**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ПРИВАТНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД**

**«ЄВРОПЕЙСЬКИЙ УНІВЕРСИТЕТ»**

**Факультет інформаційних систем і технологій**

**Кафедра інформаційних систем та математичних**

**дисциплін**

«ЗАТВЕРДЖУЮ»

Проректор

з навчально-методичної роботи

\_\_\_\_\_\_\_\_\_\_\_\_С.М. Ягодзінський

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2025 р.

**МЕТОДИЧНІ РЕКОМЕНДАЦІЇ**

до виконання курсової роботи з дисципліни

**«СИСТЕМНЕ ПРОГРАМУВАННЯ»**

галузь знань: 12 «Інформаційні технології»

Спеціальності: 121«Інженерія програмного забезпечення»,

**Київ – 2025 рік**

Методичні рекомендації до виконання курсової роботи з дисципліни «Системне програмування» для студентів за спеціальністю 121 «Інженерія програмного забезпечення», денної та заочної форми навчання. – ПВНЗ «Європейський університет», 2025. – 25 с.

Розробник:

Викладач кафедри комп’ютерних наук та програмної інженерії

Рихальський О.Ю.

Методичні рекомендації затверджено на засіданні кафедри

Протокол № \_\_ від «\_\_\_» \_\_\_\_\_\_\_\_\_ 202\_ року.

Завідувач кафедри комп’ютерних наук

та програмної інженерії \_\_\_\_\_\_\_\_\_\_\_\_\_\_ д.т. н, проф. Єнікєєв О.Ф.

**ЗАГАЛЬНІ ПОЛОЖЕННЯ**

1. Методичні рекомендації призначені для студентів спеціальностей 121 «Інженерія програмного забезпечення», а також для керівників курсовими роботами.

Мета рекомендацій − розробити загальні положення до виконання та захисту курсової роботи.

2. Виконання курсової роботи та захист її студентом на кафедрі комп’ютерних наук та програмної інженерії є заключним етапом для вивчення дисципліни: “Системне програмування”.

3. Мета курсової роботи:

* засвоїти та поглибити теоретичні знання з питань: методів аналізу та проектування програмних систем, вибору та використання методів проектування, засобів програмування, створення та тестування програмних систем;
* набути навичок проектування, програмування та тестування сучасних програмних систем або програмно-технічних комплексів;
* розвинути ініціативу та творчу активність до самостійного прийняття інженерних рішень в галузі розробки програмних систем;
* навчитися проводити аналіз зовнішнього інформаційного середовища та існуючих рішень з використанням інтернет, ChatGPT, бібліотек, користуватися навчально-технічною та довідковою літературою, складати звіти про виконану роботу, а також робити усні повідомлення (захист роботи) по вивченій темі.

4. Темами курсової роботи є задачі, які так чи інакше пов'язані з системним програмуванням, роботою з портами, потоками, мережею та ін., методами проектування та розробки програмних систем, а також програмно-технічних комплексів у різних галузях діяльності інженерів – програмістів.

5. Структура курсових робіт встановлюється методичними вказівками на підставі діючих нормативних документів.

6. Бажано при виборі теми курсової роботи, розглядати її як частину майбутньої дипломної роботи.

1. **ВИБІР ТА ЗАКРІПЛЕННЯ ТЕМИ**

1.1. Студент виконує курсову роботи протягом 2-2,5 місяців згідно методичних рекомендацій.

1.2. Перелік рекомендованих тем курсових робіт до дисципліни розробляється на базі навчальної програми, викладачами, які її ведуть (керівниками курсовими роботами). Перелік тем розглядається та затверджується на засіданні кафедри (додаток 1).

1.3. Темою курсової роботи може бути також проблема, яка розробляється студентом по замовленню підприємства, фірми та інших установ, у випадку погодження її з керівником курсовими роботами.

Студент має право зробити уточнення та змінити зміст у вибраній темі, у разі обґрунтування цих позицій, а також запропонувати самостійно сформульовану тему, яка відповідає його здібностям та науковим інтересам.

1.4. Закріплення теми курсової роботи проводиться керівником курсовими роботами по особистій заяві студента.

1.5. Керівник курсовими роботами виконує наступні функції:

- допомагає студентові у виборі або формулюванні теми курсової роботи, розробці її концепції та визначає графік її виконання;

- надає поради з питань вибору літературних та інших джерел, збору, обробці та використанню матеріалів;

* контролює хід виконання роботи;
* перевіряє підготовлений текст звіту;
* приймає участь в підготовці та захисті курсової роботи на кафедрі;
* виставляє разом з іншими представниками комісії оцінку за курсову роботу.

1.6. Студент має право вибору керівника курсової роботи (у разі обґрунтування цих позицій) із числа викладачів, які ведуть керівництво курсовими роботами з відповідної дисципліні.

1.7. Консультування студентів проводиться у вільний час від занять за встановленим графіком.

1.8. У встановлений графіком час закінчення курсової роботи студент повідомляє керівнику про ступень її завершення. Повідомлення студента може бути заслухано на засіданні кафедри у випадку значних відставань або, навпаки, з приводу підготовки найкращої роботи.

1.9. Курсова робота перевіряється на плагіат. До захисту допускається робота з показником плагіату менше 30.

**2. ОБСЯГ, СТРУКТУРА ТА ЗМІСТ КУРСОВОЇ РОБОТИ**

2.1. Курсова робота визначається структурою та її змістом, але її об’єм повинен бути 30-40 сторінок друкованого тексту, на аркушах формату А4 (210x297 мм), із використанням міжрядкового інтервалу 1.5, шрифт Times New Roman 14.

* 1. Робота повинна мати наступну структуру:
* титульний лист (додаток 2);
* зміст;
* вступ;
* постановка задачі і мета роботи;
* опис предметної області, аналіз і проектування системи;
* реалізація системи мовою програмування;
* опис програмного забезпечення;
* контрольний приклад;
* висновки;
* література;
* додаток із лістінгом програми та результатами.

У **вступі,** на підставі аналізу інформаційного середовища обґрунтовується актуальність та практична цінність програмної системи, що планується до реалізації, проводиться аналіз існуючих систем даного типу, а також проблем, що виникають при створенні програмних систем даного типу. Обсяг розділу 1 сторінка.

В розділі **“Постановка задачі і мета роботи”** визначається проблема, на вирішення якої спрямовано створення прикладної програмної управляючої системи.

Обсяг розділу 4-5 сторінок.

Для студентів третього курсу потрібно розробити прикладний програмний продукт, який базується на низькорівневому програмуванні, функціональність якого відповідає індивідуальному варіанту завдання. Додати в програмний продукт перевірки на всі можливі помилки та інформувати користувача при їх виникненні. Написати імплементацію для таких операційних систем, як Windows, MacOS, Linux.

Код має бути розташований на GitHub.

Імплементація має бути написана на мові програмування Rust.

Інсталяція програми має відбуватися шляхом клонування репозиторію та його компіляції локально на комп’ютері користувача.

В розділі **“Опис предметної області, аналіз і проектування системи”** дається характеристика системи, яка є предметом курсової роботи, проводиться аналіз системи, проводиться аналіз словника та варіантів використання системи, інформаційних потоків, процесів керування, координації в системі, тощо. На підставі моделі системи, що містить цілі функціонування підсистем, встановлюються характеристики інформаційних потоків, організаційна форма управління та інше.

Даний розділ може містити діаграми взаємодії між об'єктами (послідовності і кооперацій) мовою UML (Unified Modeling Language, Уніфікована мова об'єктно-орієнтованого моделювання), які відповідають потокам подій варіантів використання, діаграми класів аналізу. Якщо існує потреба, можна також включити діаграми діяльності. Діаграми необхідно супроводжувати поясненнями, що вказують, якому потоку подій вони відповідають (якщо це не ясно з їх назви), і коментарями стосовно об'єктів, присутніх на діаграмах.

Крім того, дається опис використовуваної інформації: масиви вхідних даних і їх структури, технологія отримання вхідних даних, масиви проміжних даних та їх структури. Крім того, задаються необхідні результати вирішення задачі програмною системою у вигляді масивів вихідних даних та їх відповідних структур.

Обсяг розділу 5-10 сторінок.

В розділі **“Реалізація системи мовою програмування”** на підставі проведеного аналізу та проектування (моделей аналізу та проектування) розробляється реалізація програмної системи однією з вибраних студентом мов програмування.

В разі необхідності (якщо окремі методи класів є достатньо складними) наводять схеми алгоритмів для окремих методів. Схема алгоритму - це графічне зображення алгоритму, в якому кожна дія позначається деякою геометричною фігурою – блочним символом (блоком).

Обсяг розділу 10-15 сторінок.

В розділі **“Опис програмного забезпечення”** наводиться структура програмного забезпечення, опис інтерфейсу користувача системи, призначення окремих програм, обмеження на комп’ютерне середовище, в якому виконуються програми та інше.

Також мають бути наведені: область застосування, рівень підготовки користувача системи, порядок інсталяції програм, зміст дистрибутива, порядок завантаження, опис виконуваних функцій, опис основних аварійних ситуацій, рекомендації по освоєнню програми.

Обсяг розділу 10-15 сторінок.

В розділі **“Контрольний приклад”** наводяться контрольні дані, на яких проводилось тестування програмної системи. Результати краще за все наводити у вигляді таблиць, графіків, діаграм. Об'єм розділу 5-10 сторінок.

У **висновках** даються короткі виводи по результатах роботи, пропозиції по використанню. Об'єм розділу 1 сторінка.

Розділ “**Література**” повинен мати перелік не менше 15 джерел, які використовуються при виконанні роботи, які розташовуються в алфавітному порядку прізвищ авторів. В бібліографічному описанні вказуються прізвище, ініціали автора, назва джерела, місто, місце видавництва, рік видання та кількість сторінок. При описанні статей указують номери сторінок у збірнику чи журналі.

**3. ОФОРМЛЕННЯ КУРСОВОЇ РОБОТИ**

Курсова робота повинна бути відповідно оформлена. Вона відкривається титульним аркушем, в якому вказуються назви навчального закладу, кафедри, теми, автора, керівника, його вченого ступеню та звання, міста, де розташований ЗВО та календарного року (див. додаток 2).

Далі розміщується зміст, який відображає її структуру (розділи, параграфи) із позначенням сторінок розміщення.

Розділи роботи розміщуються в послідовності, зазначеній у структурі пояснювальної записки.

Слід дотримуватись необхідної пропорції усіх частин роботи. Зокрема, передмова та висновки не можуть займати більш 15% загального об’єму.

Відносно рівномірним має бути розподілення матеріалу за розділами.

Текст подається у друкованому виді з міжрядковим інтервалом «півтора». На аркуші мають бути поля: ліве – 25мм, праве – 12мм, верхнє та нижнє – 20мм. Відступ абзацу – 1.25см, рівняння – по ширині, шрифт Times New Roman, розмір 14. Заголовки відокремлюються зверху та знизу одним інтервалом.

Усі сторінки нумеруються. Загальна нумерація сторінок роботи відкривається з титульного аркуша, але порядковий номер на ньому не ставиться; на усіх наступних він розміщується на середині верхнього поля. Розділи та параграфи мають заголовки , які позначаються прописними буквами.

Необхідно правильно оформити список літератури.

Обов’язкові точні (загально прийняті) сторінкові позначення при цитуванні та вживанні положень, результатів досліджень інших авторів.

В позначеннях потрібно максимально повно вказувати надруковані джерела:

* книги, брошури, статті, буклети, посилання в інтернет, матеріали конференцій, та інші документи, які освітлюють досвід за даною проблемою.

Після повного завершення курсова робота передається керівнику для перевірки та захисту.

**4. СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ**

1. Геннадій Галісєєв. Системне програмування.

Книга охоплює фундаментальні аспекти системного програмування, включаючи роботу з операційними системами та апаратним забезпеченням.

<https://www.yakaboo.ua/ua/sistemne-programuvannja.html>

1. Steve Klabnik, Carol Nichols. The Rust Programming Language.

Класичний посібник із Rust, офіційно рекомендований для вивчення мови. Містить розділи з низькорівневого програмування та роботи з пам’яттю.

<https://www.amazon.com/Rust-Programming-Language-Steve-Klabnik/dp/1593278284>

1. Jon Gjengset. Rust for Rustaceans: Idiomatic Programming for Experienced Developers.

Досліджує просунуті теми, такі як робота з потоками, низькорівнева оптимізація та інші аспекти, корисні для системного програмування.

<https://www.amazon.com/Rust-Rustaceans-Programming-Experienced-Developers/dp/1718501854>

1. Philipp Oppermann. Writing an OS in Rust.

<https://os.phil-opp.com>

1. Official Rust book

<https://doc.rust-lang.org/book/>

1. Rust by Examples

<https://doc.rust-lang.ru/stable/rust-by-example/meta/doc.html>

1. Async Rust by Maxwell Flitton, Caroline Morton

<https://www.oreilly.com/library/view/async-rust/9781098149086/>

1. Rust Embedded Book. Ресурс для роботи з вбудованими системами, який корисний для створення драйверів та роботи з апаратним забезпеченням.

<https://docs.rust-embedded.org/book/>

1. Rust Nomicon

<https://doc.rust-lang.org/stable/nomicon/>

1. Intel Corporation. Intel® 64 and IA-32 Architectures Software Developer’s Manual Combined Volumes: 1, 2A, 2B, 2C, 2D, 3A, 3B, 3C, 3D, and 4. Submitted: May 01, 2018 Last updated: May 27,2020.

<https://software.intel.com/content/www/us/en/develop/download/intel-64-and-ia-32-architectures-sdm-combined-volumes-1-2a-2b-2c-2d-3a-3b-3c-3d-and-4.html>.

1. Репозиторій github.com

<https://github.com/djnzx/rust-minigrep>

1. Репозиторій github.com

<https://github.com/kriomant/ch57x-keyboard-tool>

**Додаток 1**

**Орієнтовна тематика курсових робіт з дисципліни**

**«СИСТЕМНЕ ПРОГРАМУВАННЯ»**

**Networking and Communication / Мережі та комунікація**

**Варіант 1.**

**HTTP Server Implementation  
Реалізація HTTP-сервера**  
**Короткий опис:** Створення простого HTTP-сервера, який підтримує обробку запитів типу GET та POST. Сервер повинен відповідати специфікаціям протоколу HTTP/1.1 та забезпечувати базову взаємодію між клієнтом і сервером.  
**Мінімальна функціональність:**

* + Обробка HTTP-запитів типу GET та POST.
  + Відповідь із використанням відповідних HTTP-статусів.
  + Логування отриманих запитів та відповідей.

**Варіант 2.**

**Chat Application with TCP Sockets  
Чат-додаток з використанням TCP-сокетів**  
**Короткий опис:** Реалізація багатокористувацького чат-додатка на основі TCP-сокетів. Користувачі зможуть обмінюватися повідомленнями через консоль, підключаючись до одного сервера.  
**Мінімальна функціональність:**

* + Організація підключення декількох клієнтів до одного сервера.
  + Надсилання та отримання текстових повідомлень між клієнтами.
  + Логування подій сервера (підключення/відключення клієнтів).

**Варіант 3.**

**UDP-Based File Transfer  
Передача файлів за допомогою UDP**  
**Короткий опис:** Реалізація протоколу передачі файлів через UDP. Програма повинна забезпечувати обмін файлами між клієнтом і сервером, враховуючи втрати пакетів і необхідність повторної передачі.  
**Мінімальна функціональність:**

* + Надсилання та отримання файлів через UDP.
  + Виявлення та повторна передача втрачених пакетів.
  + Обробка повідомлень про завершення передачі.

**Варіант 4.**

**Ping Utility  
Утиліта для вимірювання затримки (Ping)**  
**Короткий опис:** Розробка утиліти, яка вимірює затримку (latency) до віддаленого сервера через ICMP-запити. Утиліта повинна надавати користувачеві статистику результатів.  
**Мінімальна функціональність:**

* + Надсилання ICMP-запитів до сервера.
  + Вимірювання часу відповіді для кожного запиту.
  + Підрахунок середньої та максимальної затримки.

**Варіант 5.**

**DNS Query Resolver  
Розв'язувач DNS-запитів**  
**Короткий опис:** Написання утиліти для розв’язання DNS-запитів. Програма повинна отримувати інформацію про домени, звертаючись до DNS-серверів.  
**Мінімальна функціональність:**

* + Відправлення DNS-запитів до сервера.
  + Відображення IP-адреси для введеного домену.
  + Підтримка як A-записів, так і AAAA-записів.

**Варіант 6.**

**NTP Time Synchronization   
Програма для синхронізації часу через NTP**  
**Короткий опис:** Реалізація клієнта для отримання точного часу через протокол NTP. Програма повинна відображати поточний час, отриманий від NTP-сервера.  
**Мінімальна функціональність:**

* + Відправлення NTP-запитів до сервера.
  + Отримання та відображення поточного часу.
  + Логування отриманих даних.

**Варіант 7.**

**Peer-to-Peer Communication System  
Система Peer-to-Peer комунікації**  
**Короткий опис:** Створення децентралізованої системи, у якій вузли можуть спілкуватися один з одним, використовуючи власний протокол. Система повинна підтримувати прямий обмін повідомленнями між вузлами.  
**Мінімальна функціональність:**

* + Реалізація базового протоколу для обміну повідомленнями.
  + Можливість підключення нових вузлів до системи.
  + Логування взаємодії між вузлами.

### Апаратне забезпечення та низькорівневе програмування

**Варіант 8.**

**I²C Sensor Reader  
Зчитувач сенсора температури через I²C (відеокарта)**  
**Короткий опис:** Розробка програми для вимірювання температури відеокарти за допомогою сенсора температури, підключеного через протокол I²C. Програма повинна забезпечити точне зчитування та відображення температури.  
**Мінімальна функціональність:**

* + Виявлення I²C-сенсора, підключеного до системи.
  + Читання даних про температуру з сенсора.
  + Відображення температури відеокарти в реальному часі.

**Варіант 9.**

**I²C Sensor Reader  
Зчитувач сенсора температури через I²C (процесор)**  
**Короткий опис:** Розробка програми для вимірювання температури процесора за допомогою сенсора температури, підключеного через I²C. Програма повинна надати користувачеві дані про поточний стан системи.  
**Мінімальна функціональність:**

* + Виявлення та ініціалізація I²C-сенсора.
  + Зчитування та обробка даних про температуру.
  + Відображення поточної температури процесора.

**Варіант 10.**

**GPIO Control Application  
Програма для керування GPIO**  
**Короткий опис:** Написання програми для роботи з GPIO-пінами на Raspberry Pi або подібному пристрої. Програма повинна дозволяти керувати станом пінів (включення/вимикання).  
**Мінімальна функціональність:**

* + Ініціалізація та доступ до GPIO-пінів.
  + Вмикання та вимикання пінів.
  + Логування змін стану GPIO.

**Варіант 11.**

**Keyboard interceptor  
Перехоплювач натискань клавіш**  
**Короткий опис:** Реалізація програми, яка буде працювати в пам'яті та перехоплювати всі натиснення клавіш. Це дозволить отримати дані про активність клавіатури.  
**Мінімальна функціональність:**

* + Перехоплення всіх натискань клавіш.
  + Запис перехоплених подій у лог-файл.
  + Можливість завершення роботи програми за допомогою спеціальної комбінації клавіш.

**Варіант 12.**

**Keyboard LED blinker  
Миготіння індикаторів клавіатури**  
**Короткий опис:** Розробка програми для керування індикаторами Caps Lock, Num Lock та Scroll Lock на клавіатурі. Програма повинна дозволяти миготіння індикаторів у заданій послідовності.  
**Мінімальна функціональність:**

* + Увімкнення та вимкнення індикаторів.
  + Налаштування частоти миготіння.
  + Логування активності програми.

**Варіант 13.**

**Custom Keyboard programmer  
Програмування зовнішньої клавіатури**  
**Короткий опис:** Реалізація утиліти для програмування зовнішньої клавіатури, яка підтримує зміну функціоналу клавіш (наприклад, макроси). Це дозволить налаштовувати пристрій під потреби користувача.  
**Мінімальна функціональність:**

* + Виявлення та ініціалізація зовнішньої клавіатури.
  + Зміна функціоналу клавіш (наприклад, призначення макросів).
  + Збереження та завантаження налаштувань клавіатури.

### Системні утиліти

**Варіант 14.**

**Hardware Resource Monitor  
Монітор апаратних ресурсів**  
**Короткий опис:** Розробка програми для відображення в реальному часі використання процесора та оперативної пам'яті. Інтерфейс програми повинен бути зручним і надавати користувачеві актуальну інформацію про ресурси системи.  
**Мінімальна функціональність:**

* Відображення завантаження процесора у відсотках.
* Відображення використання оперативної пам'яті.
* Оновлення даних у реальному часі.

**Варіант 15.**

**Custom Shell Implementation  
Користувацький Shell**  
**Короткий опис:** Реалізація базового shell-інтерпретатора, який підтримує виконання команд та передавання даних через канали (piping).  
**Мінімальна функціональність:**

* Виконання базових команд системи.
* Підтримка перенаправлення виводу через канали.
* Логування виконаних команд.

**Варіант 16.**

**Time Synchronization Client  
Клієнт для синхронізації часу**  
**Короткий опис:** Розробка клієнта для синхронізації системного часу за допомогою протоколу NTP. Програма повинна забезпечити коректність локального часу системи.  
**Мінімальна функціональність:**

* Надсилання запитів до NTP-сервера.
* Відображення отриманого точного часу.
* Синхронізація системного часу з NTP.

**Варіант 17.**

**File Indexer and Search Tool  
Індексація та пошук файлів**  
**Короткий опис:** Створення інструменту для індексації файлів у заданій директорії з можливістю швидкого пошуку за іменем файлу або вмістом.  
**Мінімальна функціональність:**

* Індексація всіх файлів у директорії.
* Пошук файлів за ключовими словами.
* Відображення шляху до знайденого файлу.

### Робота з файлами та системними процесами

**Варіант 18.**

**Grep utility  
Утиліта Grep**  
**Короткий опис:** Реалізація утиліти для пошуку текстових збігів у файлах. Утиліта повинна підтримувати пошук за регулярними виразами.  
**Мінімальна функціональність:**

* + Пошук тексту у вказаних файлах.
  + Підтримка регулярних виразів.
  + Відображення результатів із вказівкою на номер рядка.

**Варіант 19.**

**Word count utility  
Утиліта підрахунку слів**  
**Короткий опис:** Розробка програми для підрахунку кількості слів у текстовому файлі.  
**Мінімальна функціональність:**

* + Читання вказаного файлу.
  + Підрахунок загальної кількості слів.
  + Вивід результату на екран.

**Варіант 20.**

**Line count utility  
Утиліта підрахунку рядків**  
**Короткий опис:** Реалізація програми для підрахунку кількості рядків у текстовому файлі.  
**Мінімальна функціональність:**

* + Читання вказаного файлу.
  + Підрахунок кількості рядків.
  + Вивід результату на екран.

**Варіант 21.**

**Simple File Archiver  
Простий архіватор файлів**  
**Короткий опис:** Створення програми для стискання та розпаковування файлів з використанням будь-якого алгоритму стиснення.  
**Мінімальна функціональність:**

* + Стиснення файлів до архіву.
  + Розпакування файлів із архіву.
  + Логування операцій архівації.

**Варіант 22.**

**File Synchronization Tool  
Інструмент синхронізації файлів**  
**Короткий опис:** Розробка програми для синхронізації файлів між двома директоріями. Програма повинна підтримувати двонаправлену синхронізацію.  
**Мінімальна функціональність:**

* + Порівняння файлів у двох директоріях.
  + Копіювання нових або змінених файлів у потрібну директорію.
  + Логування процесу синхронізації.

**Варіант 23.**

**Custom Log Rotator  
Ротація журналів**  
**Короткий опис:** Реалізація програми для обробки великих журналів. Програма повинна розділяти файл журналу на кілька файлів за датами, а старі журнали архівувати.  
**Мінімальна функціональність:**

* + Розподіл вхідного журналу за датами.
  + Архівація файлів старше 14 днів.
  + Логування оброблених файлів.

**Варіант 24.**

**File-Based Key-Value Store  
Ключ-значення сховище на базі файлів**  
**Короткий опис:** Розробка легкого сховища даних, що використовує текстовий файл для зберігання пар "ключ-значення".  
**Мінімальна функціональність:**

* + Додавання нових записів.
  + Пошук записів за ключем.
  + Оновлення та видалення записів.

**Варіант 25.**

**Process Manager  
Менеджер процесів**  
**Короткий опис:** Створення програми для управління системними процесами. Менеджер повинен дозволяти відображати список процесів і завершувати їх.  
**Мінімальна функціональність:**

* + Відображення списку запущених процесів.
  + Завершення вказаного процесу.
  + Логування виконаних операцій.

**Варіант 26.**

**Concurrent  
Конкурентне копіювання файлів**  
**Короткий опис:** Реалізація програми для копіювання файлів з використанням потоків (threads) для пришвидшення процесу.  
**Мінімальна функціональність:**

* + Розбиття файлу на частини для копіювання.
  + Паралельне копіювання частин файлу.
  + Об'єднання частин у кінцевий файл.

### Взаємодія між процесами (IPC) та конкурентність

**Варіант 27.**

**Named Pipe Chat Application  
Чат через іменовані канали**  
**Короткий опис:** Реалізація програми для обміну повідомленнями між процесами за допомогою іменованих каналів. Кожен користувач програми виступає окремим процесом.  
**Мінімальна функціональність:**

* Створення іменованих каналів для передачі повідомлень.
* Підтримка одночасного обміну повідомленнями між кількома користувачами.
* Логування переданих повідомлень.

**Варіант 28.**

**Message Queue Manager  
Менеджер черги повідомлень**  
**Короткий опис:**Розробка програми для управління чергою повідомлень між процесами типу "виробник-споживач".  
**Мінімальна функціональність:**

* Створення черги повідомлень.
* Додавання повідомлень до черги виробником.
* Споживання повідомлень зі сторони споживача.

**Варіант 29.**

**Thread Pool Executor  
Виконувач пулу потоків**  
**Короткий опис:** Розробка легкого менеджера пулу потоків (Thread Pool Executor) для організації виконання паралельних задач.  
**Мінімальна функціональність:**

* Обмеження кількості одночасно виконуваних потоків.
* Додавання нових задач до пулу.
* Моніторинг стану виконання задач.

**Варіант 30.**

**Rust Actor System  
Акторна система в Rust**  
**Короткий опис:** Реалізація базової акторної системи для виконання конкурентних задач у Rust. Актори повинні обмінюватися повідомленнями для координації.  
**Мінімальна функціональність:**

* Створення акторів і передачі повідомлень між ними.
* Виконання асинхронних задач актором.
* Обробка помилок у виконанні задач.

### Безпека та криптографія

**Варіант 31.**

**Simple file encryptor / decryptor  
Простий шифрувальник / дешифрувальник файлів**  
**Короткий опис:** Створення програми для шифрування та дешифрування файлів із використанням криптографічних бібліотек Rust.  
**Мінімальна функціональність:**

* Шифрування файлів із використанням заданого алгоритму.
* Дешифрування файлів із правильним ключем.
* Логування операцій шифрування/дешифрування.

**Варіант 32.**

**Secure Communication Protocol  
Протокол безпечного зв’язку**  
**Короткий опис:** Реалізація кастомного протоколу зв’язку з підтримкою наскрізного шифрування для забезпечення конфіденційності передачі даних.  
**Мінімальна функціональність:**

* Шифрування повідомлень на стороні відправника.
* Розшифрування повідомлень на стороні отримувача.
* Підтримка автентифікації учасників.

**Варіант 33.**

**Authentication Server  
Сервер автентифікації**  
**Короткий опис:** Розробка невеликого сервера для автентифікації користувачів за допомогою сховища облікових даних.  
**Мінімальна функціональність:**

* Зберігання облікових даних користувачів.
* Перевірка автентичності запитів.
* Логування запитів на автентифікацію.

\*\*\*Список можливо доповнювати тематикою, що зацікавлює студента та задовольняє вимогам наступних методичних вказівок.

**Додаток 2**

**ПВНЗ «ЄВРОПЕЙСЬКИЙ УНІВЕРСИТЕТ»**

**Факультет інформаційних систем і технологій**

**К У Р С О В А Р О Б О Т А**

**на тему \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(назва теми)

**з дисципліни «Системне програмування»**

**Виконав:** студент \_\_\_\_\_ групи

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище, ініціали)

**Керівник** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище, ініціали)

Київ – 2025 р.