Lecture 2

Rust Basics

What we will cover today

Rust Programming Basics

- Variables
- Mutability
- Data types
- Control flow
- Functions

Optional Reading:

The Rust Book Chapter 3 – Common Programming Concepts

Declaring variables

```rust
// Defaults to i32
let x = 5;

// Stating the type explicitly
let x: u32 = 10;
```

In most common cases, you can declare variables without specifying the type explicitly.

Immutable variables

```rust
let x = 5;
println!("x is: {x}");

// This assignment is invalid
x = 6;
println!("x is: {x}");
```

In Rust, variables are immutable by default

Mutable variables

```rust
let mut x = 5;
println!("x is: {x}");

// This assignment is now valid
x = 6;
println!("x is: {x}");
```

We can declare mutable using the mut keyword

Scalar variable types

Rust has 4 primary scalar types:

- Integers
- Floating points
- Booleans
- Characters

Integers

Integers		
Length	Signed	Unsigned
8-bit	i8	u8
16-bit	i16	u16
32-bit	i32	u32
64-bit	i64	u64
128-bit	i128	u128
arch	isize	usize

Floating point numbers

```
let x = 2.0; // Defaults to f64

let y: f32 = 3.0;
```

Rust has 2 floating-point types – f32 and f64

Boolean

```
let condition: bool = false;
```

Pretty straightforward - true or false

Character type

```rust
let c = 'z';
let z: char = 'ℤ';
let rust = '🦀';
```

Defined with single quotes (not double quotes!)

- char type is 4 bytes in size
- Uses Unicode, can represent a lot more than ASCII

Conditionals

```rust
let x = 50;

if x < 50 {
    // Do something
} else if x == 50 {
    // Do something else
} else {
    // Do last thing
}
```

Rust uses
if … else if … else

Conditions don’t need brackets

Loops

Rust has 3 kinds of loops – loop, for, while

loop — Just keeps running until told to stop, for example with a break keyword

while — Checks a condition, and keeps looping while condition is true

for — Loops through a collection, such as an array, or loop for a specified number of times

loop

```rust
let x = 0;

loop {
    println!("x is {x}");
    if x == 50 { break; }
}
```

Loop keeps running until explicitly told to stop

Note: In this case, it is an infinite loop since x never reaches 50

while

```rust
let mut x = 0;

while x != 50 {
    println!("x is {x}");
    x += 1;
}
```

while loop checks if the condition is true before each iteration

for

```
let array = [1,2,3,4,5];

for number in array {
    println!("I love the number {number}");
}
```

for loop can loop through a collection, in this case it’s an array

for

```rust
let n = 10;

for number in 0..n {
    println!("I love the number {number}");
}
```

for loop can also loop through a given range of numbers!

Functions

```
fn plus_one(x: i32, y: bool) -> i32 {
    if y {
        return x+1;
    } else {
        return x+2;
    }
}
```

Functions - Returning

```
fn plus_one(x: i32, y: bool) -> i32 {
    x + 1
}
```

Statements:

- Instructions that do nothing, don't return anything
- e.g. let x = 5;

Expressions:

- Evaluates to a value, e.g. x+1

Implicit returning - Returns the value of the last expression

Recap

Declaring variables — Can choose to specify type or not

Mutability — Immutable by default, use mut keyword if you want mutable variables

Data types — Integers, floats, characters, booleans

Conditionals & Loops — if, else if, else & loop, while, for

Functions – Can use expression as return statement

Announcements

HW1 released today on PrairieLearn

- Due 1 week from now — Next Friday 02/07 23:59
- 70% credit for one week after the deadline

Remember to do the onboarding tasks!

- Check the onboarding form for details