# Contents

# Chapter 1

# Preface

These are my notes for Rust.

# Chapter 2

# Variables

## 2.1 Mutability

Variables by default are *immutable* in Rust. That is, we have to specify that we want a certain variable to be mutable.

```
    let var_name = value
    let mut var_name = value
```

A couple notes:

*(i)* Trying to mutate immutable variables (e.g. `let x...`) will result in an *immutability error* in the compiler.

*(ii)* Adding `mut` to a variable name (e.g. `let mut x...`) indicates that the variable is indeed mutable.

## 2.2 Constants

Constants are (by definition) *immutable*, and can ***not*** be made mutable using `mut`. They are declared as follows:

```
    const CONST_NAME: u32 = 60 * 60 * 3;
```

Note that constants require the type *and* value to be specified. Additionally, constants can only be set to constant expressions; i.e. you ***cannot*** set a constant to something computed at runtime.

***Note:*** *Naming convention is to use uppercase snake case.*

## 2.3 Shadowing

You can *shadow* variables in Rust. Consider the following function in `main.rs`:

```
1  fn main() {
2    let x = 5;
3    let x = x + 1;
4
5    {
6      let x = x * 2;
7      println!("x in the inner scope is: {x}");
8    }
9    println!("the value of x is: {x}");
10 }
```

At (2), `x = 5`.

At (3), `x = 5 + 1 = 6`.

At (6), `x = 6 * 2 = 12`.

At (7), we print the ***shadowed*** `x` (the `x` in the ***inner*** scope [`x = 12`]).

At (9), we print the `x` as normal.

### 2.3.1 `mut` v. Shadowing with `let`

We can also shadow variables using `let`. The following is perfectly legal:

```
  let spaces = "     ";
  let spaces = spaces.len();
```

Where the first and second `spaces` is a string and number type respectively.

However, doing the same using `mut` will produce a mismatched types error in the compiler:

```
  let mut spaces = "     ";
  spaces = spaces.len();
```

This is because we are not allowed to mutate a variable's type.

### 2.3.2 Summary

Rust variables are ***immutable*** by default and must be specified (using `mut`) if they are to be mutated.

Constants in Rust ***require*** a type annotation and can only be assigned to ***constant*** expressions (e.g. `10 * 10`).

Rust has shadowing with the expected behavior. However, a common thing to do in Rust (apparently) is to shadow a variable via `let`. Note that we ***cannot*** do this with variables specified with `mut`.

# Chapter 3

# Learning Journal

## 3.1 10/18/23

### Brief

*(i)* Started Rust notes.

*(ii)* Currently on `3.2 Data Types`.

*(iii)* Learned about how Rust variables worked.

*(iv)* Got PTSD from CS131 about how shadowing works.

### Major Takeaways

*(i)* Rust variables are immutable by default, and must be specified that they are mutable.

*(ii)* Constants require a type annotation.

*(iii)* Rust has shadowing with expected behavior.

*(iv)* Rust uses shadowing a lot via `let`.

*(v)* Variables with `mut` cannot be shadowed with *(iv)*.