Міністерство освіти і науки України
Черкаський державний технологічний університет
Факультет інформаційних технологій і систем
Кафедра комп'ютерних наук та системного аналізу

Методичні рекомендації

до написання курсового проєкту
з курсу «Алгоритмізація та програмування»
для студентів спеціальності
F3 – комп'ютерні науки

Черкаси - 2026

УДК 004.65 (07)
М54

Затверджено вченою радою ФІТІС, протокол № 9 від 26.02.2026 р., згідно з рішенням кафедри комп'ютерних наук та системного аналізу, протокол №9 від 05.01.2026 р.

Упорядник: Максимов А.Є., *PhD з комп'ютерних наук, викладач кафедри комп'ютерних наук та системного аналізу*

Рецензент: Заспа Г.О., *кандидат технічних наук, доцент, доцент кафедри програмного забезпечення автоматизованих систем*

М54 Методичні рекомендації до написання курсового проєкту з курсу «Алгоритмізація та програмування» для студентів спеціальності F3 – «Комп'ютерні науки» усіх форм навчання [Електронний ресурс] / [упоряд. Максимов А.Є., за ред. Триуса Ю.В.]; М-во освіти і науки України, Черкас. держ. технол. ун-т. Черкаси: ЧДТУ, 2026. 51 с.

Мета методичних рекомендацій – допомога студентам щодо написання курсового проєкту з курсу «Алгоритмізація та програмування».

УДК 004.42 (07)

Виробничо-практичне
електронне видання
комбінованого використовування

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ

до написання курсового проєкту з курсу
«Алгоритмізація та програмування»
для здобувачів освітнього ступеня «бакалавр»
галузі знань F «Інформаційні технології»
спеціальності F3 «Комп'ютерні науки»
освітньої програми «Комп'ютерні науки та прикладне програмування»
усіх форм навчання

Упорядники: **Максимов** Антон Євгенійович
В авторській редакції

ЗМІСТ

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ, ЦІЛІ ТА ЗАДАЧІ КУРСОВОГО ПРОЄКТУ

Курсовий проєкт (КП) з дисципліни «Алгоритмізація та програмування» – це самостійно виконана практична робота, спрямована на проєктування, реалізацію та тестування програмного забезпечення, яке забезпечує обробку, зберігання та аналіз даних у межах заданої предметної області.

У процесі виконання курсового проєкту студент здійснює розробку алгоритмів, вибір та використання структур даних, реалізацію програмних модулів мовою програмування Python, а також створення користувацького інтерфейсу у вигляді консольного, графічного або веб-застосунку. Після чого проводиться тестування розробленої системи.

Виконаний курсовий проєкт має продемонструвати вміння студента алгоритмічно мислити, застосовувати базові та розширені конструкції мови програмування, працювати з файлами, модулями, бібліотеками, а також виконувати аналіз ефективності алгоритмів.

На початкових етапах виконання курсового проєкту здійснюється аналіз предметної області, формулювання задачі та визначення функціональних вимог до програмного забезпечення.

Далі виконується проєктування алгоритмів, визначення необхідних структур даних, розробка логіки обробки інформації та побудова блок-схем алгоритмів.

Практична реалізація програмного проєкту передбачає створення програмних модулів мовою програмування Python, організацію зберігання даних із використанням файлів структурованих форматів, реалізацію алгоритмів обробки даних, а також розробку користувацького інтерфейсу у вигляді консольного, графічного або веб-застосунку із застосуванням відповідних бібліотек та фреймворків.

При виконанні курсового проєкту студент повинен продемонструвати вміння самостійно працювати і застосовувати на практиці теоретичні знання, отримані під час вивчення дисципліни.

Основною метою виконання курсового проєкту є закріплення студентами теоретичних знань з алгоритмізації та програмування і набуття практичних навичок проєктування, реалізації, тестування та аналізу програмного забезпечення мовою програмування Python для розв'язання задач у заданій предметній області.

Задачі курсового проєкту:

- систематизація та поглиблення теоретичних знань з основ алгоритмізації, програмування та роботи зі структуруваними даними;
- формування практичних навичок розробки алгоритмів, використання базових і складних структур даних та стандартних бібліотек мови програмування Python;
- закріплення вмінь реалізації програмних модулів, роботи з файлами та обробки помилок;
- набуття досвіду створення користувацьких інтерфейсів у вигляді консольних, графічних або веб-застосунків;
- розвиток навичок аналізу алгоритмічної складності, застосування алгоритмів пошуку, сортування та рекурсивних обчислень;
- формування вмінь тестування, налагодження та профілювання програмного коду;
- розвиток навичок інженерного розв'язання задач програмування з використанням сучасних комп'ютерних технологій, науково-технічної літератури та ресурсів мережі Internet.

У результаті виконання КР студенти повинні отримати *практичні навички* та *вміння:*

- аналізувати предметну область та формулювати функціональні вимоги до програмного забезпечення;

- розробляти концептуальні алгоритмічні рішення для вирішення задач предметної області;
- будувати блок-схеми алгоритмів та описувати логіку роботи програмних модулів;
- вибирати та реалізовувати структури даних для ефективного зберігання та обробки інформації;
- розробляти та впроваджувати алгоритми обробки даних, включаючи пошук, сортування та рекурсивні обчислення;
- оцінювати алгоритмічну складність реалізованих рішень та оптимізувати код за необхідності;
- створювати модульну структуру програми, реалізовувати окремі модулі та функції, забезпечуючи розділення логіки;
- розробляти користувацький інтерфейс (консольний, графічний або веб-застосунок), забезпечуючи взаємодію з користувачем;
- тестувати та налагоджувати програмне забезпечення, здійснювати обробку винятків та профілювання коду;
- документувати реалізацію програми та алгоритмів, оформлювати пояснювальну записку, що відображає структуру, алгоритми та призначення програмних модулів.

2 ПОРЯДОК, ЗАВДАННЯ ТА ВИМОГИ ДО ВИКОНАННЯ КУРСОВОГО ПРОЄКТУ

Тема курсового проєкту обирається студентом з переліку, запропонованого в п. 4, самостійно чи за рекомендацією викладача з урахуванням рівня підготовки студента. У випадку практичної участі студента в науково-дослідних роботах, що ведуться на кафедрі чи в університеті, або за місцем майбутньої роботи студента і мають безпосереднє відношення до дисципліни, студент може запропонувати власне формулювання теми курсового проєкту, узгодивши її з керівником КП.

Робота виконується індивідуально кожним студентом. В окремих випадках допускається об’єднання студентів у групи (2-3 студенти) для роботи над складними темами.

Завдання полягає у розробці програмного забезпечення інформаційної системи для заданої предметної області, що передбачає:

- реалізацію алгоритмів обробки даних;
- використання відповідних структур даних;
- створення програмних модулів, що виконують конкретні функції;
- організацію взаємодії модулів та об’єктів системи через головне меню або ієрархію підменю;
- створення користувацького інтерфейсу у вигляді консольного, графічного або веб-застосунку;
- документування програмної реалізації та алгоритмів.

Розробка програмного забезпечення інформаційної системи повинна виконуватися з урахуванням ефективного використання ресурсів та цілісності даних. Програмна система повинна забезпечувати:

- введення та збереження нових даних у програми або файли;
- перегляд та редагування існуючих даних;
- видалення даних при необхідності;

– виконання алгоритмів обробки та сортування даних;

– взаємодію з користувачем через консольні або графічні форми;

– генерацію звітів або підсумкових результатів обробки даних.

Обов'язкові вимоги до виконання курсового проєкту: створення концептуальної моделі програми; розробка структури даних та відповідних модулів з алгоритмами обробки даних; створення інтерфейсу користувача; проведення тестування та забезпечення модульності програми.

Створення *концептуальної моделі програми* включає в себе:

– опис предметної області та вимог користувачів;

– побудову блок-схем алгоритмів (не менше 2).

Розробка *структури даних та відповідних модулів* включає в себе:

– використання різних типів колекцій Python (списки, словники, множини, кортежі);

– реалізація **не менше 5 функцій/модулів**, що виконують окремі завдання;

– розробка *головного меню* та підменю для взаємодії користувача з модулями.

Розробка *алгоритмів обробки даних* включає в себе реалізацію **не менше 6 алгоритмів**, включно з:

– 2 – сортування без використання сторонніх бібліотек;

– 2 – пошук (лінійний та бінарний);

– 1 – рекурсивне обчислення;

– 1 – робота з файлом.

Створення *інтерфейсу користувача* включає в себе реалізацію **не менше 5 форм виведення інформації**, у т. ч.:

– 1 – головне меню з підменю (не менше 3 рівнів, загальна кількість опцій ≥ 10);

– 1 – введення нових даних з перевіркою коректності;

– 2 – вкладені форми/виводи з додатковими опціями;

– 1 – графічне або текстове відображення результатів алгоритмів.

Проведення *тестування та забезпечення модульності програми* включає в себе реалізацію:

– **не менше 5 тестів/функцій перевірки**;

– **не менше 2 модулів**, інтегрованих з головним меню або підменю;

– документування програмних модулів та алгоритмів.

Примітка. Вище наведено орієнтовний перелік компонентів курсового проєкту. Він може бути змінений за погодженням з керівником, але не може поступатися наведеному за трудомісткістю.

Результатами роботи мають бути:

1. Інформаційна система, реалізована мовою програмування Python, яка забезпечує обробку та аналіз даних у заданій предметній області та містить алгоритми, структури даних, програмні модулі і користувацький інтерфейс (консольний, графічний або веб-застосунок).

2. Пояснювальна записка до курсового проєкту за структурою, що наведена нижче (див. п. 3), підготовлена в MS Word.

3 ОФОРМЛЕННЯ КУРСОВОГО ПРОЄКТУ

Обсяг пояснювальної записки до курсового проєкту, незалежно від змісту, складає **25-35** сторінок. При комп'ютерному наборі тексту використовувати шрифт **Times New Roman**, кегль – **14**, друк через **1,5** інтервалу. Границі тексту – **30** мм від лівого краю аркуша, **15** мм від правого, по **20** мм від верхнього та нижнього країв.

Тексту роботи має передувати *титульна сторінка*, *підписане керівником завдання*, та *зміст*. Титульна сторінка оформлюється за зразком, наведеним у додатку А. Зміст повинен містити назви усіх розділів із зазначенням сторінки початку матеріалу розділу.

У кінці текстової частини наводиться *список використаних джерел* та *додатки*. Список використаних джерел повинен бути оформлений відповідно до державних стандартів (див., наприклад, [14]).

3.1 Структура пояснювальної записки до курсового проєкту

Пояснювальна записка повинна мати такі розділи:

Вступ

1. Опис предметного середовища та постановка задачі.

2. Проєктування програмного забезпечення. Алгоритмічні рішення та блок-схеми алгоритмів.

3. Структури даних та організація зберігання інформації.

4. Опис алгоритмів обробки даних. Пошук, сортування, рекурсивні обчислення та аналіз алгоритмічної складності.

5. Архітектура програмного проєкту та модульна структура. Взаємодія програмних компонентів.

6. Опис реалізації програмного проєкту.

Висновки.

Список використаних джерел.

Додатки

Нижче наведено концептуальні вимоги до змісту кожного з перелічених вище пунктів.

3.2 Титульний аркуш, зміст і вступ

Як вже наводилося вище, титульна сторінка оформлюється за зразком, наведеним у додатку А. Обов'язкові розділи і пункти змісту наведено вище.

У вступі необхідно навести відомості про призначення програмного забезпечення (інформаційної системи, підсистеми, програмного модуля тощо) та сутність задач, які можуть бути розв'язані в процесі його використання.

Необхідно також описати функціональні можливості інформаційної системи, особливості обробки даних та основні алгоритмічні рішення, що застосовуються в проєкті.

Крім того, у вступі слід навести огляд подібних програмних рішень або інформаційних систем, отриманий у результаті аналізу літературних джерел та ресурсів мережі Internet, із коротким порівнянням їх функціональних можливостей.

3.3 Опис предметної області і постановка задачі

Предметна область визначається конкретним контекстом, наприклад: облік студентів, ведення персональних даних користувачів, аналіз результатів експериментів, моделювання процесів або обробка статистичної інформації.

Метою проєкту є створення програмного продукту, який забезпечує автоматизацію рутинних завдань користувачів шляхом реалізації відповідних алгоритмів обробки даних та організації зручного інтерфейсу взаємодії. Система повинна підтримувати введення, редагування, видалення, сортування та пошук даних, а також забезпечувати підготовку підсумкових результатів у вигляді звітів або графічних виводів.

У межах предметної області розрізняють основні інформаційні задачі, що мають бути реалізовані у програмі:

– обробка та зберігання даних у відповідних структурах (списки, словники, множини тощо);
– реалізація алгоритмів сортування, пошуку та фільтрації даних;
– обчислення статистичних або аналітичних характеристик;
– взаємодія користувача з програмою через консольний або графічний інтерфейс;
– модульність і повторне використання алгоритмів шляхом розробки окремих функцій та модулів.

У результаті постановки задачі повинно бути визначено:

1. Об’єкт автоматизації: конкретний набір даних та операцій над ними.
2. Мета розробки: створення функціонального програмного продукту з реалізацією алгоритмів обробки та аналізу даних.
3. Основні функціональні можливості: введення, редагування, видалення, пошук, сортування, підготовка результатів і звітів.
4. Обмеження та вимоги: застосування ефективних структур даних та алгоритмів, модульна структура, організація взаємодії через головне меню або підменю, забезпечення можливості подальшого розширення системи.

Зміст робіт, які повинні бути виконані в цьому пункті.

Аналіз предметного середовища програмного проєкту повинен включати описи або формулювання:

– об’єкта дослідження та характеристику його організаційно-функціональної структури;
– функціональної сутності системи (підсистеми, комплексу задач, окремої задачі);
– опис вхідних даних, у тому числі їх форматів, структур та способів введення (наприклад, через консольні форми, графічні вікна або файли);
– нормативно-довідкової інформації, що використовується у процесі обробки даних (класифікатори, словники, стандарти або інші довідкові джерела);

- основних алгоритмів обробки даних та перетворення інформації, що реалізуються програмою, із зазначенням правил, методик та процедур, які їх регламентують;
- результатів обробки даних (наприклад, підсумкові повідомлення, вивід на екран, файли, графічні або текстові звіти), із визначенням їх структури, періодичності формування та способу представлення користувачу.

На основі проведеного аналізу здійснюється *постановка задачі* курсового проєкту, формулюється *мета роботи* та визначаються *основні вимоги* до:

- функціональної сутності програмного забезпечення (підсистеми, комплексу задач, окремих функцій);
- структури й організації даних, що використовуються в програмному проєкті, включно з їх типами, зв'язками та форматами зберігання;
- програмного забезпечення, яке реалізує алгоритми обробки та перетворення даних, модульну структуру та взаємодію компонентів;
- технічного забезпечення, необхідного для виконання програмного проєкту, зокрема апаратні вимоги та середовище виконання програм.

3.4 Проєктування програмного забезпечення. Блок-схеми алгоритмів та опис логіки роботи програми

Проєктування програмного забезпечення є ключовим етапом курсового проєкту, оскільки визначає структуру програмного продукту, порядок обробки даних та взаємодію компонентів. На цьому етапі реалізується перехід від аналізу предметної області та постановки задачі до конкретних алгоритмічних рішень і структур даних, що забезпечують функціонування системи.

3.4.1 Логіка роботи програми. Логіка роботи програми базується на послідовності виконання основних операцій над даними відповідно до функціональних вимог. Вона включає:

1. Введення даних користувачем через консольний або графічний інтерфейс, а також з файлів у визначених форматах.

2. Валідацію та перевірку коректності введених даних, у тому числі перевірку типів, обов’язкових полів та діапазонів значень.

3. Обробку даних згідно з алгоритмами сортування, пошуку, обчислення підсумкових значень, фільтрації або інших перетворень.

4. Збереження результатів у структурованому вигляді (списки, словники, множини або файли).

5. Вивід результатів користувачу у вигляді звітів, таблиць, графіків або повідомлень на екран.

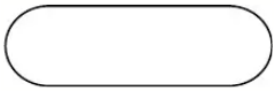
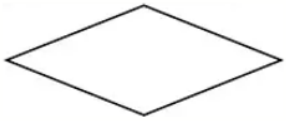
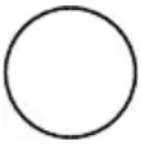
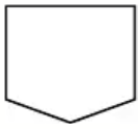
6. Повторне виконання операцій та взаємодію з користувачем через головне меню та підменю, що дозволяє виконувати різні задачі системи без перезапуску програми.

3.4.2 Блок-схеми алгоритмів. *Блок-схема* – це схематичне представлення процесу, системи та комп’ютерного алгоритму. Блок-схеми дуже часто використовують в різноманітних сферах діяльності, щоб документувати, вивчати, планувати, покращувати та пояснювати складні процеси за допомогою простих логічних діаграм. Для наочності та полегшення розробки програмного продукту складаються блок-схеми алгоритмів, які відображають:

- логічну послідовність виконання програмних операцій;
- використання умовних конструкцій (if/elif/else);
- циклічні операції (for, while) для обробки колекцій даних;
- виклики функцій та модулів, що реалізують окремі підзадачі;
- обробку виняткових ситуацій та помилок.

Блок-схеми служать *основою* для реалізації програмного коду, допомагають контролювати правильність логіки роботи та взаємодію різних компонентів. Вони також використовуються для демонстрації роботи програми під час захисту курсового проєкту.

Таблиця 3.1 – Елементи блок-схем та їх опис

Елемент блок-схеми	*Опис елементу блок-схеми*
	**Початок/кінець (Термінатор).** Застосовується для позначення початку та кінця програми. Всередині фігури записується слово «Початок» або «Кінець».
	**Процес (Дія).** Використовується для позначення процесу, дії або функції. Всередині фігури записують безпосередньо самі операції. Позначається символом прямокутника.
	**Документ.** Символізує введення та виведення документа. Під введенням документа може матись на увазі надходження звіту, електронного листа або замовлення. Приклади виводу документів: створення презентації, робочого конспекту або листа.
	**Рішення.** Визначає питання, на яке необхідно надати відповідь (зазвичай «так/ні» або «істина/неправда»). Вхід в елемент позначається лінією, яка входить зазвичай у верхню вершину елементу. На етапі цього елементу блок-схема розділяється у різних напрямках залежно від обраної відповіді. Якщо виходів два чи три, то зазвичай кожен вихід позначається лінією, яка виходить з решти вершин (бічних і нижньої). Якщо виходів більше трьох, то їх слід показувати однією лінією, яка виходить з вершини (частіше нижньої) елемента, яка потім розгалужується. Відповідні результати обчислень можуть записуватися поруч з лініями, які відображають ці шляхи. Позначається символом ромба.
	**Введення/виведення (Дані)**. Символізує дані, які доступні для введення або виведення. Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи). Позначається символом паралелограма.
	**З'єднувач.** Зазвичай застосовується в більш складних схемах для з'єднання окремих блоків в межах однієї сторінки. Елемент відображає вихід в частину схеми і вхід з іншої частини цієї схеми. Використовується для переривання лінії та продовження її в іншому місці. Відповідні сполучні символи повинні мати одне (при тому унікальне) позначення. Символ позначення: коло.
	**Міжсторінковий з'єднувач.** Часто використовується у складних схемах для поєднання окремих блоків, розташованих на окремих сторінках. Для зручності інтерпретації всередині фігури, як правило, вказується номер сторінки.
	**Коментар.** Цей символ дозволяє додати необхідний контекст, роз'яснення або коментар до певного діапазону даних. Коментар також можна приєднати до необхідного розділу блок-схеми за допомогою пунктирної лінії. Позначається фігурною скобкою.

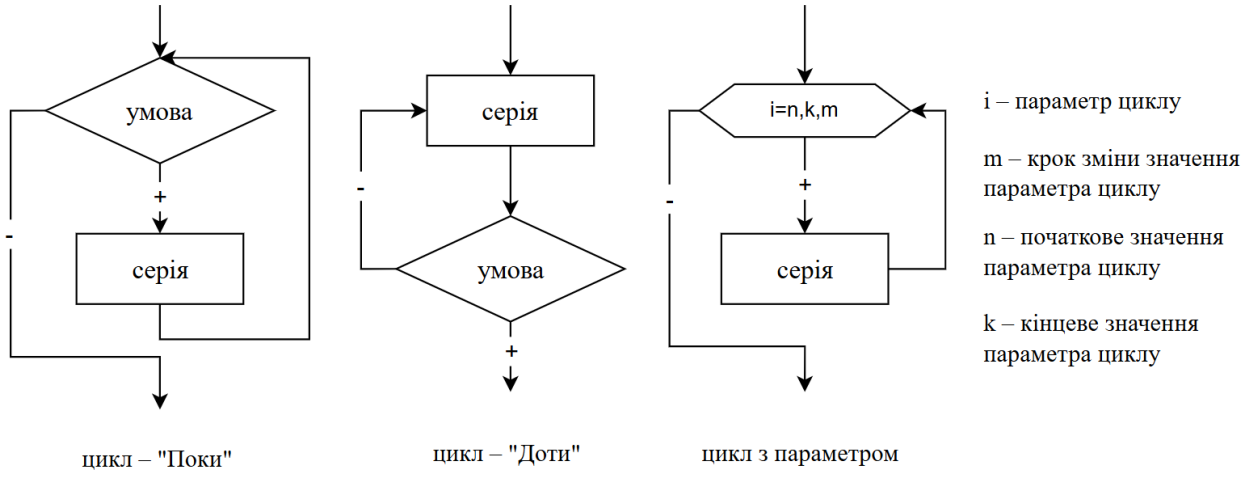


Рисунок 3.1 – Базові структури циклічних алгоритмів

Для побудови блок-схем можна використовувати сервіс DrawIO за посиланням: https://app.diagrams.net/ (рис. 3.2).

Draw.io – це безкоштовний онлайн-інструмент для створення різних типів діаграм: блок-схем, UML-діаграм, ER-моделей, схем баз даних, мережевих схем тощо. Сервіс не потребує встановлення програмного забезпечення, оскільки працює безпосередньо у браузері.

Основні можливості Draw.io:

– зручний графічний інтерфейс – елементи блок-схеми можна легко додавати шляхом перетягування (drag-and-drop) з бібліотеки;
– автоматичне вирівнювання та з'єднання – інструмент дозволяє легко поєднувати блоки стрілками та підтримує прив'язку елементів для розміщення;
– імпорт та експорт – схеми можна зберігати у форматах .png, .jpg, .pdf, .svg, .xml або інтегрувати з Google Drive, GitHub чи OneDrive;
– спільна робота – Draw.io підтримує колективне редагування, що зручно для командних проєктів;
– підтримка стандартних символів блок-схем – початок/кінець, процес, умова, ввід/вивід, цикл тощо.

Переваги використання Draw.io:

– безкоштовність і доступність онлайн;
– не потребує встановлення додаткових програм;

– інтуїтивно зрозумілий інтерфейс;
– можливість швидкого створення професійних діаграм.

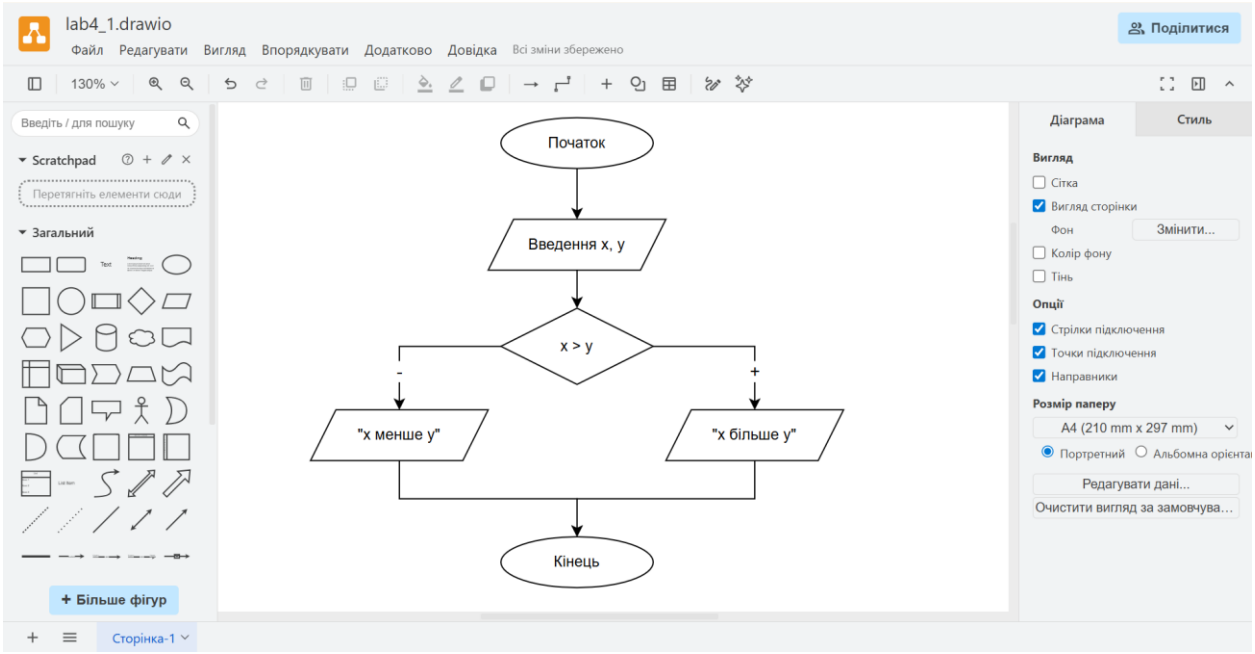


Рисунок 3.2 – Головна сторінка веб-сервісу DrawIO

3.4.3 Принципи та підходи при проєктуванні програмного забезпечення. При проєктуванні програмного забезпечення використовуються такі підходи:

– *модульність* – поділ програми на незалежні функціональні блоки та модулі;
– *повторне використання алгоритмів* – реалізація функцій та процедур, що можуть застосовуватися у різних частинах програми;
– *прозорість логіки* – блок-схеми та опис алгоритмів повинні забезпечувати зрозумілість та легкість відлагодження;
– *обробка виняткі*в – передбачення можливих помилок і непередбачуваних ситуацій під час виконання програми.

Таким чином, етап проєктування програмного забезпечення формує логічну основу для подальшої реалізації програмного продукту, забезпечуючи послідовність алгоритмів, чітку організацію даних та модульність коду.

3.5 Структури даних та організація зберігання інформації

На даному етапі курсового проєкту здійснюється вибір та обґрунтування структур даних, які використовуються для зберігання, обробки та передачі інформації в програмному забезпеченні. Правильний вибір структур даних безпосередньо впливає на ефективність алгоритмів, зручність реалізації програмної логіки та можливість подальшого розширення функціоналу.

3.5.1 Основні структури даних. Для організації даних у програмному проєкті застосовуються такі типи структур даних:

- *списки* – використовуються для зберігання впорядкованих наборів однотипних або різнотипних елементів з можливістю динамічного додавання та видалення;
- *кортежі* – застосовуються для зберігання фіксованих наборів даних, які не потребують змін під час виконання програми.
- *словники* – використовуються для зберігання даних у форматі «ключ – значення», що забезпечує швидкий доступ до елементів за ключем.
- *множини* – застосовуються для зберігання унікальних елементів та виконання операцій над множинами.
- *користувацькі структури* – за необхідності можуть використовуватися вкладені структури даних для групування логічно пов’язаних даних.

3.5.2 Організація зберігання інформації. Інформація у програмному проєкті може зберігатися:

- *у пам’яті програми* – під час виконання програмного коду, із використанням змінних та колекційних типів даних;
- *у файлах* – для забезпечення збереження результатів між сеансами роботи програми (текстові файли, файли формату CSV, JSON або інші структуровані формати).

Для роботи з файлами використовуються стандартні засоби мови програмування Python, що дозволяють виконувати читання, запис та оновлення інформації. Формат зберігання даних обирається з урахуванням

зручності обробки, читабельності та можливості подальшого аналізу інформації.

3.5.3 Забезпечення цілісності та коректності даних. Під час організації зберігання інформації особлива увага приділяється забезпеченню коректності даних, що досягається шляхом:

- перевірки вхідних даних перед їх збереженням;
- контролю типів даних та допустимих діапазонів значень;
- запобігання дублюванню інформації у структурах даних;
- обробки помилок під час читання та запису файлів.

3.5.4 Взаємозв'язок структур даних та алгоритмів. Обрані структури даних тісно пов'язані з алгоритмами, що використовуються у програмному проєкті. Наприклад, ефективність алгоритмів пошуку, сортування та фільтрації безпосередньо залежить від типу структури даних та способу її організації. Тому при проєктуванні враховуються вимоги до *швидкодії, масштабованості та зручності програмної реалізації*.

Таким чином, використання відповідних структур даних та продумана організація зберігання інформації забезпечують *ефективну роботу програмного забезпечення*, підвищують надійність та спрощують подальший супровід і розвиток програмного проєкту.

3.6 Опис алгоритмів обробки даних. Пошук, сортування, рекурсивні обчислення. Аналіз алгоритмічної складності

На даному етапі курсового проєкту розглядаються алгоритми обробки даних, що реалізуються у програмному забезпеченні. Вибір алгоритмів визначається функціональними вимогами системи, обсягом вхідних даних та вимогами до ефективності виконання програми.

3.6.1 Алгоритми пошуку. Алгоритми пошуку використовуються для знаходження необхідних елементів у структурах даних. У програмному проєкті можуть бути реалізовані такі алгоритми:

- *лінійний пошук* – послідовний перегляд елементів колекції до знаходження потрібного значення або завершення списку. Даний алгоритм є простим у реалізації, але має лінійну алгоритмічну складність;
- *бінарний пошук* – застосовується до впорядкованих структур даних та забезпечує швидший пошук за рахунок поділу масиву навпіл на кожному кроці.

Вибір конкретного алгоритму пошуку залежить від типу структури даних та необхідності попереднього сортування.

3.6.2 Алгоритми сортування. Сортування даних є важливою операцією, яка забезпечує зручність аналізу інформації та підвищує ефективність пошуку. У межах курсового проєкту можуть бути використані такі алгоритми:

- *бульбашкове сортування (Bubble Sort)* – простий у реалізації алгоритм, що базується на багаторазовому порівнянні та переміщенні сусідніх елементів;
- *сортування вибором (Selection Sort)* – полягає у виборі мінімального або максимального елемента з неупорядкованої частини та його переміщенні у впорядковану;
- *сортування вставками (Insertion Sort)* – поступово формує впорядковану послідовність шляхом вставки елементів у відповідну позицію;
- *швидке сортування (Quick Sort)* – ефективний рекурсивний алгоритм, що використовує принцип «розділяй і володарюй».

Вибір алгоритму сортування здійснюється з урахуванням розміру даних та вимог до швидкодії програми.

3.6.3 Рекурсивні обчислення. Рекурсія використовується для реалізації алгоритмів, які природно описуються через повторення однакових дій із зменшенням розміру задачі. У програмному проєкті рекурсивні обчислення можуть застосовуватися для:

- обробки ієрархічних або вкладених структур даних;
- реалізації алгоритмів сортування та пошуку;
- обчислення математичних функцій та послідовностей.

При використанні рекурсії особлива увага приділяється умовам завершення рекурсивного виклику, що запобігає зацикленню та переповненню стеку викликів.

3.6.4 Аналіз алгоритмічної складності. Для оцінки ефективності алгоритмів у курсовому проєкті проводиться аналіз їх алгоритмічної складності. Алгоритмічна складність визначає, скільки ресурсів (часу та/або пам'яті) потребує алгоритм для обробки вхідних даних. Основними характеристиками є:

- *часова складність (Time Complexity)* – кількість базових операцій, які виконує алгоритм залежно від розміру вхідних даних (n);
- *просторова складність (Space Complexity)* – обсяг оперативної пам'яті, необхідний для виконання алгоритму.

Нехай A – алгоритм для розв'язання певного класу задач, а n – розмір вхідних даних. Позначимо $f_a(n)$ як функцію, що визначає верхню межу числа базових операцій (додавання, множення тощо), які виконує алгоритм A при задачі розмірності n.

Говорять, що алгоритм A є *поліноміальним*, якщо $f_a(n)$ зростає не швидше, ніж деякий поліном від n. Якщо ж зростання перевищує поліноміальне, алгоритм відносять до *експоненційних*.

Між цими класами алгоритмів існує суттєва різниця: для великих n поліноміальні алгоритми можуть бути виконані на сучасних комп'ютерах, тоді як експоненційні алгоритми практично невиконувані через значний обсяг обчислень. Часто експоненційні алгоритми пов'язані з повним перебором усіх можливих варіантів. Тому для практичних задач застосовують ефективніші поліноміальні алгоритми, що дають наближене, а не обов'язково оптимальне рішення.

Навіть серед поліноміальних алгоритмів продуктивність може значно відрізнятися залежно від ступеня полінома, що описує залежність $f_a(n)$. Для оцінки часової складності алгоритмів широко використовується нотація «велике О» (O).

Числова функція $f(n)$ називається **функцією порядку** $O(g(n))$, де $g(n)$ – деяка числова функція, якщо існують такі числа $C > 0$ і n_0, що виконується умова

$$0 \leq f(n) \leq Cg(n) \tag{3.1}$$

для будь-яких $n \geq n_0$.

Цей факт позначається так

$$f(n) = O(g(n)),$$

при цьому функція $g(n)$ називається **асимптотичною верхньою оцінкою** функції $f(n)$ (рис. 3.3).

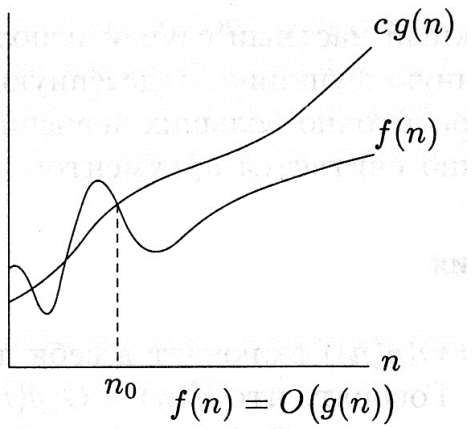


Рисунок 3.3 – Графік функції $f(n)$ порядку $O(g(n))$

Іншими словами, якщо $f(n) = O(g(n))$, то це означає, що час виконання алгоритму зростає не швидше, ніж $g(n)$, починаючи з деякого числа n_0.

Наприклад, якщо асимптотична складність алгоритму дорівнює $O(n^2)$, то час його виконання зростає пропорційно квадрату числа n.

Алгоритм A називають *поліноміальним*, якщо $f_a(n) = O(P_k(n))$, де $P_k(n)$ – поліном ступеня k:

$$P_k(n) = a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0,\ a_i \in R, i = \overline{1,k}, k \in N.$$

Наприклад, алгоритм із асимптотичною складністю $O(nlogn)$ відноситься до поліноміальних.

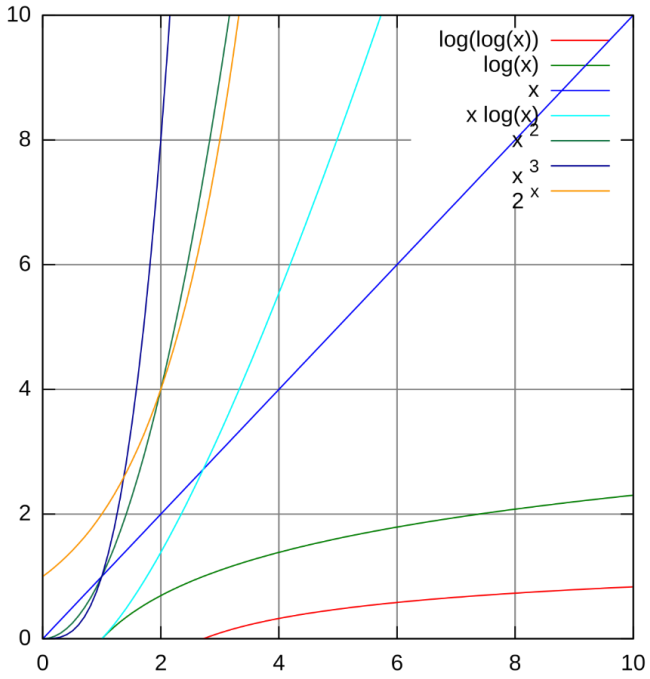


Рисунок 3.4 – Графіки функцій асимптотичної складності

У таблиці 3.2 наведено поширені асимптотичні складності з коментарями.

Таблиця 3.2 – Поширені асимптотичні складності

Складність	Коментар	Приклади
$O(1)$	Сталий час роботи, не залежить від розміру задачі	Очікуваний час пошуку в хеш-таблиці
$O(log\ log\ n)$	Дуже повільне зростання необхідного часу	Очікуваний час роботи інтерполюючого пошуку n елементів
$O(log\ n)$	Логарифмічне зростання – подвоєння розміру задачі збільшує час роботи на сталу величину	Обчислення (x^n); двійковий пошук в масиві з n елементів
$O(n)$	Лінійне зростання – подвоєння розміру задачі подвоює необхідний час	Додавання/віднімання чисел з n цифр; лінійний пошук в масиві з n елементів
$O(n\ log\ n)$	Лінеаритмічне зростання (або лінійно-логарифмічне) – подвоєння розміру задачі збільшує необхідний час трохи більше ніж вдвічі	Сортування злиттям або купою n елементів; нижня границя сортування порівнянням n елементів
$O(n^2)$	Квадратичне зростання – подвоєння розміру задачі вчетверо збільшує необхідний час	Елементарні алгоритми сортування (бульбашкою, вставками)
$O(n^3)$	Кубічне зростання – подвоєння розміру задачі збільшує час у вісім разів	Звичайне множення матриць

$O(c^n)$	Експоненційне зростання – збільшення розміру задачі на 1 призводить до с-кратного збільшення часу; подвоєння розміру задачі підносить час у квадрат	Деякі задачі комівояжера, алгоритми пошуку повним перебором

Для експоненційних алгоритмів часову складність записують так: $f_a(n) = O(k^N), k > 1$.

Розглянемо програму max_elements, яка реалізує алгоритм знаходження максимального елементу в кожному рядку масиву $A[n, n]$:

```
print("Програма знаходження максимального елемента в кожному рядку масиву")
print("============================================================")
A = [
   [3, 5, 2, 8],
   [7, 1, 9, 4],
   [6, 0, 2, 3],
   [1, 8, 5, 7]
]
print(A)
n = len(A)
comparison_count = 0

for i in range(n):
   max_value = A[i][0]
   for j in range(n):
      comparison_count += 1
      if A[i][j] > max_value:
         max_value = A[i][j]
   print(f"Максимальний елемент у рядку {i+1} =")
   print(max_value)

theoretical_complexity = n * n
base_operations = comparison_count + n * n
C = base_operations / theoretical_complexity

print("Аналіз складності алгоритму")
print("=======================")
print(f"Загальна кількість ітерацій в циклах N = {n * n}")
print(f"Загальна кількість порівнянь K = {comparison_count}")
print(f"Загальна кількість базових операцій O = {base_operations}")
print(f"Теоретична складність O(n^2): {theoretical_complexity}")
print(f"Коефіцієнт пропорційності C = O / n^2: {C:.2f}")
```

У цьому алгоритмі (програмі) змінна i змінюється від 1 до n. Для кожного значення i змінна j також змінюється від 1 до n. Отже, під час кожної з n ітерацій зовнішнього циклу внутрішній цикл виконується теж n разів.

Загальна кількість ітерацій у зовнішньому і внутрішньому циклах дорівнює: $n \times n = n^2$. Окрім того, потрібно врахувати стільки ж операцій порівняння (comparison_count) в алгоритмі. Тоді кількість базових операцій в алгоритмі дорівнює $2n^2$ і теоретично часова складність алгоритму становить $O(n^2)$ при коефіцієнті пропорційності $C = 2$ у формулі (3.1).

```
Програма знаходження максимального елемента в кожному рядку
============================================================
[[3, 5, 2, 8], [7, 1, 9, 4], [6, 0, 2, 3], [1, 8, 5, 7]]
Максимальний елемент у рядку 0=
8
Максимальний елемент у рядку 1=
9
Максимальний елемент у рядку 2=
6
Максимальний елемент у рядку 3=
8
Аналіз складності алгоритму
===========================
Загальна кількість ітерацій в циклах N = 16
Загальна кількість порівнянь K = 16
Загальна кількість базових операцій O = 32
Теоретична складність O(n^2): 16
Коефіцієнт пропорційності C = O / n^2: 2.00
```

Рисунок 3.5 – Результат виконання програми max_elements

Відповідно до отриманих результатів, часова складність алгоритму оцінюється дійсно як $O(n^2)$. Це означає, що час виконання алгоритму зростає пропорційно квадрату розміру масиву.

Порівняння алгоритмів за складністю дозволяє обґрунтувати вибір кращих рішень та продемонструвати розуміння принципів ефективної алгоритмізації.

Таким чином, реалізація та аналіз алгоритмів пошуку, сортування і рекурсивних обчислень є важливою складовою курсового проєкту, що дозволяє студентам закріпити теоретичні знання та отримати практичні навички розробки ефективних програмних рішень.

3.7 Архітектура програмного проєкту та модульна структура

Архітектура програмного проєкту визначає загальну структуру програмного забезпечення, принципи взаємодії його компонентів та порядок організації програмного коду. Правильно спроєктована архітектура сприяє

підвищенню зрозумілості, масштабованості та супроводжуваності програмного продукту.

3.7.1 Загальна архітектура програмного проєкту. Програмний проєкт будується за принципом ієрархічної та модульної структури, де кожен компонент виконує чітко визначені функції. Залежно від рівня складності курсового проєкту програмне забезпечення може бути реалізоване у вигляді:

- *консольного застосунку* з меню та підменю;
- *графічного застосунку* з використанням бібліотек графічного інтерфейсу;
- *веб-застосунку* з використанням сучасних веб-фреймворків.

Незалежно від способу реалізації, архітектура проєкту передбачає поділ програми на логічні рівні, зокрема: рівень взаємодії з користувачем, рівень обробки даних та рівень зберігання інформації.

3.7.2 Модульна структура програми. Модульна структура полягає у поділі програмного коду на окремі *модулі та функції*, кожен з яких відповідає за реалізацію конкретного набору задач. Типова структура програмного проєкту може включати:

- головний модуль, що ініціалізує програму та забезпечує взаємодію з користувачем;
- модулі для роботи зі структурами даних та файлами;
- модулі реалізації алгоритмів пошуку, сортування та обробки інформації;
- допоміжні модулі для валідації даних та обробки помилок.

Такий підхід дозволяє зменшити складність програмного коду та забезпечує можливість повторного використання окремих модулів.

3.7.3 Взаємодія модулів та область видимості. Взаємодія між модулями здійснюється через чітко визначені інтерфейси у вигляді функцій та параметрів. Особлива увага приділяється правильному використанню областей видимості змінних (scope), що дозволяє уникнути конфліктів імен та небажаних побічних ефектів.

Для забезпечення гнучкості та зрозумілості архітектури застосовуються принципи:

- мінімізації залежностей між модулями;
- використання локальних змінних та параметрів функцій;
- обмеження використання глобальних змінних.

3.7.4 Переваги модульного підходу. Застосування модульної архітектури у курсовому проєкті забезпечує:

- спрощення процесу розробки та відлагодження;
- підвищення читабельності програмного коду;
- зручність тестування окремих компонентів;
- можливість подальшого розширення функціоналу без суттєвих змін існуючої структури.

Таким чином, продумана архітектура програмного проєкту та використання модульної структури є важливою умовою створення якісного, надійного та ефективного програмного забезпечення, що відповідає вимогам курсового проєкту з дисципліни «Алгоритмізація та програмування».

3.8 Опис реалізації програмного проєкту

На даному етапі курсового проєкту здійснюється безпосередня реалізація програмного забезпечення відповідно до розробленої архітектури, модульної структури та описаних алгоритмів. Реалізація проєкту виконується із застосуванням мови програмування Python та стандартних і додаткових бібліотек, необхідних для виконання поставлених задач. Якщо програмний код займає великий обсяг (більше 2 сторінок), то він виноситься в *додатки*.

3.8.1 Середовище розробки та використані засоби. Для розробки програмного проєкту використовується сучасне середовище програмування (IDE), яке забезпечує підтримку синтаксису мови Python, засобів відлагодження та тестування, наприклад PyCharm або Visual Studio Code з додатковими плагінами.

3.8.2 Реалізація основних функціональних можливостей. Послідовність робіт при реалізації функціональної частини програмного проєкту може бути такою:

- розробка *головного меню користувача* та логіки навігації між функціональними блоками програми;
- створення *модулів введення даних* із перевіркою коректності та обробкою помилок;
- реалізація *алгоритмів обробки даних* (пошук, сортування, фільтрація, обчислення показників тощо);
- розробка *інтерфейсів взаємодії з користувачем* (консольних або графічних) для виконання основних операцій;
- формування *виводу результатів роботи програми* у вигляді повідомлень, таблиць, файлів або графічних представлень;
- реалізація *програмних модулів та функцій*, що забезпечують модульність, повторне використання коду та логічну завершеність проєкту.

3.8.3 Обробка помилок та виняткових ситуацій. У процесі реалізації програмного проєкту передбачено обробку помилок та виняткових ситуацій, що можуть виникати під час введення, обробки або збереження даних. Для цього використовуються стандартні механізми обробки виключень, що дозволяють запобігти аварійному завершенню програми та підвищують її надійність.

3.8.4 Тестування та перевірка коректності роботи. Після реалізації основних модулів проводиться перевірка коректності роботи програмного забезпечення шляхом тестування окремих функцій та всієї програми в цілому. Тестування дозволяє виявити помилки логіки, перевірити правильність обробки крайових випадків та оцінити стабільність роботи програми.

3.8.5 Інтерфейс користувача. Інтерфейс користувача реалізується у вигляді системи меню або графічних елементів, що забезпечують інтуїтивно зрозумілу взаємодію з програмою. Для консольної версії використовується

текстове меню, для графічної – відповідні елементи керування (кнопки, поля введення, списки тощо).

Таким чином, реалізація програмного проєкту забезпечує практичне втілення алгоритмічних і програмних рішень, розроблених на попередніх етапах, та демонструє здатність студента створювати повноцінні програмні продукти відповідно до вимог дисципліни «Алгоритмізація та програмування».

3.9 Вимоги до оформлення таблиць, формул, переліків, ілюстрацій, приміток і посилань

3.9.1 Таблиці. Числовий матеріал, як правило, оформляють у вигляді таблиць. Таблицю слід розміщати безпосередньо після тексту, в якому вона згадується вперше, або на наступній сторінці. На всі таблиці повинні бути посилання в тексті. Нумеруються таблиці згідно з вимогами 1.11. Слово “Таблиця” розміщують ліворуч над таблицею. Після слова «Таблиця» і її номера ставиться дефіс і після нього вказується назва таблиці.

Приклад 3.1. *Оформлення таблиці*

У таблиці 3.3[1] наведені результати порівняльного аналізу роботи точного і жадібного алгоритмів розв’язання задачі.

Таблиця 3.3 – Результати порівняльного аналізу роботи алгоритмів

Розмірність задачі (кількість шляхів)	Кількість задач	Точний алгоритм			Жадібний алгоритм		
		Потужність покриття	Avg	Dev	Потужність покриття	Avg	Dev
(10-20)	50	7	7,14	2,44	7	7,14	2,73
21-30	50	5	9,8	0,64	5	7	2,8
31-40	50	3	19,33	0,88	4	21	11,5
41-50	50	8	17,37	6,96	8	17,5	7,125
51-60	50*	15	6,8	3,14	15	6,86	3,70
61-100	50*	7	23,57	10,65	12	20,16	7,58

[1] У такому вигляді робиться посилання на таблицю.

Примітка. Символом * помічені задачі, для яких використовувався точний алгоритм з обмеженням за часом.

3.9.2 Формули. Формули й рівняння наводять безпосередньо після тексту, в якому вони згадуються, посередині рядка, з полями зверху й знизу не менше 12 пунктів. Формули створюються за допомогою старнадртного редактору формул *Office Math ML*.

Формули й рівняння в тексті роботи (за винятком формул і рівнянь, наведених у додатках) треба нумерувати порядковою нумерацією *в межах розділу*. Номер формули або рівняння складається *з номера розділу й порядкового номера,* розділених крапкою, наприклад, формула (1.3) – третя формула першого розділу. Номер формули (рівняння) указують на рівні формули (рівняння) у дужках *у крайньому правому положенні на рядку* (номер повинен бути вирівняний по правому краю рядка, а сама формула – по центру).

Пояснення символів і числових коефіцієнтів формул необхідно наводити безпосередньо під формулою, у тій же послідовності, у якій вони представлені у формулі. Перший рядок пояснення починають *із абзацу словом "де" без двокрапки.* Пояснення кожного символу необхідно *починати з нового рядка* (наприкінці рядків – «;», у самому кінці – крапка).

Приклад 3.2. *Оформлення формул*

Алгоритм A називають *поліноміальним*, якщо $f_a(n) = O(P_k(n))$, де $P_k(n)$ – поліном ступеня k:

$$P_k(n) = a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0,\ a_i \in R, i = \overline{1,k}, k \in N. \quad (3.2)$$

Переносити формули або рівняння на наступний рядок допускається тільки на знаках виконуваних операцій, повторюючи знак операції на початку наступного рядка. Якщо переносять формули або рівняння на знаку операції множення, застосовують знак "х". Якщо в тексті тільки одна формула або рівняння, *їх нумерують*. Формули, які йдуть одна за одною і не розділені текстом, *відокремлюють комою*.

Після кожної формули повинна стояти кома, якщо далі йдуть пояснення або наступна формула. Якщо ж ні, то - крапка.

Приклад 3.4.

$$T_1(n) = n^2, T_1(n) \le T_{1\max}, \quad (3.3)$$

$$T_2(n) = n\log_2 n, T_2(n) \le T_{2\max}. \quad (3.4)$$

3.9.3 Переліки. Переліки, якщо буде потреба, можуть бути наведені в пунктах або підпунктах. *Перед* переліком ставлять *двокрапку*.

Перед кожною позицією переліку слід ставити *малу літеру* українського алфавіту *з дужкою,* або, не нумеруючи, – *дефіс* (перший рівень деталізації).

Для подальшої деталізації переліку треба використовувати арабські цифри з дужкою (другий рівень деталізації).

Переліки першого рівня деталізації друкують малими літерами з абзацу, другого рівня – з відступом щодо місця розташування переліків першого рівня.

Приклад 3.5. *Оформлення однорівневого переліку*

У другому семестрі першого курсу складаються екзамени з таких дисциплін:

– алгоритмізація та програмування;

– вища математика;

– філософія;

– комп’ютерні мережі.

Приклад 3.6. *Оформлення дворівневого переліку*

Нижче наведено приклад дворівневої деталізації:

а) форма й розмір клітин;

б) живий склад клітин;

1) частини клітин;

2) неживі включення протопластів;

в) утворення тканини.

3.9.4 Ілюстрації. Ілюстрації необхідно розміщувати *безпосередньо після тексту*, у якому вони згадуються вперше, або на наступній сторінці. *На всі ілюстрації мають бути посилання* в роботі. На всі запозичені ілюстрації теж повинні бути посилання. Усі ілюстрації, які виносяться на захист, необхідно вказати в основній частині роботи, або в додатках.

Креслення, рисунки, графіки, схеми, діаграми мають відповідати вимогам стандартів ЄСКД і СПДС.

Ілюстрації нумеруються арабськими цифрами в межах розділу (номер ілюстрації складається *з номера розділу і номера за порядком всередині розділу*) і називаються «Рисунок», що разом з назвою ілюстрації (при необхідності) розміщуються під рисунком, наприклад: «Рисунок 3.1 – Структура бази даних підприємства» (перший рисунок третього розділу). *На всі рисунки повинні бути посилання в тексті роботи.*

Приклад 3.7. *Оформлення рисунків*

На рисунку 3.5 зображено приклад блок-схеми до програми, яка порівнює два числа (x та y) і виводить результат порівняння.

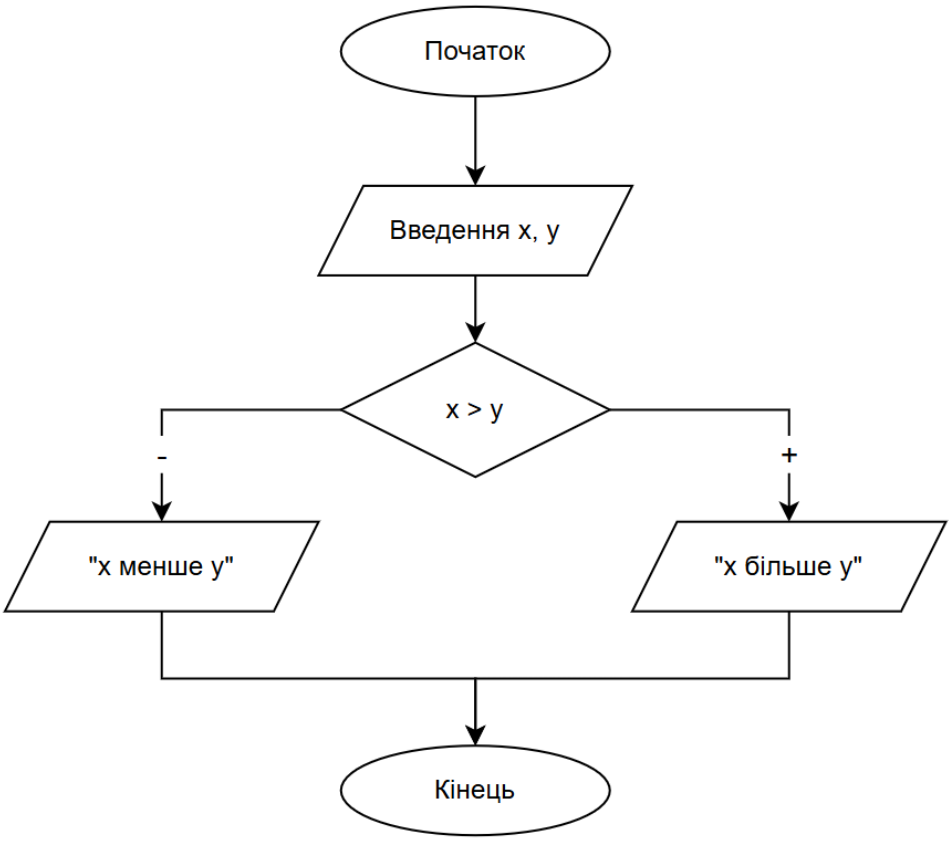


Рисунок 3.6 – Приклад блок-схеми до програми

3.9.5 Примітки. Примітки розміщують у курсовому проєкті в разі потреби пояснення змісту тексту, таблиці або ілюстрації. Примітки розташовують безпосередньо після тексту, таблиці, ілюстрації, до яких вони відносяться. Одну примітку не нумерують.

Слово «Примітка» друкують із великої літери з абзацу, не підкреслюючи, після слова «Примітка» ставлять крапку й з великої літери в тому ж рядку наводять текст примітки.

Кілька приміток нумерують послідовно арабськими цифрами із крапкою. Після слова «Примітки:» ставлять двокрапку й з нового рядка з абзацу після номера примітки з великої літери наводять текст примітки.

Приклад 3.8 *Оформлення приміток*

Примітки:

1. _____

2. _____

3.9.6 Посилання.

Посилання на використану літературу

Посилання в тексті ПЗ на джерела слід указувати порядковим номером відповідно до переліку посилань, виділеним двома квадратними дужками, наприклад, “... у роботах [1-7]...”.

Перелік використаної літератури розташовують відповідно до порядку появи посилань у тексті ПЗ, або, при великій кількості джерел, – за алфавітом.

Приклади бібліографічного опису наведені в додатку В.

Посилання на елементи пояснювальної записки

При посиланнях на розділи, підрозділи, пункти, ілюстрації, таблиці, формули, рівняння, додатка вказують їхні номера.

При посиланнях слід писати: “... у розділі 4...”, “... дивись 2.1...”, “... по 3.3.4...”, “... відповідно до 2.3.4.1...”, “... на рис. 1.3...”, “... або “... на рисунку 1.3...”, “... у таблиці 3.2...”, “... по формулі (3.1)...”, “... у рівняннях (1.23) - (1.25)...”, “... у додатку Б....”.

Якщо на розділ, підрозділ, пункт, ілюстрацію, таблицю, формулу необхідно послатися другий (третій і т.д.) раз, то це робиться так “... (див. табл. 3.2)...”, “... (див. рис. 4.6)...”.

3.10 Висновки. Наводяться висновки стосовно ефективності роботи системи та пропозиції щодо практичного впровадження розробленого програмного забезпечення.

3.11 Список використаних джерел. Список використаних джерел наводиться з нового аркуша після висновків. Бібліографічні описи наводять у порядку їх згадування в тексті та відповідно до стандартів з бібліотечної та видавничої справ (Додаток В).

3.12 Додатки. Додатки містять таблиці, форми документів, тексти програм, копії екранних форм тощо, які винесені із основної частини по причині їх великого обсягу або довідкового характеру.

Додатки слід оформляти як продовження курсового проєкту. Додатки необхідно розміщувати *в порядку появи посилань* на них у тексті.

Кожний додаток повинен починатися *з нової сторінки*. Додаток повинен мати заголовок, надрукований вгорі малими літерами з першої великої симетрично відносно тексту сторінки. Посередині рядка над заголовком великими літерами повинно бути надруковано слово “ДОДАТОК __” і велика літера, що позначає додаток.

Додатки слід позначати послідовно великими літерами української абетки, за винятком літер Ґ,Є,З,І,Ї,Й,О,Ч,Ь, наприклад, додаток А, додаток Б і т.д. Один додаток позначається як додаток А. Додатки повинні мати спільну з рештою звіту наскрізну нумерацію сторінок.

Якщо є потреба, текст додатків може розділятися на розділи, підрозділи, пункти й підпункти, які слід нумерувати в межах кожного додатка. У цьому випадку перед кожним номером ставлять позначення додатка (літеру) і крапку, наприклад: А.2 – другий розділ додатка А.

Ілюстрації, формули, рівняння й таблиці, які містяться в тексті додатка, слід нумерувати в межах кожного додатка, наприклад, таблиця А.2 – друга таблиця додатка А.

Якщо в додатку одна ілюстрація, одна формула, одне рівняння, одна таблиця, їх нумерують, наприклад, рисунок А.1. У посиланнях у тексті на

ілюстрації, таблиці, формули, рівняння рекомендується писати: «…на рисунку А.2…». Переліки й примітки в тексті додатків, якщо вони складаються з одного пункту теж необхідно нумерувати.

Якщо Додаток являє собою *документ самостійного значення* й оформляється згідно з вимогами до документа даного виду, то перед його копією вкладають аркуш, на якому *посередині* друкують “ Додаток <літера> ” і його найменування.

Сторінки копії документа нумерують, продовжуючи наскрізну нумерацію сторінок звіту, незважаючи на власну нумерацію сторінок документа (це стосується *акту про впровадження*).

4 ЗАВДАННЯ І ТЕМИ КУРСОВИХ ПРОЄКТІВ

Головним завданням курсового проєкту з дисципліни «Алгоритмізація та програмування» є розробка програмного застосунку для розв’язання задач певної предметної області, що передбачає аналіз предметної області, формалізацію задачі, розробку алгоритмів її розв’язання та програмну реалізацію із використанням мови програмування Python.

Реалізований програмний продукт повинен автоматизувати виконання типових операцій з даними предметної області, зменшити обсяг рутинної роботи користувача, забезпечити коректну обробку даних, їх збереження, пошук, сортування та виведення результатів у зручній для користувача формі.

Загальна частина завдань для кожної теми курсового проєкту:

1. Провести аналіз предметної області та сформулювати постановку задачі.
2. Розробити алгоритмічну модель розв’язання задачі, визначивши основні етапи обробки даних.
3. Вибрати та обґрунтувати структури даних, що використовуються для зберігання й обробки інформації.
4. Побудувати блок-схеми основних алгоритмів та описати логіку їх роботи.
5. Реалізувати алгоритми обробки даних (введення, перевірка коректності, пошук, сортування, обчислення, рекурсивні процедури тощо).
6. Розробити програмні модулі та функції, що забезпечують модульну структуру програми та повторне використання коду.
7. Створити користувацький інтерфейс (консольний або графічний) та головне меню для доступу до функціональних можливостей програми.
8. Забезпечити виведення результатів роботи програми у вигляді повідомлень, таблиць або файлів.
9. Провести аналіз алгоритмічної складності основних алгоритмів програми.

10. Провести модульне тестування програмних компонентів із використанням unittest або pytest.

11. Підготувати звіт з курсового проєкту, що містить опис предметної області, алгоритмів, структур даних та програмної реалізації.

12. Захистити курсовий проєкт: доповідь у формі презентації та демонстрація роботи програми на комп'ютері.

Індивідуальне завдання на виконання курсового проєкту формулюється студенту керівником курсового проєкту і оформляється у формі бланку завдання на курсовий проєкт (додаток Б).

Вибір варіанта теми курсового проєкту здійснюється з переліку тем, затверджених кафедрою (таблиця 1), за бажанням студента шляхом узгодження з керівником курсового проєкту, при цьому кожна тема повинна виконуваться одним студентом або командою у кількості 2-3 студентів, якщо створюється веб-орієнтована інформаційна система із використанняи фреймворків. Студент також може запропонувати власну тему курсового проєкту, погодивши її з керівником курсового проєкту.

Кафедра може обрати інших спосіб закріплення тем курсових проєктів за студентами, наприклад, за номером залікової книжки, або порядковим номером за журналом академічної групи тощо.

У таблиці 4.1 наведено орієнтовний перелік тем курсових проєктів.

Таблиця 4.1 – Теми курсових проєктів з курсу «Алгоритмізація та програмування» для студентів 1-го курсу спеціальності F3 – комп'ютерні науки

№ вар.	**Теми курсового проєкту**
1	Розробка програмного забезпечення для роботи магазину комп'ютерної техніки
2	Розробка програмного забезпечення для діяльності станції технічного обслуговування автомобілів
3	Розробка програмного забезпечення для планування особистих завдань
4	Розробка програмного забезпечення для роботи автосалону

5	Розробка програмного забезпечення для прокату туристичного спорядження
6	Розробка програмного забезпечення для обліку абонентів місцевої бібліотеки
7	Розробка програмного забезпечення для ведення електронного щоденника
8	Розробка програмного забезпечення для роботи кінотеатру
9	Розробка програмного забезпечення для діяльності туристичної фірми
10	Розробка програмного забезпечення для обліку спортивних тренувань
11	Розробка програмного забезпечення для формування розкладів навчального процесу
12	Розробка програмного забезпечення для діяльності готелю
13	Розробка програмного забезпечення для обліку сировини та матеріалів складу
14	Розробка програмного забезпечення для діяльності косметичного салону
15	Розробка програмного забезпечення для роботи книжкового магазину
16	Розробка програмного забезпечення для обліку проєктів ІТ-компанії
17	Розробка програмного забезпечення для прокату легкових автомобілів
18	Розробка програмного забезпечення для діяльності аптеки
19	Розробка програмного забезпечення для діяльності агентства нерухомості
20	Розробка програмного забезпечення для роботи театральних кас
21	Розробка програмного забезпечення для керування колекцією медіафайлів
22	Розробка програмного забезпечення для діяльності фотостудії
23	Розробка програмного забезпечення для роботи відділу логістики транспортної компанії
24	Розробка програмного забезпечення для обліку досягнень студента
25	Розробка програмного забезпечення для організації екскурсій
26	Розробка програмного забезпечення для обліку особистих фінансів
27	Розробка програмного забезпечення для роботи поліклінічного відділення
28	Розробка програмного забезпечення для діяльності видавництва
29	Розробка програмного забезпечення для обліку персоналу підприємства
30	Розробка програмного забезпечення для організації роботи служби таксі

5 ЗАСОБИ ДЛЯ РОЗРОБКИ ПРОГРАМНОГО ПРОЄКТУ

Для реалізації курсового проєкту студенти можуть використовувати середовища розробки, мови програмування та бібліотеки, зокрема Python.

Середовища розробки (IDE / редактори коду):

- PyCharm, Visual Studio Code, Sublime Text – основні середовища для розробки на Python;
- IDLE, Thonny – для початкового етапу навчання та консольних проєктів;
- інші середовища за погодженням із керівником.

Мова програмування та бібліотеки:

- Python – основна мова курсового проєкту;
- бібліотеки для графічного інтерфейсу (GUI): Tkinter, PyQt, Kivy, PySide;
- бібліотеки для веб-розробки та фреймворки: Flask, Django, FastAPI;
- бібліотеки для роботи з даними: pandas, NumPy;
- бібліотеки для тестування та контролю якості коду: unittest, pytest;
- інші мови та бібліотеки за погодженням із керівником.

Інструменти для моделювання та проєктування

- draw.io, Mermaid, Figma, Lucidchart, Microsoft Visio, Miro, PlantUML, Graphviz, Excalidraw – для побудови блок-схем алгоритмів, структур даних та інтерфейсів користувача;
- інші засоби за погодженням із керівником.

6 ОРГАНІЗАЦІЯ ЗАХИСТУ І ОЦІНЮВАННЯ КУРСОВОГО ПРОЄКТУ

Представлення та захист курсового проєкту виконуються студентом на кафедрі в терміни, передбачені деканатом у відповідності з графіком освітнього процесу.

Захист курсового проєкту проводиться публічно. Для проведення захисту курсового проєкту з провідних викладачів кафедри, викладачів, які читали лекції і проводили лабораторні заняття, а також керівників курсових проєктів, створюється комісія склад якої затверджує завідувач кафедри.

На захист курсового проєкту студент повинен представити комплекс вихідних матеріалів, необхідних для об’єктивного оцінювання результатів його роботи.

За результатами прилюдного захисту керівник курсового проєкту від кафедри виставляє оцінку на титульному листку пояснювальної записки, у відомість та залікову книжку з відміткою дати захисту курсового проєкту.

Підсумкова оцінка за курсовий проєкт виставляється як інтегрована оцінка за такі види навчальної діяльності:

1. Оцінка за пояснювальну записку з курсового проєкту у друкованому варіанті – **50** балів;
2. Оцінка програмної реалізації бази даних з відповідної предметної області та інформаційної системи, що її використовує – **25** балів.
3. Оцінка за презентацію звіту про виконання курсового проєкту – **10** балів;
4. Оцінка за прилюдний захист курсового проєкту – **15** балів;

Всього балів: **100**.

Підсумкова оцінка за курсовий проєкт виставляється за національною шкалою і 100-бальною шкалою за таблицею 6.1.

Таблиця 6.1 – Шкала оцінювання: національна, ЗВО та ЄКТС

Сума балів за всі види навчальної діяльності	Оцінка за національною шкалою
	для практики
90-100	відмінно
82-89	добре
74-81	
64-73	задовільно
60-63	
35-59	незадовільно з можливістю повторного захисту
1-34	незадовільно з обов’язковим повторним вивченням дисципліни

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ДСТУ ISO/IEC 2382:2017 «Інформаційні технології. Словник термінів» (ISO/IEC 2382:2015, IDT). К., ДП «УкрНДНЦ», 2017.

2. Mark Lutz. Learning Python 6th Edition, O'Reilly, 2025. 1169 p.

3. Lubanovic B. Introducing Python: Modern Computing in Simple Packages 2nd ed. O'Reilly, 2019. 627 p.

4. Пол Беррі. Head First. Python. Видавництво "Фабула", 2021. 624 с.

5. Eric Matthes. Python Crash Course, 3rd Edition. No Starch Press, 2023. 552 p.

6. Luciano Ramalho. Fluent Python. 2nd Edition. O'Reilly Media, 2022. 1014 p.

7. Allen B. Downey. Think Python: How to Think Like a Computer Scientist. 2nd Edition. O'Reilly Media, 2015. 292 p.

8. Brett Slatkin. Effective Python: 90 Specific Ways to Write Better Python. 2nd Edition. Addison-Wesley, 2019. 480 p.

9. Офіційний сайт Python. URL: https://www.python.org (дата звернення: 10.01.2026).

10. W3Schools Python Tutorial. URL: https://www.w3schools.com/python/default.asp (дата звернення: 10.01.2026).

11. Sorting Algorithm Cheat Sheet. URL: https://www.interviewcake.com/sorting-algorithm-cheat-sheet (дата звернення: 10.01.2026).

12. Python Essentials 1 by Cisco. URL: https://www.netacad.com/courses/python-essentials-1 (дата звернення: 10.01.2026).

13. Python Essentials 2 by Cisco. URL: https://www.netacad.com/courses/python-essentials-2 (дата звернення: 10.01.2026).

14. ДСТУ 8302:2015 Інформація і документація. Бібліографічне посилання. Загальні положення та правила складання. Чинний від 01.07.2016. Вид. офіц. Київ : ДП «УкрНДНЦ», 2016. 30 с.

15. ДСТУ 3008:2015 Інформація і документація. Звіти в сфері науки та техніки. Структура та правила оформлення. Чинний від 01.07.2017. Вид. офіц. Київ : ДП «УкрНДНЦ», 2016. 26 с.

16. Маттес Ерік. Пришвидшений курс Python : практичний, проєктно-орієнтований вступ до програмування. Львів : Видавництво Старого Лева, 2021. 556 с.

17. Висоцька В. А., Оборська О. В. Python : алгоритмізація та програмування : навчальний посібник / В. А. Висоцька, О. В. Оборська. 2-ге видання, стереотипне. Львів : «Новий Світ-2000», 2026. 514 с.

18. Копей В. Б. Мова програмування Python для інженерів і науковців : навчальний посібник. Івано-Франківськ : ІФНТУНГ, 2019. 272 с.

19. Яковенко А. В. Основи програмування. Python. Частина 1 : підручник; КПІ ім. Ігоря Сікорського. Київ : КПІ ім. Ігоря Сікорського, 2018. 195 с.

20. Селіверстов Р., Мельничин А. Основи програмування мовою Python : навчальний посібник. Львів : ЛНУ, 2020. 190 с.

21. Васильєв О. М. Програмування в PYTHON. Теорія і практика : навчальний посібник. Київ : Ліра-К, 2023. 462 с.

22. Чичкарьов Є.А., Зінченко О.В., Єльченко С.В. Прикладне програмування на Python. Частина 1. Основи програмування на Python. Навчальний посібник. Київ: ДУТ, 2022. 160 с.

ДОДАТОК А
Титульний аркуш пояснювальної записки

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХОЛОГІЙ І СИСТЕМ

Кафедра комп'ютерних наук та системного аналізу

КУРСОВИЙ ПРОЄКТ

з дисципліни **«Алгоритмізація та програмування»**

на тему «**Розробка програмного забезпечення для роботи магазину комп'ютерної техніки**»

Студента 1 -го курсу групи КН-25

спеціальності F3 Комп'ютерні науки

освітня програма «Комп'ютерні науки та прикладне програмування»

Ім'я ПРІЗВИЩЕ

(прізвище та ініціали студента)

Керівник

Антон МАКСИМОВ

(прізвище та ініціали викладача)

Національна шкала:_____

Кількість балів: ____

Члени комісії:

_____ Григорій ЗАСПА__

(підпис) (прізвище та ініціали)

_____ Антон МАКСИМОВ

(підпис) (прізвище та ініціали)

_____ _____

(підпис) (прізвище та ініціали)

Черкаси – 2026

ДОДАТОК Б
Бланк завдання на курсовий проєкт студенту

Черкаський державний технологічний університет

Факультет _____інформаційних технологій і систем_____
(повна назва)
Кафедра _____комп'ютерних наук та системного аналізу_____
(повна назва)
Освітньо-кваліфікаційний рівень _____бакалавр_____

Напрям підготовки_____F3 – комп'ютерні науки_____
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____
"____" _____20___року

З А В Д А Н Н Я
НА КУРСОВИЙ ПРОЄКТ СТУДЕНТУ

_____
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи)_____
_____
_____

Керівник проєкту (роботи)_____
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
Затверджена на засіданні кафедри Черкаського державного технологічного університету від « _____» _____ 20___року, протокол № _.
2. Строк подання студентом проєкту (роботи)_____

3. Вихідні дані до проєкту (роботи)_____
_____
_____
_____

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
_____
_____
_____
_____
_____
_____

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
_____
_____
_____
_____
_____

6. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання курсового проєкту (роботи)	Строк виконання етапів курсового проєкту (роботи)	Примітка
1.	Постановка завдань до курсового проєкту	До 20.02.26	
2.	Опис предметної області і постановка задачі	До 06.03.26	
3.	Проєктування програмного забезпечення	До 20.03.26	
4.	Визначення структур даних та організація зберігання інформації	До 03.04.26	
5.	Опис алгоритмів обробки даних. Аналіз алгоритмічної складності.	До 17.04.26	
6.	Розробка архітектури програмного проєкту та модульної структури	До 01.05.26	
7.	Реалізація та опис реалізації програмного забезпечення	До 12.05.26	
8.	Оформлення курсового проєкту	До 20.05.26	
9.	Подання курсового проєкту на перевірку	До 22.05.26	
10.	Захист курсового проєкту	До 29.05.26	

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи) _____ _____
(підпис) (прізвище та ініціали)

ДОДАТОК В

Приклади оформлення бібліографічного опису у списку використаних джерел з урахуванням Національного стандарту України ДСТУ 8302:2015

Характеристика джерела	Приклад оформлення
Книги: Один автор	1. Катренко А. В. Системний аналіз обєктів та процесів комп'ютеризації : навчальний посібник. Львів : Новий світ-2000, 2003. 424 с. 2. Зайченко Ю. П. Дослідження операцій : підручник. 7-ме видання, перероб. та доп. К.: Видавничий дім «Слово», 2006. 816 с. 3. Снитюк В. Є. Прогнозування. Моделі. Методи. Алгоритми : навч. посіб. К. : Маклаут, 2012. 364 с. 4. Прокопенко Т. О. Теорія систем та прийняття управлінських рішень : навчальний посібник. Черкаси : ЧДТУ, 2018. 188 с. 5. Escoffier C. Building Reactive Microservices in Java. Sebastopol : O'Reilly Media. 2017. 83 p.
Два автори	1. Волошин О. Ф., Мащенко С. О. Моделі та методи прийняття рішень : навч. посіб. для студ. вищ. навч. закл. 2-ге вид., перероб. та допов. К. : Видавничо-поліграфічний центр «Київський університет», 2010. 336 с. 2. Бідюк П. І., Коршевнюк Л. О. Проектування комп'ютерних інформаційних систем підтримки прийняття рішень : навчальний посібник. Київ : ННК «ІПСА» НТУУ «КПІ», 2010. 340 с. 3. Триус Ю. В., Галасун К. І. Нечіткі моделі і методи в системах прийняття рішень : посібник для студентів спеціальностей «Системи і методи прийняття рішень» і «Інформаційні управляючі системи і технології». Черкаси : ЧДТУ, 2014. 108 с.
Три автори	1. Дубовой В. М., Москвіна С. М., Никитенко О. Д. Моделювання процесів і систем керування. Вінниця : ВНТУ, 2009. 103 с. 2. Аніловська Г. Я., Марушко Н. С., Стоколоса Т. М. Інформаційні системи і технології у фінансах : навч. посіб. Львів : Магнолія 2006, 2015. 312 с. 3. Файнзільберг Л. С., Жуковська О. А., Якимчук В. С. Теорія прийняття рішень : підручник для студентів спеціальності «Комп'ютерні науки та інформаційні технології», спеціалізації «Інформаційні технології в біології та медицині». Київ : Освіта України, 2018. 246 с. 4. Карагодова О. О., Кігель В. Р., Рожок В. Д. Дослідження операцій: навч. посібник. Київ : Центр учбової літератури, 2007. 256 с.
Чотири і більше авторів	1. Інформаційно-аналітична система контролю та оцінювання навчальної діяльності студентів ВНЗ : монографія / Ю. В. Триус та ін. Черкаси : Видавництво МакЛаут, 2010. 300 с. 2. Операційне числення : навч. посіб. / С. М. Гребенюк та ін. Запоріжжя : ЗНУ, 2015. 88 с. 3. Клименко М. І., Панасенко Є. В., Стреляєв Ю. М., Ткаченко І. Г. Варіаційне числення та методи оптимізації : навч. посіб. Запоріжжя : ЗНУ, 2015. 84 с. 4. Томашевський О. М., Цегелик Г. Г., Вітер М. Б., Дудук В. І. Інформаційні технології та моделювання бізнес-процесів : навч. посіб. Київ : «Видавництво «Центр учбової літератури»», 2012. 296 с.

Автор(и) та редактор(и)/ упорядники	1. Інформаційні системи і технології в економіці. Посібник / за ред. д.е.н. В. С. Пономаренка. К. : Видавничий центр «Академія», 2002. 542 с. 2. В. Ф. Ситник, Т. А. Писаревська, Н. В. Єрьоміна, О. С. Краєва. Основи інформаційних систем : навч. посібник. Вид. 2-ге, пе-реробл. і доп. / за ред. В. Ф. Ситника. К. : КНЕУ, 2001. 420 с. 3. Березенко В. В. PR як сфера наукового знання : монографія / за заг. наук. ред. В. М. Манакіна. Запоріжжя ЗНУ, 2015. 362 с.
Без автора	1. Аналітичне забезпечення управлінських рішень : опорн. консп. лекцій / уклад. С. М. Скочиляс. Тернопіль : ТНЕУ, 2019. 183 с. 2. Проектування інформаційних систем : навч. посібник / за заг. ред. В. С. Пономаренка. Київ : Академія, 2002. 486 с. 3. CASE-технология моделирования процессов с использованием средств BPWin и ERWin : учебное пособие / за ред.: А. Ф. Похилько, И. В. Горбачев. Ульяновск : УлГТУ, 2008. 120 с. 4. Теорія прийняття рішень : підручник для студентів спеціальності «Комп'ютерні науки та інформаційні технології», спеціалізації «Інформаційні технології в біології та медицині» / за ред.: Л. С. Файнзільберг, О. А. Жуковська, В. С. Якимчук. Київ : Освіта України, 2018. 246 с.
Багатотомні видання	1. Енциклопедія Сучасної України / редкол.: І. М. Дзюба та ін. Київ : САМ, 2016. Т. 17. 712 с. 2. Енциклопедія кібернетики : у 2 т. / За ред. В. М. Глушкова. К. : Головна редакція Української радянської енциклопедії, 1973. Т. 1. 584 с.
Автореферати дисертацій	1. Медведєв Д. О. Метод ефективного кодування відеокадрів для підвищення продуктивності інформаційних систем : автореф. дис. ... кандт. техн. наук : 05.13.06. Черкаси. 2019. 22 с. 2. Муха А. А. Моделі, методи та технічні засоби створення гарантоздатних компютерних систем критичного призначення з двоканальною структурою обробки даних : автореф. дис. ... канд. техн. наук : 05.13.06. Київ, 2020. 26 с.
Дисертації	1. Дмітрієв О. М. Інформаційна технологія та методи підтримки прийняття рішень при ситуаційному аналізі повітряної обстановки : дис. ... докт. техн. наук : 05.13.06. / Льотна академія Національного авіаційного університету. Кропивницький. 2020. 448 с. 2. Бєгун В. В. Методологічні основи інформаційної технології управління безпекою на основі орієнтованого підходу : дис. ... докт. техн. наук : 05.13.06. / Інститут проблем математичних машин і систем Національної академії наук України. Київ. 2020. 553 с. 3. Медведєв Д. О. Метод ефективного кодування відеокадрів для підвищення продуктивності інформаційних систем : дис. ... кандт. техн. наук : 05.13.06. / Черкаський державний технологічний університет. Черкаси. 2019. 22 с.

Законодавчі та нормативні документи	1. Про освіту : Закон України від 05.09.2017 р. № 2145-VIII. *Голос України*. 2017. 27 верес. (№ 178-179). С. 10–22. 2. Про вищу освіту : Закон України від 01.07.2014 р. № 1556-VII. Дата оновлення: 28.09.2017. URL: http://zakon2.rada.gov.ua/laws/show/1556-18 (дата звернення: 15.11.2017). 3. Деякі питання стипендіального забезпечення : Постанова Кабінету Міністрів України від 28.12.2016 р. № 1050. *Офіційний вісник України*. 2017. № 4. С. 530–543. 4. Про затвердження Положення про проведення практики студентів вищих навчальних закладів України : Наказ Міністерства освіти і науки України від 08.04.1993 р. № 93. 5. Про затвердження Вимог до оформлення дисертації : наказ Міністерства освіти і науки від 12.01.2017 р. № 40. *Офіційний вісник України*. 2017. № 20. С. 136–141. 6. Про практичну підготовку студентів : Лист Міністерства освіти і науки України від 07.02.09 № 1/9-93. 7. Положення про організацію освітнього процесу в Черкаському державному технологічному університеті від 18.12.2017, № 7. URL: https://chdtu.edu.ua/normative/regulations/item/3636-polozhennya-proorhanizatsiyu-osvitnoho-protsesu-v-cherkaskomu-derzhavnomu-tekhnolohichnomuuniversyteti (дата звернення 01.09.2019).
Архівні документи	1. Організація документів на рівні архіву. Системні вимоги: AdobeAcrobatReader. URL: http:// ttp://poznayka.org/s41930t1.html (дата звернення: 20.03.2017). 2. Кисельова А. Веб-сайт Державного комітету архівів України: історія, реалії, перспективи. *Архіви України.* 2003. № 4–6. С. 129. 3. Лавренюк А. Центральний державний електронний архів України : пошук відповідей на виклики часу. *Архіви України.* 2008. Вип. 1–2. С. 67–71.
Патенти	1. Тепловізор : пат. 119337 Україна. № u201702348; заявл. 13.03.2017; опублік. 25.09.2017; Бюл. №18. 4 с. 2. Спосіб діагностування запам'ятовуючих пристроїв : пат. 114992 Україна. № u201611064; заявл. 03.11.2016; опублік. 27.03.2017; Бюл. №6. 3 с.
Стандарти	1. ДСТУ 7152:2010. Видання. Оформлення публікацій у журналах і збірниках. [Чинний від 2010-02-18]. Вид. офіц. Київ, 2010. 16 с. (Інформація та документація). 2. ДСТУ 3008:2015. Звіти у сфері науки і техніки. Структура та правила оформлювання / Нац. Стандарт України. Вид. офіц. [На заміну ДСТУ 3008-95; чинний від 2017-07-01]. Київ : ДП «УкрНДНЦ», 2016. 31 с. (Інформація та документація). 3. ДСТУ 8302:2015. Бібліографічне посилання. Загальні положення та правила складання / Нац. Стандарт України. Вид. офіц. [Уведено вперше; чинний від 2016-07-01]. Київ : ДП «УкрНДНЦ», 2016. 17 с. (Інформація та документація). – З внесеними правками. 4. ДСТУ ISO/IEC 24767-1:2016. Інформаційні технології. Безпека внутрішньої мережі. Частина 1. Вимоги щодо безпеки (ISO/IEC 24767-1:2008, IDT) / Нац. Стандарт України. Вид. офіц. [Введено вперше; чинний від 2016-10- 10]. Київ : ДП «УкрНДНЦ», 2016. 5. ДСТУ EN 60950-1:2015. Обладнання інформаційних технологій. Безпека. Частина 1. Загальні вимоги (EN 60950-1:2006; A11:2009; A1:2010; A12:2011; AC:2011; A2:2013, IDT) / Нац. Стандарт України. Вид. офіц.

	[На заміну ДСТУ EN 60950-1:2014; чинний від 2017-01-01]. Київ : ДП «УкрНДНЦ», 2017. 359 с. 6. Стандарт вищої освіти України першого (бакалаврського) рівня ступеня «бакалавр» за галуззю знань 12 «Інформаційні технології» спеціальністю 122 «Комп'ютерні науки та інформаційні технології» // Видання Міністерства освіти і науки України. 2016. 25 с. URL: http://mon.gov.ua/activity/education/reforma-osviti/naukovo-metodichna-rada-ministerstva/proekti-standartiv-vishhoyi-osviti.html (дата звернення 05. 04. 2017).
Каталоги	1. Каталог наукових розробок Черкаського державного технологічного університету / Черкаський держ. технол. ун-т. Черкаси : Видавництво ЧДТУ, 2009. 102 с. 2. Каталог наукових розробок Черкаського державного технологічного університету [Текст] / М-во освіти і науки України, Черкаський держ. технол. ун-т ; уклад.: А. І. Садовий, І. В. Мельник, Л. М. Арестова. 2-е вид., допов. Черкаси : Видавництво ЧДТУ, 2009. 102 с.
Бібліографічні покажчики	1. Володимир Віталійович Грабко [Текст] : біобібліогр. покажч. до 55-річчя від дня народж. / ВНТУ, НТБ ВНТУ ; уклад.: Л. Д. Андронік, Л. М. Желюк, відп. за вип. Т. Є. Притуляк. Вінниця : ВНТУ, 2015. 72 с. (Вчені нашого університету). 2. Юрій Ярославович Бобало : біобібліогр. покажч. : до 70-річчя від дня народж. / Нац. ун-т «Львів. Політехніка», наук.-техн. б-ка ; уклад. О.Б. Ніколюк ; ред. рада О.В. Шишка (голова). Львів : Видавництво Львівської політехніки, 2015. 80 с. (Біобібліографія вчених Львівської політехніки ; вип. 56). 3. Архипова Світлана Петрівна : біобібліогр. покаж. / Черкас. нац. ун-т ім. Б. Хмельницького ; [наук. ред. В. В. Масненко; у поряд. О. З. Медалієва]. Черкаси : [Вид. від. ЧНУ], 2007. 35 с. (Бібліографія вчених Черкась кого національного у ніверситету ім. Б. Хмельницького ; вип. 8). 4. Рекомендаційний - бібліографічний покажчик «Праці викладачів ЧДТУ, які знаходяться в фондах бібліотеки ЧДТУ за 2013-2014 рік». Черкаси : ЧДТУ, 2015. URL : http://lib.chdtu.edu.ua/resursi/bibliografichni
Частина видання: книги	1. Кравченко П. Блокчейн і децентралізовані системи : навч. посібник у 3 ч. Ч. 1 / за ред. П. Кравченко, Б. Скрябін, О. Дубініна. Харків : ПРОМАРТ, 2019. 452 с. 2. Кушнаренко Н.М. Індексування документів / за ред. Н. М. Кушнаренко, В. К. Удалова. *Наукова обробка документів.* 2003. Розд. 3. С. 105–210.
Частина видання: матеріалів конференцій (тези, доповіді)	1. Ступницький О. І., Дашкуєв М. А. Інформаційні технології у інфраструктурі глобальних логістичних мереж. *Актуальні проблеми міжнародних відносин :* Зб. наук. праць. Випуск 122, Частина ІІ. К. : Київ. нац. ун-т ім. Т. Шевченка, Ін-т міжнар. відносин, 2014. С. 104–115. 2. Аль-Амморі Алі, Дяченко П. В. Аналіз особливостей розвитку інформаційних процесів і технологій. *Інформаційні технології в освіті, науці і техніці :* тези доповідей V Міжнародної науково-практичної конференції, м. Черкаси, 21-23 травня 2020 р. Черкаси, 2020. С. 18-20. 3. А. Максимов, О. Новосад. Інформаційна технологія системи аналітичної обробки множинних даних. *Проблеми зняття з експлуатації об'єктів ядерної енергетики та відновлення навколишнього середовища (INUDECO 2020) :* тези доп. IV Міжнародної конференції (м. Славутич, 27–29 квітня 2020). Чернігів : ЧНТУ, 2020. 142-143 с.

Частина видання: довідкового видання	1. *Короткий англо-український тлумачний словник з комп'ютерної техніки* / уклад. Р. Р. Сіренко, М. О. Сапронов, Ю. М. Пугач, Л. В. Левків. Львів : Вид. центр ЛНУ ім. І. Франка, 2005. 98 с. 2. Дербенцев В. Д., Семьонов Д. Є. та ін. *Словник термінів інформаційних систем і технологій* / ред.-лексикограф Л. О. Симоненко. К. : КНЕУ. 2008. 256 с. 3. *Термінологічний тлумачний словник з інформатики та інформаційних технологій з ілюстраціями* / укл. Кушерець В. І., Дибкова Л. М. Київ-Донецьк: Університет сучасних знань, 2010. 304 с. 4. Палагін О. В., Петренко М. Г. *Тлумачний онтографічний словник з інженерії знань.* Київ : ТОВ «НВП Інтерсервіс», 2017. 478 с. 5. *Тлумачний словник з інформатики* / ред. Г. Г. Півняк, Б. С. Бусигін, М. М. Дівізінюк та ін. Д., Нац. гірнич. ун-т, 2010. 600 с.
Частина видання: продовжуваного видання	1. І. В. Миронець, В. М. Пономаренко. Автоматизована система захисту програмного забезпечення для операційної системи Android. *Вісник Черкаського державного технологічного університету. Технічні науки.* Черкаси, 2020. № 1. С. 43–49. 2. Згуровський М. З., Сергієнко І. В. Інформаційні технології у сучасному суспільстві. *Вісник НАН України.* 2000. №12. С. 9–16. 3. Тимченко А. А., Крижановський Є. М., Мельник В. П., Підгорний М. В. Інформаційна технологія підтримки прийняття рішень при ліквідації надзвичайних ситуацій. *Вісник ЧДТУ. Інформаційні технології, обчислювальна техніка і автоматика.* Черкаси. 2014. № 3. С.5-11.
Частина видання: періодичного видання (журналу, газети)	1. В. М. Теслюк, О. І. Поцілуйко, Т. В. Теслюк. Моделі та засоби системи зосередження уваги водіїв транспортних засобів для мобільних пристроїв. *Український журнал інформаційних технологій.* 2019. № 1. С. 24–34. 2. Д. О. Кушнір, Я. С. Парамуд. Методи пошуку та розпізнавання об'єктів у відеозображеннях на мобільній платформі IOS в реальному часі. *Комп'ютерні системи та мережі.* 2019. Випуск 1, № 1. С. 24–34. 3. У. Ю. Дзелендзяк, В. В. Самотий, В. О. Палюшок. Розроблення інтерактивного веб-додатка з використанням нереляційної бази даних. *Автоматика, вимірювання та керування.* 2019. № 1. С. 25–31.
Електронні ресурси	1. План тестування хмарного сховища TuchaBackup. URL: https://tucha.ua/uk/blog/support/plan-testuvannya-khmarnogo-skhovischatuchabackup (дата звернення: 20.09.2019). 2. Главчева Ю. Н. Регистрация в Google Scholar Citations. URL: http://repository.kpi.kharkov.ua/bitstream/KhPIPress/21556/1/Glavcheva_Registratsiya_v_Google_2016.pdf (дата звернення: 27.01.2018). 3. Діденко Ю. В., Радченко А. І., Коваль Н. В. Інформаційна система Web of Science : дзеркало чи інструмент? *Наука та іннов.* 2016. № 6. С. 45–54. URL: http://dx.doi.org/10.15407/scin12.03.014 (дата звернення: 15.04.2017). 4. Ю. О. Бабій, В. П. Нездоровін, Є. Г. Махрова, Л. П. Луцкова. Хмарні обчислення проти розподілених обчислень: сучасні перспективи. *Вісник Хмельницького національного університету. Технічні науки*. Хмельницьк, 2011. № 6. С. 80–85. URL: http://elar.khnu.km.ua/jspui/bitstream/123456789/381/1/6_4.pdf. (дата звернення: 15.11.2017).