Funkce a sbírky

A smrt kalkulačky lol

Funkce

- Kus kódu, který může být opakovaně spouštěn
- Jiné názvy - subrutina, procedura
- Šetří kód
- Může brát parametry, které programátor vyhodnotí a změní chování funkce
- Může vracet hodnotu (V Rustu jí vrací vždycky)

Syntaxe

```
fn nazev(parametr1: Typ, parametr2: Typ) -> TypVraceneHodnoty {

    prikaz1();
    prikaz2();

    vracena_hodnota

}
```

Sbírky

- Také nazývány kolekce nebo listy
- Jsou to seznamy hodnot, my jsme se už setkali s vektorem (tj. typ Vec)
- Slouží k přehlednému zpracování dat o počtu hodnot, který se může měnit
- Existují i sbírky, které používají klíče

V Rustu

- Vec - jednoduchý seznam
- HashMap - Seznam, kde má každý prvek klíč
- Range - Posloupnost čísel od..do, mimo čísla ‘do’. Explicitně se typ nepoužívá
- [Typ] nebo [Typ; počet] - pole, neměnný počet prvků
- &[Typ] - slice - náhled do sbírky. Proměnný počet prvků
- HashSet - jako HashMap, ale jen klíče

Poslední kalkulačka, zkráceně

```
#[macro_use] extern crate text_io;
use std::collections::HashMap;
fn main() {
    let mut stack = Vec::new();
    let mut operace: HashMap<String, fn(&mut Vec<f64>)> = HashMap::new();
    operace.insert("+".to_string(), plus);
    loop {
        let op: String = read!();
        if operace.contains_key(&op) {
               let fun = operace[&op];
               fun(&mut stack);
        } else if let Ok(cislo) = op.parse() { // nebo op.parse::<f64>()
               stack.push(cislo);
        } else { println!("chyba lol"); }
    }
}
fn plus(stack: &mut Vec<f64>) {
    let a = stack.pop().unwrap();
    let b = stack.pop().unwrap();

    stack.push(a + b);
}
```

Nahrání na git

git config --global ‘email@domena.cz’

ve složce Cargo projektu

git add .

git commit -m ‘první revize’

git remote add origin https://github.com/rust-gjk/nazev-repa

git push -u origin master