Rust

Osnovni koncepti - 3. deo
Enum, strukture

Struktura

Struktura

- Prilagođeni tip podatka koji omogućava da grupišete više povezanih osobina.
- Slična je torci:
  - Razlike:
    - Imenuje svaki podatak → fleksibilnija je - ne morate da vodite računa na kojoj poziciji se nalazi neki podatak

Struktura - definisanje

```
struct User {
    name: String,
    username: String
    active: bool,
    age: i32,
}
```

Struktura - upotreba

```
fn main(){
    let user = User {
        name: String::from("Pera"),
        username: String::from("pera"),
        active: true,
        age: 25,
    }

    println!("Name: {}", user.name);
}
```

Struktura - sintaksa ažuriranja

```
fn main(){
    let user = User {
        name: String::from("Pera"),
        username: String::from("pera"),
        active: true,
        age: 25,
    }
    let mika = User {
        name: String::from("Mika"),
        ..user
    }
}
```

Tuple struktura

- Poseduju dodatno značenje koje daje struktura, ali nemaju imena povezana sa poljima nego samo tipove polja kao torke.
- Korisne su kada želite torci da date ime i učinite je drugačijim tipom od drugih torki.

```
struct Color(i32, i32, i32);

struct Point(i32, i32, i32);


fn main() {

    let black = Color(0, 0, 0);

    let origin = Point(0, 0, 0);

 }
```

Ispis instance strukture

- Makro *println*
  - {} - ne može, zato što se koristi *Display* format koji za strukturu nije definisan
    - Možete definisati *Display* metodu za strukturu
  - {:?} - može, koristi *Debug* format
    - Neophodno je da eksplicitno navedete da ovu funkcionalnost hoćete da koristite
      - *#[derive(Debug)]*
- Makro *dbg*
  - Koristi *Debug* format za štampanje
  - Prikazaće datoteku, vrednost izraza i broj linije

```rust
#[derive(Debug)]
struct User {
    name: String,
    username: String
    active: bool,
    age: i32,
}
```

```rust
fn main() {
      let user = User {
        name: String::from("Pera"),
        username: String::from("pera"),
        active: true,
        age: 25,
    }
    println!("{}", user);
    println!("{:?}", user);
    dbg!(&user);
}
```

Display metoda

```
impl std::fmt::Display for User {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "({}, {})", self.name, self.username)
    }
}
```

Metode

- Slične su funkcijama, za razliku od funkcija definisane su nad strukturom i njihov prvi parametar je uvek *self*.
  - Mogu da preuzmu vlasništvo nad *self,* da ga nepromenljivo ili promenljivo pozajmljuju.
  - Mogu da imaju isto ime kao i polja u strukturi.
  - Definišu se u implementacionom bloku, koji se zove isto kao i struktura.

```rust
struct User {
    name: String,
    username: String
    active: bool,
    age: i32,
}
impl User {
    fn print(&self) {
        println!("{} {} {}", self.name, self.username, self.age);
    }
}
```

```rust
fn main() {
      let user = User {
        name: String::from("Pera"),
        username: String::from("pera"),
        active: true,
        age: 25,
    }
    user.print();
}
```

Pridružene funkcije

- Sve funkcije definisane u implementacionom bloku
- Možete da definišete i funkcije koje nemaju *self* (zato nisu metode)
  - Koriste se kao konstruktori
  - Često se nazivaju *new*
  - *String::from(), String::new()*

```rust
#[derive(Debug)]
struct User {
    name: String,
    username: String
    active: bool,
    age: i32,
}

impl User {
    fn new(name: String, username: String, active: bool, age:i8) -> Self {
        Self{
            name,
            username,
            active,
            age,
        }
    }
    fn print(&self) {
        println!("{} {} {}", self.name, self.username, self.age);
    }
}

fn main() {
      let user = User::new(String::from("Pera"),String::from("pera"),true, 25);
      user.print();
}
```

Pridružena funkcija

Metoda

Enum

Enum

- Definisanje novog tipa nabrajanjem njegovih mogućih vrednosti.
- Predstavlja način da kažete da je vrednost jedna od mogućih vrednosti iz skupa.
- Možete definisati metode kao i za strukture.

Enum

```
enum BankAccountType {
    DIN,
    FOREIGN_CURRENCY,
}

fn main(){
    let accountType = BankAccountType::DIN;
}
```

Enum

- Vrednosti enuma mogu da čuvaju podatke.
- Svaka varijanta enuma može da ima vrednost različitog tipa.
    - *Primer:* možemo da kažemo da ako je račun devizni onda hoćemo da kažemo i koja valuta je u pitanju.

```
enum BankAccountType {
    DIN,
    FOREIGN_CURRENCY(String),
}

fn main(){
    let accountType = BankAccountType::DIN;
    let accountType2 = BankAccountType::FOREIGN_CURRENCY(String::from("EUR"));
}
```

Option enum

- Definiše ga standardna biblioteka. Kodira scenario u kojem vrednost može da bude nešto ili može da bude ništa.
- Sprečava greške koje su izuzetno česte u drugim jezicima.
  - Pr. pristup prvoj stavci prazne liste
- Rust nema null vrednost

```
enum Option<T> {

    None,

    Some(T),

}
```

Match

- Omogućava da uporedite vrednost sa nizom obrazaca i zatim izvršite kod na osnovu toga koji obrazac odgovara.
- Izraz koji se koristi u ovoj konstrukciji može da vrati vrednost bilo kog tipa.
- Mora da pokrije sve opcije.
  - *other* - sve ostale vrednosti
  - _ - bilo koja vrednost, ali se ne vezuje za vrednost. Govori Rust-u da nećemo koristiti vrednost, tako da Rust neće upozoriti na neiskorišćenu promenljivu.

```rust
enum Coin {
    Penny,
    Nickel,
    Dime,
    Quarter,
}

fn value_in_cents(coin: Coin) -> u8 {
    match coin {
        Coin::Penny => 1,
        Coin::Nickel => 5,
        Coin::Dime => 10,
        Coin::Quarter => 25,
    }
}
```

Match i Option

```
fn plus_one(x: Option<i32>) -> Option<i32> {
    match x {
        None => None,
        Some(i) => Some(i + 1),
    }
}

let five = Some(5);
let six = plus_one(five);
let none = plus_one(None);
```

IF LET

- Rukovanje vrednostima koje odgovaraju jednom obrascu, a sve ostale mogućnosti se ignorišu

```
let config_max = Some(3u8);
match config_max {
   Some(max) => println!("The maximum is configured to be {}", max),
   _ => (),
 }
```

```
let config_max = Some(3u8);
if let Some(max) = config_max {
    println!("The maximum is configured to be {}", max);
}
```

ZADATAK

Napraviti progam za evidenciju knjiga u biblioteci. Program treba da omogući:

- Unos nove knjige
- Prikaz svih knjiga
- Prikaz knjiga određenog žanra