Discussion 1

January 30, 2023

Slides: Neil Kaushikkar

Goals For Today

- Tips to Succeed
- MP0 Preview
- More Data Types (Strings in Rust)

More Course Info

- Lectures will be posted on the website, on YouTube, and on Discord
  - Website: https://honors.cs128.org/lectures
- You can view the schedule on the website
  - This will barely change - most things are set for the semester
- OPTIONAL In-Person Discussion Section
  - Mondays 6-7:30pm (Siebel Center Room 1302)
  - We will:
    - Introduce MPs + give some hints
    - Review content from lecture
    - Collaborative Coding
    - Q&A and more!

How To Do Well In This Class

1. Watch lectures
    a. If you don't like the style, think they are too slow, too fast, etc..., please tell us and we'll try our best to improve
2. Start the MPs early
    a. We don't want you cramming close to the deadline!
3. Go to office hours / Ask questions in Discord!
    a. The entire course staff has experience coding in Rust, interpreting weird compiler errors, and has helpful hints on all the assignments

You Get Out What You Put In

- If you just want your honors credit, you can just complete the assignments and focus on your other classes
- If put time into understanding the language, understanding the assignments, asking questions, reading about concepts that make you curious, ...
  - You will be setting yourself up for success in the future
  - Arul and I teach Rust through the lens of other classes in the CS curriculum
    - The content we cover will come up later on and you will already be prepared for it!
- The assignments are meant to prepare you for your final project
  - If you struggle with the compiler and the syntax for the assignments, you will be well on your way to knowing how to debug errors in your project

Extensions & Getting Help

- If you find yourself spending 3+ hours on MPs or 1.5+ hours on homeworks and you are still struggling, PLEASE contact us!
  - This is an honors course - your other assignments take priority
- We have the 70% credit deadlines built in to help you if you have an especially busy week (you got sick, midterm szn, etc... it happens)
- If you need need an extension beyond that, contact Arul and I
  - We generally accept all extensions according to the student code

Collaboration

- You can work in groups of 2-3 people on homework assignments
- We ask that you work on MPs by yourself
  - MPs are more involved and are meant to challenge you
- Again - you get out what you put in
  - If you want to reply on partners, get code from your peers, you won't be challenging yourself and will not be learning as much

MP0 - Calculator

- You are building a terminal-based calculator
- This MP focuses heavily on error handling
- This MP will seem difficult/annoying (that is expected!)
  - Rust pushes a lot of the warnings/bugs/errors to you at compile time
  - You won't have to debug your code much, but...
  - It may feel like you are fighting with the compiler/autograder to get your code to just finally compile
  - Come to discussions and office hours!
  - Send your errors on Discord and staff/your peers can help decipher the compiler message!

MP0 - Calculator

- We've been working on improving error messages in the test cases
  - (you might have to scroll horizontally to read the entire message)
  - Please let us know if we can make error messages clearer!
- Demo!

Strings in Rust

- **&str** vs **String**
- We will have a lecture dedicated to why there is a difference after the lectures on Rust's memory management mechanism
- Generally, you can modify a variable of type **String** (but not one of type **&str**)
- Both types have functions to parse, trim, extract, split, etc...
- You might see **&String** - think of this as a pointer to a **String**

Creating Strings

- **&str**
  - Just type out your string surrounded by double quotes
  - Ex: **let y = “CS 128 Honors”;**
- **String**
  - **String::from(“hello”)**
  - **“hello”.to_string()**
  - **String::new()** - Creates an empty string
- Converting variables/values of numeric (and many other) types to **String**s
  - **<value>.to_string()**
  - Ex: **5.to_string()**
  - Ex: **let x = 3.1415; x.to_string()**

Converting Between Types

- **&str** to **String**
  - **.as_str()**
  - There are many more ways, but you will understanding those after the next couple lectures
- **String** to **&str**
  - **String::from(“hello”)**
  - **“hello”.to_string()**

Q&A