Prof. D. Müller
HTW Dresden
SS 2025

Dresden, am 17.03.2025

Programmieren in Rust

Praktikumsaufgabe 2: Anweisungen, Ausdrücke und Operatoren

1. Korrigieren Sie das folgende Demo-Programm auf *zwei* verschiedene Arten durch jeweils eine minimale Änderung!

```
fn main() {
      let v = {
            let mut x = 1;
            x += 2
      };
      assert_eq!(v, 3);
      println!("Success!");
}
```

   Dabei ist `assert_eq!` ein Makro, das die Gleichheit der beiden als Parameter übergebenen Ausdrücke sicherstellt.

2. Warum gibt es in *Rust* im Gegensatz zu vielen anderen Sprachen keinen Inkrement-Operator, warum kann man also nicht `x++` schreiben, sondern muss das längere `x+=1` (oder klassisch `x=x+1`) nutzen? Zeigen Sie das anhand eines Beispiels mit einer Gegenüberstellung von *C*- und *Rust*-Code.

3. Berechnen Sie näherungsweise die Eulersche Zahl e! Nutzen Sie dazu die Reihendarstellung $e=\sum_{k=0}^{\infty}\frac{1}{k!}$! Wie viele Nachkommastellen der Zahl konnten Sie somit korrekt berechnen, wenn e ca. 2.71828182845904523536028747135266249775724709369995957496696762 ist?

4. Gegeben sei das folgende *Rust*-Programm, das einige Probleme/Ungenauigkeiten in sich trägt. Welche und wie viele finden Sie? Nutzen Sie dann `cargo check` oder `cargo fix`, um zu schauen, welche der Probleme (halb-)automatisch behoben werden können!

```
use std::io;

fn foo () -> f64
{
      return (2.7)
}

fn main() {
      let x = 3;
      let mut y:f64 = 3.1415;
      let BAD_NAME = y *3.0;
      while (16.0 > 15.0) {
            println!("Pi is approximately {}.", y);
      }
}
```