Typy - struktury a výčty

(De)serializace, Síť

Typ

- Druh nebo význam, kterých smí nabývat proměnná
- Definuje obor hodnot a zároveň i proveditelné operace
- V Rustu rozlišujeme několik kategorií:
  - Primitivní typy
  - Pole
  - Struktury
  - Výčty
  - Ukazatele
  - Tuply/N-tice
- Pomocí základních typů tvoří programátor typy nové

Primitivní

- i8, i16, i32, i64
- u8, u16, u32, u64
- f32, f64
- str
- char
- ()
- bool

Pole

- Kolekce/sbírky prvků
- T reprezentuje typ prvků, K klíče, N počet prvků
- [T; N]
- &[T]
- Vec<T>
  HashMap<K, T>
  HashSet<K>

Tuply

- Česky n-tice
- Zkombinování jednoho a více txpů
- Příklady:
  - () - jednotka, prázdný tuple
  - (i64) - “jednotice”
  - (String, char) - dvojice
  - (i8, Vec<String>, bool) - trojice
  - ! - prázdný set, značí divergenci

Ukazatele

- ‘Ukazují’ na data která jsou jinde
- &
- &mut
- Box<T> - paměť na heapu
- Rc<T> - sdílení paměti
- *const - nahý ukazatel
- *mut - mutabilní nahý ukazatel

Struktury

- Strukturově kombinuje jině datové typy
- Ty jsou nazývány členy a mají vlastní jméno
- Členy.se definují podobně jako proměnné, akorát bez ‘let’
- Exiatují i prázdné a n-ticové struktury, o těch možná později

Syntaxe

```
struct Nazev {

    clen1: i64,

    clen2: String,

    jiny_clen: Vec<String>,

}
```

Výčet

- Datový typ, který má několik variant
- Ve většině jazyků každá varianta může reprezentovat číslo, v Rustu může také obsahovat nehomogenní data
- Varianta může být prázdná, nebo mít formu tuplu, či struktury
- Varianta se získá pomocí syntaxe ‘Vycet::Varianta’, popř ‘Vycet::Varianta(jmeno)’ nebo ‘Vycet::Varianta { pole1, pole2, .. }’

Syntaxe

```
enum MujVycet {
    PrazdnaVarianta,
    Dvojice(char, i64),
    Struktura { clen1: char, clen2: i32 },
}
```