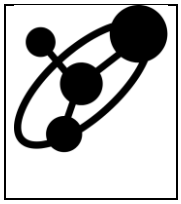
Національний технічний університет України
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»



Software Engineering in
Energy Industry Department

NOVEL PROGRAMMING LANGUAGES

Syllabus

Catalog Description

Higher education level	***First (Undergraduate)***
Knowledge field	*Information Technologies*
Profession	*121 Software Engineering*
Curriculum	*Software Engineering of Intelligent Cyberphysical Systems in Energy Industry*
Course status	*Elective*
Form of training	*Full-time*
Grade, term	*Fourth grade, fall semester*
Credits (hours)	*4 credits / 120 hours (36 hours of lectures, 18 hours of practice, 66 hours of individual assignments)*
Term control	*Exam, modular test*
Schedule	*http://schedule.kpi.ua/*
Teaching language	*Ukrainian/English*
Instructors	Lecturer: *DSc. (Econ), professor Andrii Sihaiov* Seminars: Laboratory work: *Andrii Sihaiov*
URL	GitHub Classroom, eCampus

Course Program

1 Course description, aim, subject, and expected outcomes

Why future specialist should study this course?

It wasn't always so clear, but the Rust programming language is fundamentally about empowerment: no matter what kind of code you are writing now, Rust empowers you to reach farther, to program with confidence in a wider variety of domains than you did before.

Take, for example, "systems-level" work that deals with low-level details of memory management, data representation, and concurrency. Traditionally, this realm of programming is seen as arcane, accessible only to a select few who have devoted the necessary years learning to avoid its infamous pitfalls. And even those who practice it do so with caution, lest their code be open to exploits, crashes, or corruption.

Rust breaks down these barriers by eliminating the old pitfalls and providing a friendly, polished set of tools to help you along the way. Programmers who need to "dip down" into lower-level control can do so with Rust, without taking on the customary risk of crashes or security holes, and without having to learn the fine points of a fickle toolchain. Better yet, the language is designed to guide you naturally towards reliable code that is efficient in terms of speed and memory usage.

Programmers who are already working with low-level code can use Rust to raise their ambitions. For example, introducing parallelism in Rust is a relatively low-risk operation: the compiler will catch the classical mistakes for you. And you can tackle more aggressive optimizations in your code with the confidence that you won't accidentally introduce crashes or vulnerabilities.

But Rust isn't limited to low-level systems programming. It's expressive and ergonomic enough to make CLI apps, web servers, and many other kinds of code quite pleasant to write — you'll find simple examples

of both later in the book. Working with Rust allows you to build skills that transfer from one domain to another; you can learn Rust by writing a web app, then apply those same skills to target your Raspberry Pi.

This course fully embraces the potential of Rust to empower its users. It's a friendly and approachable text intended to help you level up not just your knowledge of Rust, but also your reach and confidence as a programmer in general. So dive in, get ready to learn—and welcome to the Rust community!

Course Aim. *To familiarize students with modern Rust programming language.*

Course Subject. *An overview of Rust, including:*

- *ownership, borrowing, generic types, lifetimes, and traits;*
- *guaranteed code safety;*
- *testing, error handling, and effective refactoring;*
- *smart pointers, concurrency, patterns and matching;*
- *working with Cargo and Crates.io;*
- *advanced features of Unsafe Rust.*

Expected Outcomes.

Professional Competencies.

FC 16. To know script and declarative programming languages.

Program Learning Outcomes.

PLO 15. To choose programming languages and development technologies in a substantiated way for task solving of creating and maintaining software.

PLO 32. To apply in practice fundamental concepts, paradigms and basic functioning principles of linguistic, instrumental and computing tools of software engineering.

2 Course prerequisites (Where the course fits into our curriculum)

The course is taken in fall semester of final year. Discrete Math, Programming Basics, and Systems Software Architecture are the prerequisites. There is no required course that has this course as a prerequisite.

3 Course contents

1. *Getting Started. Programming a Guessing Game. Common Programming Concepts.*
2. *Understanding Ownership.*
3. *Using Structs to Structure Related Data.*
4. *Enums and Pattern Matching.*
5. *Managing Growing Projects with Packages, Crates, and Modules.*
6. *Common Collections.*
7. *Error Handling.*
8. *Generic Types, Traits, and Lifetimes.*
9. *Writing Automated Tests.*
10. *An I/O Project: Building a Command Line Program.*
11. *Functional Language Features: Iterators and Closures.*
12. *More About Cargo and Crates.io.*

13. *Smart Pointers.*
14. *Fearless Concurrency.*
15. *Object-Oriented Programming Features of Rust.*
16. *Patterns and Matching.*
17. *Advanced Features.*
18. *Final Project: Building a Multithreaded Web Server.*

4 Course textbooks and materials

Required reading:

Klabnik, S., Nichols, C. The Rust Programming Language: San Francisco, CA: No Starch Press, 2019. 560 c. URL: http://libgen.rs/book/index.php?md5=83D9E4EBE32F219D6436A438076216A3

Optional reading:

1. *McNamara, T. S. Rust in Action: Shelter Island, NY: Manning Publications, 2021. 456 c. 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 c. 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 c. URL: http://libgen.rs/book/index.php?md5=9C11FE8FF7D64EE9E10BBC8817EE3882*

Educational Content

5 Pedagogical advice

1. *Getting Started. Programming a Guessing Game. Common Programming Concepts.*
   - 1.1. *Rust Installation.*
     - 1.1.1. *Installing Rust on Linux or macOS.*
     - 1.1.2. *Installing rustup on Windows.*
     - 1.1.3. *Updating and Uninstalling.*
     - 1.1.4. *Troubleshooting.*
     - 1.1.5. *Local Documentation.*
   - 1.2. *Programming a Guessing Game. Setting Up a New Project.*
   - 1.3. *Processing a Guess.*

6 Individual Assignments

Students are expected to spend 3-4 hours a week outside of class on course material.

Course Rules and Assessment Policy

7 Course study rules

Students receive points relative to the correct and timely completion of coursework. The total grade consists of: 1) laboratories (programming assignments) 60%, 2) final exam 40%.
Presently there are three programming assignments, each worth up to 20% of the total grade. Student have to submit correctly fulfilled assignment during fortnight period from the date of give out to obtain full score for it, otherwise penalty points are applied but not more than 40% of the total score for laboratory work.

8 Assessment policy

How students are assessed: modular test, programming assignments
Calendar control: conducted twice a term to monitor the current state of compliance with the requirements of the syllabus.
Term assessment: final exam
Admission condition of term assessment: all programming assignment submission, start score not less than 40 points.
Exam scores map to the course grade according to the table:

Score	*Grade*
100-95	Excellent
94-85	Very Good
84-75	Good
74-65	Satisfactory
64-60	Sufficient
Less than 60	Unsatisfactory
Conditions for exam admission not met	Not allowed

9 Additional topics

- *Exam questions (see appendix).*

Syllabus:

Developed by Software Engineering in Energy Industry Department Professor, Sc. D. Andrii Sihaiov

Approved by Software Engineering in Energy Industry Department (minutes #28 on May 15, 2023)

Endorsed by Methodical Commission of Heat Power Faculty (minutes #9 on May 26, 2023)