Національний технічний університет України
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Кафедра інженерії
програмного забезпечення
в енергетиці

НОВІТНІ МОВИ ПРОГРАМУВАННЯ

Робоча програма навчальної дисципліни (Силабус)

Реквізити навчальної дисципліни	
Рівень вищої освіти	***Перший (бакалаврський)***
Галузь знань	*Інформаційні технології*
Спеціальність	*121 Інженерія програмного забезпечення*
Освітня програма	*Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем в енергетиці*
Статус дисципліни	*Вибіркова*
Форма навчання	*Заочна*
Рік підготовки, семестр	*4-й курс, 1-й семестр*
Обсяг дисципліни	*4 кред. /120 год. (заочна форма: лекцій 6 год., лаб. 4 год., СРС 110 год.)*
Семестровий контроль/ контрольні заходи	*Залік, модульна контрольна робота*
Розклад занять	*http://schedule.kpi.ua/*
Мова викладання	*Українська/Англійська*
Інформація про керівника курсу / викладачів	Лектор: *д. е. н., професор Сігайов Андрій Олександрович* Практичні / Семінарські: Лабораторні: *Сігайов А. О.*
Розміщення курсу	GitHub Classroom, eCampus

Програма навчальної дисципліни

1 Опис навчальної дисципліни, її мета, предмет вивчання та результати навчання

Чому майбутньому фахівцю варто вчити саме цю дисципліну?

Це не одразу очевидно, але мова програмування Rust повністю ґрунтується на розширенні можливостей: незалежно від того, який код ви зараз пишете, Rust дає вам змогу досягти більшого, програмувати з упевненістю у ширшому спектрі доменів, ніж ви це робили раніше.

Візьмемо, наприклад, роботу на системному рівні, яка розглядає низькорівневі деталі управління пам'яттю, представлення даних та багатопотоковість. Традиційно ця сфера програмування вважається таємничою, доступною лише небагатьом обраним, які присвятили необхідні роки навчанню, щоб уникнути її сумнозвісних підводних каменів. І навіть ті, хто це практикує, роблять це з обережністю, щоб їх код не був відкритий для вразливостей, збоїв чи ушкодженням.

Rust руйнує ці бар'єри, усуваючи старі пастки та надаючи дружній, перевірений набір інструментів, які допоможуть вам на цьому шляху. Програмісти, яким потрібно "зануритися" в низькорівневе управління, можуть зробити це за допомогою Rust, не беручи на себе звичний ризик збоїв або дірок у безпеці та не вивчаючи тонкощів мінливого ланцюга інструментів. Ще краще, ця мова розроблена для того, щоб природно направити вас до надійного коду, ефективного з точки зору швидкості та використання пам'яті.

Програмісти, які вже працюють з низькорівневим кодом, можуть використовувати Rust для підняття своїх амбіцій. Наприклад, запровадження багатопотоковості в Rust є операцією з

відносно низьким ризиком: компілятор буде ловити для вас класичні помилки. І ви зможете займатися агресивнішою оптимізацією у своєму коді з упевненістю, що ви випадково не внесете збої чи вразливості.

Але Rust не обмежується програмуванням систем низького рівня. Він є виразним та ергономічним настільки, що робить досить приємним написання застосунків командного рядку, веб-серверів та багато інших видів коду - прості приклади будуть надані протягом цього курсу. Робота з Rust дозволяє сформувати навички, які передаються з одного домену до іншого; Ви можете вивчити Rust, написавши вебзастосунок, а потім застосувати ці самі навички у розробці коду для Raspberry Pi.

Цей курс повністю охоплює потенціал Rust, надаючи його користувачам розширені можливості. Це дружній та доступний матеріал, призначений для того, щоб допомогти вам підвищити рівень не тільки своїх знань про Rust, але й ваших досягнень та впевненості в собі як у програмісті взагалі. Тож занурюйтесь, готуйтесь до навчання - і ласкаво просимо до спільноти Rust!

Мета дисципліни. *Ознайомити студентів з сучасною мовою програмування Rust.*

Предмет дисципліни. *Огляд мови Rust, включаючи:*

- *володіння та запозичення, тривалість життя та риси;*
- *гарантована безпека програм;*
- *тестування, оброблення помилок та ефективний рефакторинг;*
- *розумні вказівники, багатопотоковість та зіставлення зі зразком;*
- *робота зі вбудованим менеджером пакетів Cargo для створення, тестування, документування коду та управління залежностями;*
- *просунуті засоби роботи з Unsafe Rust.*

Очікувані результати навчання.

Фахові компетентності.

ФК 16. Володіти скриптовими та декларативними мовами програмування.

Програмні результати навчання.

ПРН 15. Мотивовано обирати мови програмування та технології розробки для розв’язання завдань створення і супроводження програмного забезпечення.

ПРН 32. Застосовувати на практиці фундаментальні концепції, парадигми та основні принципи функціонування мовних, інструментальних і обчислювальних засобів інженерії програмного забезпечення.

2 Пререквізити та постреквізити дисципліни (місце в структурно-логічній схемі навчання за відповідною освітньою програмою)

Дисципліна вивчається у сьомому семестрі. Пререквізитами є курси “Комп'ютерна дискретна математика”, “Основи програмування” та “Архітектура системного програмного забезпечення”. Постреквізитів у даного курсу на бакалаврському рівні немає.

3 Зміст навчальної дисципліни

1. *Початок роботи з Rust. Програмування гри-відгадайки. Загальні концепції програмування.*
2. *Концепція володіння.*

3. *Конструкція Struct для структурування пов’язаних даних.*
4. *Перерахування та зіставлення зі зразком.*
5. *Управління проєктами, що зростають, за допомогою пакетів, скриньок і модулів.*
6. *Спільні колекції.*
7. *Оброблення помилок.*
8. *Узагальнені типи, риси та тривалість життя.*
9. *Автоматизовані тести.*
10. *Проєкт введення-виведення: побудова програми командного рядку.*
11. *Засоби функціонального програмування: ітератори та замкнення.*
12. *Докладніше про Cargo та Crates.io.*
13. *Розумні вказівники.*
14. *Багатопотоковість без остраху.*
15. *Засоби об'єктноорієнтованого програмування у Rust.*
16. *Зівставлення зі зразком.*
17. *Просунути засоби.*
18. *Фінальний проєкт: побудова багатопотокового веб-сервера.*

4 Навчальні матеріали та ресурси

Базова література:

Клабник, С., Николс, К. Программирование на Rust: СПб.: Питер, 2021. 592 с. URL: http://libgen.rs/book/index.php?md5=57FC451B6927457078601721158474B9

Додаткова література:

1. *McNamara, T. S. Rust in Action: Shelter Island, NY: Manning Publications, 2021. 456 с. URL: http://libgen.rs/book/index.php?md5=38FD5C0062ED46D7574BE9987952D655*
2. *Learn Rust in Y Minutes. URL: https://learnxinyminutes.com/docs/rust/ , Learn X in Y minutes: Scenic Programming Language Tours, (дата звернення: 31.08.21)*
3. *Amos. A half-hour to learn Rust. URL: https://fasterthanli.me/articles/a-half-hour-to-learn-rust , fasterthanli.me, (дата звернення: 02.01.21)*
4. *Biedert, R. Rust Language Cheat Sheet. URL: https://cheats.rs/ , Rust Language Cheat Sheet, (дата звернення: 25.04.21)*
5. *Rust Design Patterns. URL: https://rust-unofficial.github.io/patterns/ , Rust Design Patterns, (дата звернення: 03.01.21)*
6. *Nethercote, N. The Rust Performance Book. URL: https://nnethercote.github.io/perf-book/title-page.html , The Rust Performance Book, (дата звернення: 25.04.21)*
7. *Messier, R. Beginning Rust Programming: Indianapolis, IN: Wiley, 2021. 416 с. URL: http://libgen.rs/book/index.php?md5=0FB67E531BF1CB7B188BF84F817291AE*
8. *Wolverson, H. Hands-on Rust: Effective Learning through 2D Game Development and Play: Raleigh, NC: Pragmatic Bookshelf, 2021. 332 с. URL: http://libgen.rs/book/index.php?md5=9C11FE8FF7D64EE9E10BBC8817EE3882*

Навчальний контент

5 Методика опанування навчальної дисципліни (освітнього компонента)

1. *Початок роботи з Rust. Програмування гри-відгадайки. Загальні концепції програмування.*
    - *1.1. Встановлення Rust.*
        - *1.1.1. Встановлення Rust у Linux та macOS.*
        - *1.1.2. Встановлення у Windows.*
        - *1.1.3. Оновлення та деінсталяція Rust.*
        - *1.1.4. Усунення несправностей.*
        - *1.1.5. Локальна документація.*
    - *1.2. Програмування гри-відгадайки. Налаштування нового проєкту.*
    - *1.3. Обробка здогадки.*
        - *1.3.1. Зберігання значень за допомогою змінних.*
        - *1.3.2. Обробка потенційного збою за допомогою типа Result.*
        - *1.3.3. Друк значень за допомогою заповнювачів команди println!*
        - *1.3.4. Тестування першої частини.*
    - *1.4. Генерація таємного числа.*
        - *1.4.1. Використання скриньки для отримання більшої функціональності.*
        - *1.4.2. Генерація випадкового числа.*
    - *1.5. Порівняння здогадки з таємним числом.*
    - *1.6. Циклічна обробка здогадок.*
        - *1.6.1. Вихід з гри після правильної здогадки.*
        - *1.6.2. Обробка неприпустимого введення.*
    - *1.7. Загальні концепції програмування. Змінні та мінливість.*
        - *1.7.1. Розбіжності між змінними та константами.*
        - *1.7.2. Затінення.*
    - *1.8. Типи даних.*
        - *1.8.1. Скалярні типи.*
        - *1.8.2. Складені типи.*
    - *1.9. Функції.*
        - *1.9.1. Параметри функцій.*
        - *1.9.2. Інструкції та вирази в тілах функцій.*
        - *1.9.3. Функції, що повертають значення.*
    - *1.10. Коментарі.*
    - *1.11. Потік управління.*
        - *1.11.1. Вирази if.*
        - *1.11.2. Цикли.*
2. *Концепція володіння.*
    - *2.1. Що таке володіння?*
        - *2.1.1. Правила володіння.*
        - *2.1.2. Область видимості змінної.*
        - *2.1.3. Рядковий тип.*
        - *2.1.4. Пам’ять та її виділення.*
        - *2.1.5. Володіння та функції.*
        - *2.1.6. Повернені значення та область видимості.*
    - *2.2. Посилання та запозичення.*
        - *2.2.1. Змінювані посилання.*
        - *2.2.2. Посилання, що бовтаються.*
        - *2.2.3. Правила посилань.*
    - *2.3. Зрізовий тип.*
        - *2.3.1. Рядкові зрізи.*
        - *2.3.2. Інші зрізи.*

3. *Конструкція Struct для структурування пов'язаних даних.*
   - 3.1. *Визначення та інстанціювання Struct.*
     - 3.1.1. *Стисла ініціалізація полів: коли у змінних та полів однакові імена.*
     - 3.1.2. *Створення екземплярів з інших екземплярів за допомогою синтаксису Struct Update.*
     - 3.1.3. *Використання кортежів без іменованих полів для створення різних типів.*
     - 3.1.4. *Unit-подібні Struct без жодних полів.*
   - 3.2. *Приклад програми з використанням Struct.*
     - 3.2.1. *Рефакторинг за допомогою кортежів.*
     - 3.2.2. *Рефакторинг за допомогою Struct: додавання більшого сенсу.*
     - 3.2.3. *Додавання корисної функціональності за допомогою похідних рис.*
   - 3.3. *Синтаксис методів.*
     - 3.3.1. *Визначення методів.*
     - 3.3.2. *Методи з більшою кількістю параметрів.*
     - 3.3.3. *Пов'язані функції.*
     - 3.3.4. *Множинні блоки impl.*
4. *Перерахування та зіставлення зі зразком.*
   - 4.1. *Визначення перерахування.*
     - 4.1.1. *Значення Enum.*
     - 4.1.2. *Option Enum та його переваги перед значеннями Null.*
   - 4.2. *Вираз match як оператор потоку управління.*
     - 4.2.1. *Патерни, що прив'язуються до значень.*
     - 4.2.2. *Зіставлення з Option<T>.*
     - 4.2.3. *Збіги є вичерпними.*
     - 4.2.4. *Заповнювач _.*
   - 4.3. *Стислий потік управління за допомогою if let.*
5. *Управління проєктами, що зростають, за допомогою пакетів, скриньок і модулів.*
   - 5.1. *Пакети та скриньки.*
   - 5.2. *Визначення модулів для управління областю видимості та приватністю.*
   - 5.3. *Маршрути для посилання на елемент у дереві модуля.*
     - 5.3.1. *Демонстрація шляхів за допомогою ключового слова pub.*
     - 5.3.2. *Початок відносних шляхів за допомогою ключового слова super.*
     - 5.3.3. *Позначення Struct і Enum як публічних.*
   - 5.4. *Введення маршрутів у область видимості за допомогою ключового слова use.*
     - 5.4.1. *Створення ідіоматичних шляхів use.*
     - 5.4.2. *Надання нових імен за допомогою ключового слова as.*
     - 5.4.3. *Реекспорт імен за допомогою ключового слова pub.*
     - 5.4.4. *Використання зовнішніх пакетів.*
     - 5.4.5. *Використання вкладених маршрутів для очищення великих списків use.*
     - 5.4.6. *Оператор glob.*
   - 5.5. *Розділення модулів у різні файли.*
6. *Спільні колекції.*
   - 6.1. *Зберігання списків значень за допомогою векторів.*
     - 6.1.1. *Створення нового вектора.*
     - 6.1.2. *Оновлення вектора.*
     - 6.1.3. *Відкидання вектора відкидає всі його елементи.*
     - 6.1.4. *Читання елементів вектора.*
     - 6.1.5. *Перебирання значень вектора.*
     - 6.1.6. *Використання Enum для зберігання кількох типів.*
   - 6.2. *Зберігання тексту у кодуванні UTF-8 за допомогою рядків.*
     - 6.2.1. *Що таке тип String?*
     - 6.2.2. *Створення нового рядку.*

17.1.6. Коли використовувати небезпечний код.

17.2. Просунуті риси.

17.2.1. Специфікація типів-заповнювачів у визначеннях рис за допомогою пов'язаних типів.

17.2.2. Параметри узагальненого типу за умовчанням та перезавантаження операторів.

17.2.3. Повний синтаксис для усунення неоднозначності: виклик методів з однаковим іменем.

17.2.4. Використання супер-рис, які вимагають функціональності однієї риси всередині іншої.

17.2.5. Використання патерну Newtype для реалізації зовнішніх рис у зовнішніх типах.

17.3. Просунуті типи.

17.3.1. Використання патерну Newtype задля безпеки типів і абстракції.

17.3.2. Створення синонімів типів за допомогою псевдонімів типів.

17.3.3. Тип Never, який ніколи не повертається.

17.3.4. Динамічно змінювані типи та риса Sized.

17.4. Просунуті функції та замкнення.

17.4.1. Вказівники функцій.

17.4.2. Замкнення, що повертаються.

17.5. Макроси.

17.5.1. Відмінність між макросами та функціями.

17.5.2. Декларативні макроси за допомогою macro_rules! для загального метапрограмування.

17.5.3. Процедурні макроси для генерації коду з атрибутів.

17.5.4. Як написати власний налаштовуваний макрос derive.

17.5.5. Макроси, подібні до атрибутів.

17.5.6. *Макроси, подібні до функцій.*

18. Фінальний проєкт: побудова багатопотокового вебсервера.

18.1. Побудова однопотокового сервера.

18.1.1. Прослуховування TCP-з'єднання.

18.1.2. Читання запиту.

18.1.3. Детальніший погляд на HTTP-запит.

18.1.4. Написання відповіді.

18.1.5. Повернення реального HTML.

18.1.6. Перевірка запиту та вибіркова відповідь.

18.1.7. Невеличкий рефакторинг.

18.2. Перетворення нашого однопотокового сервера на багатопотоковий сервер.

18.2.1. Моделювання повільної відповіді у поточній реалізації сервера.

18.2.2. Покращення пропускної здатності за допомогою пула потоків.

18.3. Коректне відключення та очищення.

18.3.1. Реалізація риси Drop для об'єкта ThreadPool.

18.3.2. Подача потокам сигналу про припинення прослуховування завдань.

6 Самостійна робота студента/аспіранта

Студент витрачатиме 3-4 година на тиждень на самостійну роботу з матеріалом курсу.

Політика та контроль

7 Політика навчальної дисципліни (освітнього компонента)

Студенти отримують бали за правильне та вчасне виконання лабораторних робіт. Загальний рейтинг (кількість балів) складається з: 1) лабораторних робіт (у формі практичних завдань з програмування) 60%, 2) заліку 40%.
Наразі в курсі наявні три лабораторні роботи, кожне оцінюється до 20 балів. Студент повинен здати правильно виконану лабораторну роботу протягом двох тижнів з дня видачі завдання для отримання повної кількості балів, в іншому випадку застосовуються штрафні бали не більше 40% від загальної кількості за лабораторну роботу.

8 Види контролю та рейтингова система оцінювання результатів навчання (РСО)

Поточний контроль: МКР, лабораторні роботи
Календарний контроль: заочною формою навчання не передбачений.
Семестровий контроль: залік
Умови допуску до семестрового контролю: зарахування усіх лабораторних робіт, семестровий рейтинг не менше 40 балів.
Таблиця відповідності рейтингових балів оцінкам за університетською шкалою:

Кількість балів	*Оцінка*
100-95	Відмінно
94-85	Дуже добре
84-75	Добре
74-65	Задовільно
64-60	Достатньо
Менше 60	Незадовільно
Не виконані умови допуску	Не допущено

9 Додаткова інформація з дисципліни (освітнього компонента)

- *перелік питань, які виносяться на семестровий контроль (див. додаток до силабусу).*

Робочу програму навчальної дисципліни (силабус):

Складено Професор кафедри інженерії програмного забезпечення в енергетиці, д.е.н., професор А. О. Сігайов

Ухвалено кафедрою інженерії програмного забезпечення в енергетиці (протокол № 28 від 15 травня 2023 р.)

Погоджено Методичною комісією Навчально-наукового інституту атомної і теплової енергетики КПІ ім. Ігоря Сікорського (протокол № 9 від 26 травня 2023 р.)