Hochschule RheinMain
Faculty of Design Computer Science Media - Computer Science
Andreas Hellenbrand
Fabio Campos

Rust
Selected Topic in Computer Engineering
LV 7281
Summer 2024

Lab #04

1 rustlings

Take a look at:

- `05_vecs`
- `08_enums`
- `11_hashmaps`
- `12_options`
- `17_tests`

2 cargo, modules and crates

Until no we only have worked on small things. But as projects and code grow we need ways to organize. Please read Chapter 7, 'Managing Growing Projects with Packages, Crates, and Modules'[1] of the Rust Programming Language book to familiarize yourself with these concepts.
To conclude your readings, take a look at rustlings `10_modules`.

[1]`https://doc.rust-lang.org/book/ch07-00-managing-growing-projects-with-packages-crates-and-module`
`html`

3 Enumerations

1. Define an enum to represent different types of transportation. Each variant should include relevant information. Options could be:
   - Car with a speed field
   - Bus with seats
   - Trains with a number of wagons
   - Bikes of different types (e.g. Mountainbike, E-Bike, City-Bike, . . .)
   - . . .
2. Implement a function `summary()` for this enum that returns a message describing the transportation mode. Use match guards to differentiate between cases. For example:
   - Fast and slow cars (e.g., fast if speed $>$ 120 km/h).
   - Long and short trains.
   - . . .
3. Test your function with various instances of the Transportation enum to ensure it works as expected.

Feel free to try other things and extend the code with different functions as you like.

4 Collections

Sublist

- Define a function that takes two vectors and decides whether:
    1. The vectors are equal.
    2. One vector is a subset of the other.
    3. The vectors are disjoint (that is, they have no values in common).
- Choose an appropriate representation for the return type of your function.
- To test your program, use the Cargo test environment.
- Is there a collection type more suitable for this task than a vector? If so, update your code accordingly.

Wordcount

1. Create a Rust program that takes a string of text as input.
2. Implement a function that counts the number of times each word occurs in the text.
3. Print the count of each word.

Write Unit-tests for your code. Expand your programm to read an input via `stdin` or from an file.