

1-2: Setting-up programming environment

Artem Pavlov, TII, Abu Dhabi, 03.05.2024

Install rustup

- Go to the rustup website: <https://rustup.rs>
- Follow instructions (select stable toolchain during installation)
- You should have rustup and cargo commands available in your terminal

Create a new project

- Create new project using `cargo new p12`
- Move to the created project folder
- Inspect created project and `src/main.rs` file in particular
- Compile the project using `cargo build --release`
- Find executable in folder `target/release` and execute it
- Run project using `cargo run --release`

Installing and using Nightly toolchain

- Install Nightly toolchain using `rustup` (read its help using `rustup help`)
- Compile and run the project using `cargo +nightly run --release`
- Switch default toolchain to Nightly using `rustup`
- Switch toolchain back to stable

Installing new targets and cross-compiling

- Install new `x86_64-unknown-linux-musl` target using `rustup`
- Build for the project for the installed MUSL target (read help for `cargo build`)
- Execute the resulting binary (look in for compiled binary in `target/x86_64-unknown-linux-musl/release/`)

Installing IDE

- Install Visual Studio Code:
<https://code.visualstudio.com>
- Open the project
- Install **rust-analyzer** plugin:
<https://rust-analyzer.github.io>
- Setup **rust-analyzer** to run rustfmt and Clippy on save

Testing rust-analyzer

- Try change formatting of the `main` function. Save it and ensure that code gets automatically formatted
- Insert new unused function `fn foo() {}` and verify that you get compiler warning in IDE
- Verify that you get Clippy warnings by adding the following function:
`pub fn bar(a: u64) -> u64 { a as u64 }`

GitHub repository for exercises

- Create new GitHub repository
- Make the repository public or invite me as a collaborator if it's private
- Create GitHub workflow for build, testing, checking formatting and Clippy lints by copying the `.github` folder from <https://github.com/newpavlov/tii-workshop-2025>

Creating pull request

- Move the `p12` project to the repository folder
- Create `Cargo.toml` file in the repository root with the following content:

```
[workspace]  
resolver = "2"  
members = [  
    "p12",  
]
```
- Move `p12/.gitignore` file to the repository root
- Create new branch (e.g. called `p12`) and commit the changes to it
- Create new pull request from the branch and ensure that it passes the workflow checks
- Check that the workflow catches bad formatting, compiler and Clippy warnings
- Request my review for the PR and after my approval merge it