RUST

Konzepte der Programmiersprachen

Jassin Akhlaqi, Lukas Waller

Überblick

- von Rostpilze (engl. *rust*)*
- Designer: Graydon Hoare, The Rust PD
- Seit 2012 (public) bzw. 2015 (stable)
- Open Source
- Kernprinzipien
  - Speichersicherheit
  - Performanz

* https://www.reddit.com/r/rust/comments/27jvdt/internet_archaeology_the_definitive_endall_source/

Grundlagen

- Rust Compiler `rustc`
- Versionsmanagement `rustup`
- Paketmanager `cargo`
- `crates.io`

```rust
fn main() {
    println!("Hello, world!");
}
```

Typsystem

- Stark typisiert
- Statisch typisiert
- Variablen standardmäßig immutable (`mut`)

```
fn main() {
    let a:u32 = 10;
    let b:u16 = 90;
    let ab = a + b;
    a = 15;
}
```

```
fn main() {
    let mut a:u32 = 10;
    let b:u16 = 90;
    let ab = a + b as u32;
    a = 15;
}
```

Rust Prinzipien: Ownership

- Jede Ressource ein Besitzer
- Besitzer wird entfernt
  => wird der Speicher freigegeben
  (kein Garbage Collector nötig)
- Kann übertragen werden (move)

```
fn main () {
        let s1  =  String::from( „Hallo, Rust!“ );
        let length  =  berechne_laenge(s1);

        println!(„Der Wert von s1 ist: {}", s1);
        println!("Die Länge des Strings ist: {}", length);
}

fn berechne_laenge(s: String) -> usize {
        s.len()         // Expression-based return
}
```

Rust Prinzipien: Borrowing

- Keine Besitzübergabe nötig
- Referenzübergabe (&T)
  als lesender Zugriff
- Mutables Borrowing (&mut T)

```
fn main () {
        let s1  =  String::from( „Hallo, Rust!“ );
        let length  =  berechne_laenge(&s1);

        println!(„Der Wert von s1 ist: {}", s1);
        println!("Die Länge des Strings ist: {}", length);
}

fn berechne_laenge(s: &String) -> usize {
        s.len()         // Expression-based return
}
```

Rust Prinzipien: Lifetime

- Lebensdauer seiner Referenz
- Meistens implizit
- Explizite Lifetime-Anmerkung (‘a)
- Kein Zeiger auf ungültige Daten

```
fn main () {
        let s1  =  String::from( „Hallo, Rust!“ );
        let result;
        {
            let s2 = String::from("Rust");
            result = laengstes_wort(&s1, &s2);
        }

        println!(„Das längste Wort ist: {}", result);
}

fn laengstes_wort<'a>(x: &'a String, y: &'a String)
        -> &'a String {
                if  x.len() > y.len() {
                        x
                } else {
                        y
                }
 }
```