8
7.	Перевантаження оператора приведення типу	6	8
8.	Множинне спадкування	6	8
9.	Статичні методи і поля класу	6	8
10.	Простори імен (namespace, using). Простір імен std.	6	8
11	Опрацювання матеріалу лекцій, робота з рекомендованою літературою	40	128
12	Виконання лабораторних робіт	50	60
13	Підготовка звітів з лабораторних робіт	50	60
14	Підготовка до контрольних заходів	24	-
	Разом	**224**	**328**

8. МЕТОДИ НАВЧАННЯ

Використовуються наступні методи навчання: лекції, лабораторні заняття, самостійна робота.

Навчальна лекція – це логічне, послідовне викладання змісту навчання, яке характеризується судженнями, висновками, підсумком. Вона охоплює основний теоретичний матеріал однієї або кількох тем навчальної дисципліни. Призначенням лекції є формування у здобувачів фундаментальних знань з дисципліни, а також визначає основний зміст і характер усіх інших навчальних занять та самостійної роботи здобувачів із цієї дисципліни.

Лабораторне заняття - форма організації навчання, яку проводять за завданням і під керівництвом НПП. Основні дидактичні цілі – експериментальне підтвердження вивчених теоретичних положень навчальної дисципліни та формування вмінь й навичок їх практичного застосування. Проведення лабораторного заняття ґрунтується на попередньо підготовлених наборах завдань різної складності для розв'язання на занятті. Лабораторне заняття проводиться у навчальних лабораторіях з використанням пристосованого до умов навчального процесу устаткування.

Самостійна робота здобувача є основним способом оволодіння навчальним матеріалом у час, вільний від обов'язкових аудиторних занять. Мета виконання самостійної роботи – поглиблення, узагальнення й закріплення теоретичних знань і практичних умінь здобувачів із дисципліни шляхом вироблення вміння самостійної роботи з навчальною і фаховою літературою та інформацією в мережі Інтернет.

Самостійна робота здобувачів здійснюється у формі: підготовки до лекцій і лабораторних занять, виконанні самостійних проектів. Самостійну роботу здобувач може виконувати у бібліотеці, комп'ютерних класах, а також у домашніх умовах.

Підготовка до лекцій передбачає самостійне опрацювання теоретичного матеріалу. При цьому необхідно звернути увагу на необхідність чіткого засвоєння основних термінів та визначень, розуміння їх змісту, обов'язкового аналізу використання теоретичних положень для розв'язання наданих прикладів.

Самоперевірку засвоєння навчального матеріалу здобувач здійснює за контрольними запитаннями, що надано після кожної теми у конспекті лекцій та іншій літературі, та після кожного лабораторного заняття у відповідних методичних вказівках. Якщо на деякі запитання здобувач не може надати відповіді, то необхідно повторити вивчення навчального матеріалу, або визначити правильну відповідь з викладачем на консультації.

Під час вивчення даної дисципліни використовуються:

− мультимедійні освітні технології: інтерактивні лекції (презентації) із використанням програми MS Power Point у поєднанні з анімацією та звуковим супроводом; перегляд відеороликів за окремими пунктами тем занять, використання електронних посібників;

− діалогові технології: організація групових обговорень, використання «мозкового штурму».

Лекції проводяться з використанням технічних засобів навчання й супроводжуються демонстрацією презентацій за допомогою проектора.

У разі виникнення необхідності забезпечення навчального процесу в дистанційному режимі супровід та контроль знань реалізовується за допомогою дистанційного курсу, розробленого в Google ClassRoom. Онлайн лекції, консультації та усні відповіді на питання, захист проєктів проводиться за допомогою Google Meet або Zoom.

9. МЕТОДИ КОНТРОЛЮ

Основними завданнями контролю знань здобувачів вищої освіти з дисципліни є оцінювання засвоєння теоретичних знань і практичних навичок, отриманих під час навчання.

Контрольні заходи мають виконувати наступні функції:

- стимулювати систематичну самостійну роботу над навчальним матеріалом;
- забезпечувати закріплення та реалізацію набутих теоретичних знань при підготовці до лабораторних занять;
- прищеплювати навички відповідального ставлення до своїх обов'язків, самостійного цілеспрямованого пошуку потрібної інформації, чіткої організації свого робочого дня.

Оцінювання знань здобувачів складається з поточного, модульного та підсумкового контролю.

Поточний контроль знань здобувачів вищої освіти передбачає оцінювання за наступними основними напрямами:

- перевірка теоретичних знань;
- перевірка підготовки до лабораторних занять.

З даних компонентів складаються загальні бали, які фіксуються в журналі викладача.

Оцінювання рівня засвоєння теоретичних знань здобувачів вищої освіти проводиться під час усної співбесіди зі здобувачами по теоретичним матеріалам, за результатами захисту лабораторних робіт й виконання самостійних робіт. Підсумковим контролем є залік у І семестрі та екзамен у ІІ семестрі.

10. РОЗПОДІЛ БАЛІВ, ЯКІ ОТРИМУЮТЬ ЗДОБУВАЧІ

Контроль успішності студента з дисципліни «Основи програмування» здійснюється у формі поточного, модульного та семестрового контролю.

Поточний контроль має за мету перевірку якості засвоєння матеріалу студентами та залік кредитного модуля навчальної дисципліни. Він здійснюється під час проведення лабораторних занять, а також контрольної модульної роботи. Для цього використовується модульно-рейтінгова система оцінювання, яка передбачає розподіл балів за вивчення всіх запланованих видів робіт. При цьому максимальна кількість балів за модуль при умові його бездоганного виконання дорівнює 100. Ця сума складається з балів, що їх накопичив студент у ході виконання лабораторних робіт.

КМР у будь-якому модулі відображує теоретичні знання і відповідає 25-ти відсоткам його ваги, тобто вона може дати максимально 25 балів при найвищій якості виконання. При зниженні якості КМР знижується і сума балів відповідно до шкали, що наводиться у таблиці:

Шкала оцінювання контрольно-модульних робіт

Відсоток вірних компонентів КМР	0 – 30	31 – 60	61 – 75	76 – 85	86 – 94	95 – 100
Сума балів за КМР	0	5	10	15	20	25

Успішність студентів-заочників оцінюється аналогічним чином, за 100-бальною шкалою, але без проведення контрольно-модульних робіт.

Шкала оцінювання лабораторних робіт

№ модуля	№ зан.	Вид роботи	Тема	Максимальна кількість балів	
				денна	заочна
1	1	Лабораторна робота № 1	Програмування лінійних алгоритмів мовою С++	15	20
	2, 3	Лабораторна робота № 2	Програмування задач з перевіркою кількох умов мовою С++	15	20
	4	Лабораторна робота № 3	Програмування багатоваріантного вибіру мовою С++	15	20
	5,6	Лабораторна робота № 4	Програмування простих циклів мовою С++	15	20
	7	Лабораторна робота № 5	Програмування ітераційних циклів мовою С++	15	20
Контрольно-модульна робота				**25**	-
Разом за модуль				**100**	**100**
2	8	Лабораторна робота № 6	Програмування задач пошуку max / min в масиві мовою С++	15	20
	9, 10	Лабораторна робота № 7	Програмування циклів обробки двовимірних масивів мовою С++	15	20
	11, 12	Лабораторна робота № 8	Функції користувача у С++	15	20
	13, 14	Лабораторна робота № 9	Покажчики у С++	15	20
	15, 16	Лабораторна робота № 10	Рядки у С++	15	20
Контрольно-модульна робота				**25**	-
Разом за модуль				**100**	**50**
3	1,2	Лабораторна робота № 11	Перевантаження функцій. Шаблони	15	20
	3	Лабораторна робота № 12	Структури С++	15	20
	4,5	Лабораторна робота № 13	Класи С++	15	20
	6	Лабораторна робота № 14	Файли С++	15	20
	7,8	Лабораторна робота № 15	Базові конструкції мови Python	15	20
	Контрольно-модульна робота			**25**	-
	Разом за модуль			**100**	**100**

№ модуля	№ зан.	Вид роботи	Тема	Максимальна кількість балів	
				денна	заочна
4	9, 10	Лабораторна робота № 16	Рядки та колекції Python	15	20
	11, 12	Лабораторна робота № 17	Функції Python	15	20
	13, 14	Лабораторна робота № 18	Декоратори, ітератори та генератори Python	15	20
	15, 16	Лабораторна робота № 19	Класи Python	15	20
	17, 18	Лабораторна робота № 20	Файли та менеджери контексту Python	15	20
Контрольно-модульна робота				**25**	-
Разом за модуль				**100**	**100**

Лабораторні роботи у модулі відображують оволодіння навичками та вміння застосовувати знання на практиці і сукупно відповідають 75-ти відсоткам ваги модуля для денної форми навчання і 100-та відсоткам для заочної форми. Таким чином всі лабораторні роботи модуля при бездоганному їх виконанні можуть дати 75/100 балів. Ці бали розподіляються поміж лабораторними роботами модуля у відповідності до їх відносної складності. При зниженні якості виконання тієї чи іншої лабораторної роботи, знижується і кількість балів, якою вона оцінюється.

Оцінювання кожної задачі лабораторної роботи ведеться за наступними показниками:

1. Своєчасність практичного виконання лабораторної роботи (у тиждень згідно із графіком робіт) (0-3 бали).
2. Своєчасність захисту виконаної лабораторної роботи (у тиждень наступний за тижнем планового виконання роботи) (0-3 бали).
3. Якість знайдених студентом рішень (ефективність алгоритму, доречність використання елементів інтерфейсу, тощо) (1-9 балів).

Для студентів-заочників показники оцінюються наступним чином:

1. Своєчасність практичного виконання лабораторної роботи (0-6 бали).
2. Якість знайдених студентом рішень (ефективність алгоритму, доречність використання елементів інтерфейсу, тощо) (1-9 балів).

Оцінка всієї лабораторної роботи знаходиться підсумовуванням балів за кожний з показників.

Модульний контроль здійснюється 2 рази за семестр.

Поточний контроль за змістовими модулями 1 і 2 проводиться на 8-ому і 15-ому тижнях 1-го семестру відповідно. Поточний контроль за модулями 3 і 4 проводиться на 9-ому і 17-ому тижнях 2-го семестру відповідно.

Контрольно-модульна робота складається з контрольних питань та завдання.

Оцінювання контрольних питань розподіляється пропорційно їх кількості.

Для допуску до підсумкового контролю студент повинен виконати графік навчального процесу, усі види запланованих завдань і протягом семестру отримати в сумі не менше 50 балів за кожен зі змістових модулів.

Семестровий контроль здійснюється у формі заліку після першого семестру вивчення дисципліни і екзамену після другого семестру вивчення дисципліни для всіх форм навчання.

У разі виконання студентом усіх видів поточних контрольних заходів залік виставляється студенту на підставі зарахованих протягом семестру балів.

На заліку і екзамені студент заочної форми навчання повинен мати при собі виконану контрольну роботу та змінний носій з виконаними лабораторними роботами.

Результати екзамену оцінюються за 15-бальною шкалою. Підсумкова оцінка з дисципліни розраховується як середня арифметична чи зважена з оцінок по модулях, включаючи екзаменаційну.

У відомість оцінка проставляється як у балах національної шкали, так і за шкалою ECTS:

При наявності у здобувачів результатів неформального навчання за освітнім компонентом «Основи програмування» у повному обсязі, визнання та оцінювання результатів здійснюється відповідно до «Положення про порядок визнання у Криворізькому національному університеті результатів навчання, отриманих в умовах неформальної освіти». У випадку, якщо за підсумками визнання результатів неформального навчання визнається тільки частина результатів навчання, заявнику зараховуються окремі види навчальної роботи за освітнім компонентом «Основи програмування».

Нижче наведені окремі види навчальної роботи, які можуть бути зараховані здобувачеві при наявності сертифікату про успішне проходження рекомендованих онлайн курсів.

Модулі	Посилання на рекомендовані курси
Змістовий модуль 1. Основи структурного програмування мовою С++	https://www.coursera.org/specializations/coding-for-everyone https://www.coursera.org/learn/cpp-basic-structures-vectors-pointers-strings-and-files https://www.coursera.org/learn/codio-cpp-basics https://courses.prometheus.org.ua/courses/course-v1:Prometheus+CS50+2019_T1
Змістовий модуль 2. Функції та управління пам’яттю у мові С++	https://www.coursera.org/specializations/coding-for-everyone https://www.coursera.org/learn/cpp-basic-structures-vectors-pointers-strings-and-files https://www.coursera.org/learn/codio-cpp-basics https://courses.prometheus.org.ua/courses/course-v1:Prometheus+CS50+2019_T1
Змістовий модуль 3. Об’єктно-орієнтоване програмування на С++	https://www.coursera.org/learn/cs-fundamentals-1 https://courses.prometheus.org.ua/courses/course-v1:Prometheus+CS50+2019_T1/
Змістовий модуль 4. Програмування мовою Python	https://training.epam.ua/Training/Details/3566?lang=ua https://foxminded.ua/python-start-1/ https://www.coursera.org/learn/python-programming-intro https://courses.prometheus.org.ua/courses/course-v1:Prometheus+CS50+2019_T1/

ШКАЛА ОЦІНЮВАННЯ

Національна шкала успішності	Оцінка ECTS	Визначення ECTS	100-бальна система оцінювання
відмінно/ зараховано	**A**	ВІДМІННО - відмінне виконання лише з незначними помилками	90...100
добре/ зараховано	**B**	ДУЖЕ ДОБРЕ - вище середнього рівня з кількома помилками	80...89
	C	ДОБРЕ - у цілому правильно робота з певною кількістю помилок і недоліків	71...79
задовільно/ зараховано	**D**	ЗАДОВІЛЬНО - непогано, але зі значною кількістю грубих помилок	61...70
	E	ДОСТАТНЬО - виконання задовольняє мінімальні потреби	50...60
незадовільно/ не зараховано	**FX**	НЕЗАДОВІЛЬНО - із можливістю повторного складання	30...49
	F	НЕЗАДОВІЛЬНО - з обов'язковим повторним вивчення дисципліни	0...29

11. ПЕРЕЛІК ПИТАНЬ ДЛЯ ПІДСУМКОВОГО КОНТРОЛЮ ЗНАНЬ, УМІНЬ, НАВИЧОК

Питання С++

1. Основи синтаксису мови Сі ++, структура консольного застосунку.
2. Базові типи даних (bool, char, int, double). Визначення змінних та констант. Оператор sizeof ().
3. Вирази, операції, коментарі. Приведення типів.
4. Пріоритет операторів в виразах.
5. Блоки і правила видимості змінних.
6. Структурні оператори: умовний оператор (if, goto), оператор множинної альтернативи (switch), цикли (while, do, for).
7. Математичні функції стандартної бібліотеки С++ (<cmath>).
8. Масиви. Передача масивів у функції.
9. Визначення функції. Прототип функції. Рекурсія.
10. Роздільна компіляція програмних модулів. Використання * .h файлів.
11. Зовнішні (extern) і статичні (static) глобальні змінні і функції.
12. Покажчики та оператори, з ними пов'язані.
13. Покажчик на функцію.
14. Функції роботи з динамічною пам'яттю. Динамічні масиви.
15. Рядки С. Функції роботи з рядками.
16. Призначені для користувача типи даних (enum, struct, union).
17. Складні декларації, визначення синонімів типів (typedef).
18. Директиви препроцесора і їх використання.
19. Що таке: інкапсуляція, успадкування, поліморфізм? Пояснити механізм реалізації кожного з принципів об'єктно-орієнтованого програмування в синтаксисі мови програмування С ++.
20. Класи. Конструктори, деструктори. (Визначити клас, визначити конструктор і деструктор в ньому). Чи може в класі бути кілька деструкторів? Кілька конструкторів?
21. Перевантаження операторів і функцій. Особливості перевантаження і повернення значень в операторах «=» і «+». Перевантаження оператора «+ =».
22. Перевантаження операторів введення і виведення, що працюють з потоками С ++.
23. Перевантаження оператора приведення типу.
24. Оператор «::». Визначення тіла методу поза класом.
25. Поліморфізм, віртуальні функції. Проілюструвати різницю в роботі звичайного і віртуального методів.
26. Віртуальний деструктор. Навіщо застосовується?
27. Динамічна пам'ять. Оператори new і delete.
28. Дружні класи і функції (friend). Навіщо застосовуються?
29. Значення аргументів функцій за замовчуванням. Як задати? Чи завжди можна задати за замовчуванням перший аргумент функції?
30. Одиночне спадкування. Проілюструвати роботу успадкованого, перевизначеного і нового методів в похідному класі.
31. Інкапсуляція. Права доступу до членів класу: private, protected, public. Проілюструвати різницю.
32. Множинне спадкування.
33. Чисто віртуальні функції. Що таке? Навіщо застосовуються?

34. Абстрактні класи.
35. Передача і повернення параметрів в функції за значенням, за посиланням, за вказівником.
36. Потоки введення-виведення C ++ cin і cout і їх використання. Маніпулятори потоків endl, flush. Зміна формату виведення цілих і дійсних чисел.
37. Потоки введення-виведення C ++, що визначаються в заголовних файлах: <iostream>, <fstream>, <sstream>. Призначення, особливості використання.
38. Файлове введення-виведення в C ++. Відкриття потоку, пов'язаного з файлом на читання, запис. Закриття потоку.
39. Посилання. Чим відрізняються від покажчиків? Чи потрібно форматувати посилання при їх оголошенні? Робота оператора присвоювання з посиланням. Чи можна повернути посилання з функції на змінну, оголошену в цій функції?
40. Статичні методи і поля класу.
41. Покажчики. Оператори взяття адреси та взяття значення за адресою.
42. Простори імен (namespace, using). Простір імен std.

Питання Python

1. Типи даних.
2. Змінні.
3. Числові типи даних. Операції над числовими типами даних.
4. Рядки. Рядки unicod.
5. Введення даних. Виведення даних.
6. Форматне введення / виведення.
7. Списки. Вирази в списках.
8. Оператор del.
9. Використання списків, як стеків.
10. Використання списків, як черг.
11. Операції порівняння для списків.
12. Діапазони.
13. Кортежі. Відмінність кортежів від словників.
14. Словники.
15. Оператор if. Особливості операторів порівняння.
16. Оператори циклу. Оператор for. Оператор while. Завершення циклу.
17. Продовження циклу. Оператор pass.
18. Визначення функції.
19. Простір імен функції.
20. Передача параметрів. Ключі.
21. Передача в функцію змінного числа аргументів.
22. Елементи функціонального програмування.
23. Використання лямбда-функцій.
24. Функції роботи зі структурами даних.
25. Функція map (). Приклади застосування.
26. Функція filter (). Приклади застосування.
27. Функція reduce (). Приклади застосування.
28. Документування функцій.
29. Створення модулів.
30. Пошук модулів.
31. Компіляція модулів на Python.

32. Стандартні модулі Python.
33. Пакети і файлова система.
34. Клас File.
35. Відкриття файлу.
36. Методи класу для File введення-виведення.
37. Взаємодія з файлової системою.
38. Модуль path.
39. Об'єкти і файловий введення-виведення.
40. Визначення класу.
41. Управління атрибутами і методами класу.
42. Визначення об'єктів.
43. Множинне спадкування.

Приклад екзаменаційного білету

1. Пріоритет операторів в виразах.
2. Кортежі в Python. Відмінність кортежів від словників.
3. Скласти таблицю імен змінних, алгоритм розв’язання задачі та програми мовами C++ та Python.

Обчислити значення квадратів парних чисел натурального ряду від K до M.

12. НАВЧАЛЬНО-МЕТОДИЧНІ МАТЕРІАЛИ З ДИСЦИПЛІНИ

12.1 Навчальна та довідкова література

1. Основи алгоритмізації та програмування мовами С++, Visual Basic, Turbo Pascal: навч. посіб. / Азарян А. А., Карабут Н. О., Козикова Т. П., Рибальченко О. Г., Трачук А. А., Шаповалова Н. Н. Кривий Ріг, 2014. 308 с.
2. Schildt H. C++ The Complete Reference. 5th Edition. McGraw-Hill Osborne Media. 2014. 805 p.
3. Rao S. C++ in One Hour a Day, Sams Teach Yourself. 9th Edition. Sams Publishing, 2022. 848 p.
4. Booch G., Engle M.W., Young B.J., Houston K.A. Object-Oriented Analysis and Design with Applications. 3rd Edition. Addison-Wesley Professional, 2017. 720 p.
5. Базові алгоритми та основи програмування. Теорія і практика: навч. посіб. / О.Є. Тверитникова, В. А. Крилова, О. Г. Васильченков. Нац. техн. ун-т "Харків. політехн. ін-т". Харків : Панов А. М., 2020. 264 с.
6. Марк Лутц. Вивчаємо Python. 5-е видання. Київ: Діалектика, 2019, 832 с.
7. Васильєв О. Програмування мовою Python. Тернопіль: Навчальна книга - Богдан, 2019. 504 с.
8. Зеленяк О. Програмування мовою Python. Алгоритмічні структури і стратегії. Теорія та практика. Київ: Ліра-К, 2024. 338 с.

12.2. Методична література

1. Рибальченко О.Г., Білашенко С.В. Методичні вказівки до виконання лабораторних робіт з дисципліни «Основи програмування, 1 частина» для студ. всіх форм навч. за спец. 121 «Інженерія програмного забезпечення». Кривий Ріг, 2023. 72 с.

2. Рибальченко О.Г., Білашенко С.В. Методичні вказівки до виконання лабораторних робіт з дисципліни «Основи програмування, 2 частина» для студ. всіх форм навч. за спец. 121 «Інженерія програмного забезпечення». Кривий Ріг, 2024. 48 с.

13. ІНФОРМАЦІЙНІ РЕСУРСИ

До складу інформаційних ресурсів навчальної дисципліни входять:

1. Бібліотека Криворізького національного університету. URL: http://lib.knu.edu.ua/ (дата звернення 20.12.2024).

Internet-ресурси:

2. Рибальченко О.Г. Основи програмування С++: консп. лекцій. URL: https://classroom.google.com/c/MTYzMzMxMDIwNTMy?hl (дата звернення 20.12.2024).

3. Рибальченко О.Г. Основи програмування Python: консп. лекцій. URL: https://classroom.google.com/c/NTMzNDU5NDg2MDFa?hl (дата звернення 20.12.2024).

4. A Byte of Python. URL: https://python.swaroopch.com/ (дата звернення 20.12.2024)/

5. Dive Into Python. URL: https://diveintopython.org/ (дата звернення 20.12.2024).

14. ТЕРМІНОЛОГІЧНИЙ СЛОВНИК

Алгоритм (Algorithm) - is a set of instructions that describe the order of actions of the performer to achieve the result of solving the problem in a finite number of actions; a system of rules for the execution of a discrete process that achieves the set goal in a finite time. Block diagrams are often used to visualize algorithms .

Блок-схема (Block scheme) - Representation of an algorithm for solving or analyzing a problem using geometric elements (blocks) that denote operations, flow, data, etc. It is customary to mark the input and output data block with a parallelogram , the data calculation (processing) block with a rectangle , the decision-making block with a rhombus , and the beginning and end of the algorithm with an ellipse .

Математична модель (Mathematical model) - is a system of mathematical relationships that describe the process or phenomenon being studied . The mathematical model is important for such sciences as: economics , ecology , sociology , physics , chemistry , mechanics , computer science , biology , etc. Any mathematical tools can be used to create mathematical models - differential or integral equations, set theory , abstract algebra , mathematical logic , probability theory , graphs , etc. The process of creating a mathematical model is called mathematical modeling. This is the most general and most widely used research method in science, in particular, in cybernetics .

Типи значень (Value types) - a value type is a data type that directly stores its value in memory, rather than storing a reference to an object that contains the value. Examples of value types in C# include integral types (such as int, long, and byte), floating-point types (such as float and double), and the bool, char, and decimal types. Value types are typically smaller and more efficient than reference types, as they do not require memory allocation on the heap and do not need to be garbage collected. In C#, value types are passed by value, meaning that a copy of the value is passed to a method or assigned to a variable, rather than a reference to the original value. However, value types can also be boxed, which means that they can be wrapped in a reference type and treated as objects.

Структура даних (Data structure) - in programming and computer science, a data structure is a way of organizing data in computers . Often, a specific list of operations that can be performed on data organized into such a structure is associated with the data structure .The correct selection of data structures is extremely important for the effective functioning of the relevant algorithms for their processing. Well-constructed data structures allow you to optimize the use of machine time and computer memory to perform the most critical operations.

Інструкція (Instruction) - in computer programming, an instruction is a syntactic unit of an imperative programming language that indicates a certain action to be performed. A program written in this language is a sequence of instructions. An instruction can have internal components (for example, expressions).

Умовний перехід (Conditional transition) – is a construction of the programming language that allows you to perform various actions depending on the boolean value of the condition specified by the programmer. Most often, a conditional transition has two stages: in the first, some values that determine the transition condition are compared, and in the second, the transition itself is performed.

Безумовний перехід (Unconditional transition) - transition to a given point of the program without checking the fulfillment of any conditions. In many programming languages, this transition corresponds to a special goto instruction.

Багатоваріантний вибір (Switch statement) - also known as the selection instruction and the switch operator is a special type of programming language instruction that provides multidirectional (multiple) branching in the program. The name of the instruction may vary in different languages. This selection mechanism exists in most imperative programming languages.

Цикл (Cycle) - is a type of control structure in high-level programming languages , designed to organize repeated execution of a set of instructions (commands). A loop can also be any repeatedly executed sequence of commands, organized in any way (for example, using a conditional transition).

Ітерація (Iteration) - is a multi-meaning term that, depending on the context, can mean: repeated use of a mathematical operation (with changed data) when solving computational problems, which makes it possible to gradually approach the correct result or the result of multiple repetition of some mathematical operation.

Ітераційний цикл (Iterative cycle) - a loop operator for which the number of iterations of the loop body is unknown in advance. In iterative cycles, at each step of the calculations, the condition for achieving the desired result is successively approximated and checked. Exit from the iteration cycle is carried out in case of fulfillment of the given condition.

Цикл з передумовою (A loop with a precondition) - is a loop that is executed until some condition specified before its start is true. This condition is checked before the execution of the loop body begins, so the body may not be executed once (if the condition is initially false). In most procedural languages, programming is carried out using the instruction while, hence its second name is the while loop.

Цикл з післяумовою (A loop with a postcondition) - is a loop in which the condition is checked after the loop body is executed. It follows that the loop body is always executed at least once.

Цикл з лічильником (A cycle with a counter) - is a loop in which some variable changes its value from a given initial value to a final value in some increments, and for each value of that variable the body of the loop is executed once. In most procedural languages, programming is implemented with a statement forthat specifies a counter (called a "loop variable"), the required number of passes (or the limit value of the counter), and possibly the step by which the counter changes.

Вкладені цикли (Nested loops) - It is possible to create a loop inside the body of the second loop. Such a loop is called a nested loop . A nested loop relative to the loop in whose body it is nested will be called an inner loop , and conversely, a loop in the body of which there is a nested loop will be called outer relative to the nested loop. Inside a nested loop can be the next nested loop, forming the next level of nesting and so on. The number of nesting levels, as a rule, is not limited.

Масив (Array) - is an ordered set of a fixed number of elements of the same type, stored in sequentially located RAM cells, having a sequence number and a common name provided by the user.

Характеристики масиву (Characteristics of the array) - dimensionality — the number of element indices (one-dimensional, two-dimensional, ..., multidimensional); size is the total number of elements in the array; arrays can be numeric, character and other types; each language has its own rules for describing arrays.

Рекурсія (Recursion) - in computer science, recursion is a method of solving a computational problem in which the solution relies on the solutions of smaller cases of the same problem. Recursion solves such recursive problems using functions or procedures that call themselves. This approach can be applied to many problems, and recursion is one of the central ideas of computer science.

Алгоритм сортування (A sorting algorithm) - is an algorithm that solves the sorting problem , that is, arranges a linear list (array) of elements. For the sorting algorithm (as for any other modern algorithm), the main characteristics are: the time required to arrange an n-element array, the need for additional memory for sorting, stability — stable sorting does not change the relative arrangement of elements with the same keys.

Алгоритм пошуку (A search algorithm) - is an algorithm that solves a search problem , that is, finds information that is stored in a certain data structure . Data structures can be implemented using linked lists , arrays , search trees , hash tables , or other information storage methods. The direct search algorithm depends on the data structure for which it is implemented. Very often, the search algorithm includes special commands that specify the data structure.

Абстрактний тип даних (Abstract data type (ADT)) - is a mathematical model for data types, defined by its behavior (semantics) from the point of view of a user of the data, specifically in terms of possible values, possible operations on data of this type, and the behavior of these operations. This mathematical model contrasts with data structures, which are concrete representations of data, and are the point of view of an implementer, not a user. ADTs are a theoretical concept, used in formal semantics and program verification and, less strictly, in the design and analysis of algorithms, data structures, and software systems. Most mainstream computer languages do not directly support formally specifying ADTs.

Ітератор (Iterator) - is an object that progressively provides access to each item of a collection, in order. A collection may provide multiple iterators via its interface that provide items in different orders, such as forwards and backwards. An iterator is often implemented in terms of the structure underlying a collection implementation and is often tightly coupled to the collection to enable the operational semantics of the iterator.

Рядок (String) - traditionally a sequence of characters, either as a literal constant or as some kind of variable. The latter may allow its elements to be mutated and the length changed, or it may be fixed (after creation). A string is generally considered as a data type and is often implemented as an array data structure of bytes (or words) that stores a sequence of elements, typically characters, using some character encoding. String may also denote more general arrays or other sequence (or list) data types and structures. When a string appears literally in source code, it is known as a string literal or an anonymous string. In formal languages, which are used in mathematical logic and theoretical computer science, a string is a finite sequence of symbols that are chosen from a set called an alphabet.

Коллекція (Collection, Abstract data type) - a grouping of some arbitrary number of data elements (possibly zero) that have some common meaning for the problem being solved, and that need to be worked with together in some controlled way. Typically, the data elements will be of the same type or, in languages that support inheritance, will be descended from a common ancestor type. A collection is a concept applicable to abstract data types , and does not imply a specific implementation as a concrete data structure, although there is usually a choice.

Асоціативний масив (Associative Array), map, symbol table, or dictionary is an abstract data type that stores a collection of (key, value) pairs, such that each possible key appears at most once in the collection. In mathematical terms, an associative array is a function with finite domain. It supports 'lookup', 'remove', and 'insert' operations.

Список (List, Abstract data type) - a collection of items that are finite in number and in a particular order. An instance of a list is a computer representation of the mathematical concept of a tuple or finite sequence. A list may contain the same value more than once, and each occurrence is

considered a distinct item. A singly-linked list structure, implementing a list with three integer elements. The term list is also used for several concrete data structures that can be used to implement abstract lists, especially linked lists and arrays. In class-based programming, lists are usually provided as instances of subclasses of a generic "list" class, and traversed via separate iterators.

Кортеж (Tuple) - a finite sequence of objects. Sometimes, the finite sequence is also called an ordered list. This means that the order of the objects matter. In a tuple, the objects are either enclosed within parentheses or within angle Each of the objects in the list has a certain type. A tuple consisting of n entries is called an n-tuple. Tuples are used to describe mathematical objects that are made of certain, well-defined components. They are also used very frequently with databases.

Множина (Set, Abstract data type) - an abstract data type that can store unique values, without any particular order. It is a computer implementation of the mathematical concept of a finite set. Unlike most other collection types, rather than retrieving a specific element from a set, one typically tests a value for membership in a set. Some set data structures are designed for static or frozen sets that do not change after they are constructed. Static sets allow only query operations on their elements - such as checking whether a given value is in the set, or enumerating the values in some arbitrary order. Other variants, called dynamic or mutable sets, allow also the insertion and deletion of elements from the set. A multiset is a special kind of set in which an element can appear multiple times in the set.

Словник (Dictionary, associative array, map, symbol table) - an abstract data type that stores a collection of (key, value) pairs, such that each possible key appears at most once in the collection. In mathematical terms, an associative array is a function with finite domain. It supports 'lookup', 'remove', and 'insert' operations. The dictionary problem is the classic problem of designing efficient data structures that implement associative arrays. The two major solutions to the dictionary problem are hash tables and search trees. It is sometimes also possible to solve the problem using directly addressed arrays, binary search trees, or other more specialized structures. Many programming languages include associative arrays as primitive data types, while many other languages provide software libraries that support associative arrays. Content-addressable memory is a form of direct hardware-level support for associative arrays.

Декоратор (Decorator) - is any callable Python object that is used to modify a function, method or class definition. A decorator is passed the original object being defined and returns a modified object, which is then bound to the name in the definition. Python decorators were inspired in part by Java annotations, and have a similar syntax; the decorator syntax is pure syntactic sugar, using @ as the keyword.

Додаток до робочої програми
Робочий план

з дисципліни «Основи програмування»
для здобувачів вищої освіти першого (бакалаврського) рівня вищої освіти
за спеціальністю 121 «Інженерія програмного забезпечення»

Перший семестр

Вид навчальної роботи	Годин у семестрі/кредитів	Розподіл годин по тижнях																Вид підсумкового контролю
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
Лекційні заняття	32	2 ПК	2 ПК	2 ПК	2 ПК	2 ПК	2 ПК	2 ПК	2 ПК	2 ПК	2 ПК	2 ПК	2 ПК	2 ПК	2 ПК	2 ПК	2 ПК	залік
Лабораторні заняття	32	2 ПК	2 ПК	2 ПК	2 ПК	2 ПК	2 ПК	2 ПК	2 ЗМ	2 ПК	2 ПК	2 ПК	2 ПК	2 ПК	2 ПК	2 ПК	2 ЗМ	
Самостійна робота	116	7	7	7	7	7	7	8	8	7	7	7	7	7	7	8	8	
Всього годин/ кредитів	180/6	11	11	11	11	11	11	12	12	11	11	11	11	11	11	12	12	

Другий семестр

Вид навчальної роботи	Годин у семестрі/кредитів	Розподіл годин по тижнях																Вид підсумкового контролю
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
Лекційні заняття	36	2 ПК	2 ПК	2 ПК	2 ПК	2 ПК	2 ПК	2 ПК	2 ПК	2 ПК	2 ПК	2 ПК	2 ПК	2 ПК	2 ПК	2 ПК	2 ПК	екзамен
Лабораторні заняття	36	2 ПК	2 ПК	2 ПК	2 ПК	2 ПК	2 ПК	2 ПК	2 ЗМ	2 ПК	2 ПК	2 ПК	2 ПК	2 ПК	2 ПК	2 ПК	2 ЗМ	
Самостійна робота	108	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	
Всього годин/ кредитів	180/6	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	

Позначки:

ПК - поточний контроль
ЗМ - складання змістових модулів

ЗМІНИ ТА ДОПОВНЕННЯ ДО РОБОЧОЇ ПРОГРАМИ

№ з/п	Дата внесення змін	Зміст змін та доповнень	Підстава до внесення змін (№ і дата наказу, рішення вченої ради, засідання кафедри)
1	2	3	4

Схвалено на засіданні кафедри
_____

Протокол № __ від “ ___ “ _____201 р.

Завідувач кафедри _____

Схвалено на засіданні кафедри
_____

Протокол № __ від “ ___ “ _____ 201 р.

Завідувач кафедри _____

Схвалено на засіданні кафедри
_____

Протокол № __ від “ ___ “ _____201 р.

Завідувач кафедри _____

Схвалено на засіданні кафедри
_____

Протокол № __ від “ ___ “ _____ 201 р.

Завідувач кафедри _____