

# Osnovni koncepti

Paralelne i distribuirane arhitekture i jezici  
Zimski semestar, školska 2025./26.  
Branislav Ristić

# Simboličke adrese

- Vrednosti podrazumevano nepromenljive.
- Definicija i inicijalizacija:
  - `let name: type = value`
- Primer:
  - *01\_standard.rs*

# Promenljive

- Simboličke adrese čije su vrednosti promenljive.
- Definicija i inicijalizacija:
  - `let mut name: type = value`
- Primer:
  - *02\_variables.rs*

# Konstante

- Identifikatori kojima se dodeljena vrednost ne može menjati.
- Definisanje i inicijalizacija:
  - `const name: type = value`
- Primer:
  - *03\_constants.rs*

# Pitanje

- Kada koristiti:
  - Nepromenljive simboličke adrese?
  - Konstante?

Konstante (**const**) vs nepromenljive simboličke adrese (**let**)

Feature	Constants ( <b>const</b> )	Immutable Variables ( <b>let</b> )
<b>Mutability</b>	Always immutable	Immutable by default, can be made mutable
<b>Declaration</b>	<b>const</b>	<b>let</b>
<b>Type Annotation</b>	Mandatory	Optional (type is inferred)
<b>Evaluation Time</b>	Compile-time	Runtime
<b>Memory Usage</b>	Inlined into code (no memory)	Stored in memory during runtime
<b>Scope</b>	Global or module-level	Block-scoped (local to functions/blocks)
<b>Allowed Expressions</b>	Must be compile-time constants	Any valid runtime expression

# Shadowing

- Dozvoljeno je definisanje simboličkih adresa sa istim imenom više puta.
- Uvek je važeća simbolička adresa koja je poslednja deklarirana.
- Važi:
  - Dok se ne deklariraju nova sa istim identifikatorom.
  - Dok ne izađe iz opsega u kom je definisana.
- Primer:
  - `04_shadowing.rs`

## Tipovi podataka u Rust-u

Category	Type	Description	Example
Scalar	<code>i8, i16, i32, i64, i128, isize</code>	Signed integers (8 to 128 bits, or pointer-sized)	<code>let x: i32 = 10;</code>
	<code>u8, u16, u32, u64, u128, usize</code>	Unsigned integers (8 to 128 bits, or pointer-sized)	<code>let y: u64 = 100;</code>
	<code>f32, f64</code>	Floating-point numbers (32 and 64 bits)	<code>let z: f64 = 6.4;</code>
	<code>bool</code>	Boolean (true or false)	<code>let flag: bool = true;</code>
	<code>char</code>	Unicode scalar value (4 bytes)	<code>let letter: char = 'a';</code>
Compound	<code>tuple</code>	Fixed-size list of values of different types	<code>let tup = (1, 2.0, 'c');</code>
	<code>array</code>	Fixed-size list of values of the same type	<code>let arr = [1, 2, 3];</code>
Unit Type	<code>()</code>	Empty tuple, represents "no value"	<code>let unit = ();</code>

## Operacije nad skalarnim tipovima u Rust-u

Type	Operations	Example
<b>Integers</b> (i8, u8, etc.)	<ul style="list-style-type: none"> <li>- Addition (+)</li> <li>- Subtraction (-)</li> <li>- Multiplication (*)</li> <li>- Division (/)</li> <li>- Remainder (%)</li> <li>- Comparison (==, !=, &lt;, &gt;, &lt;=, &gt;=)</li> </ul>	<pre>let sum = 5 + 10; let diff = 8 - 3; let prod = 4 * 2; let quot = 9 / 2; let rem = 7 % 3; let is_equal = 5 == 5;</pre>
<b>Floating-Point</b> (f32, f64)	<ul style="list-style-type: none"> <li>- Same as integers</li> </ul>	<pre>let div = 6.4 / 3.2; let is_greater = 3.2 &gt; 2.1;</pre>
<b>Boolean</b> (bool)	<ul style="list-style-type: none"> <li>- Logical AND (&amp;&amp;)</li> <li>- Logical OR (  )</li> <li>- Logical NOT (!)</li> </ul>	<pre>let a = true; let b = false; let res = a &amp;&amp; b;</pre>
<b>Character</b> (char)	<ul style="list-style-type: none"> <li>- Comparison (==, !=, &lt;, &gt;, &lt;=, &gt;=)</li> </ul>	<pre>let is_equal = 'a' == 'b'; let is_less = 'a' &lt; 'z';</pre>

# *stdin*

- Primer:
  - *05\_stdin.rs*

# Tuple

- Torka.
- Grupisanje više vrednosti različitih tipova u jedan složen tip.
- Fiksne dužine.
- Jednom definisan, nepromenljiv.
- Definicija:
  - `let tup: (type1, type2, ...) = (value1, value2, ...)`
- Primer:
  - `06_tuple.rs`

# Array

- Niz elemenata istog tipa.
- Fiksne dužine.
- Definicija i inicijalizacija:
  - `let name: [type; no_elems] = [e11, e12, ...],`
  - `let name = [no_elems, elem];`
- Primer:
  - `07_array.rs`

# Array - traits

- *Copy.*
- *Clone.*
- *Debug.*
- *Iterator (implemented for [T; N], &[T; N] and &mut [T; N]).*
- *PartialEq, PartialOrd, Eq, Ord.*
- *Hash.*
- *AsRef, AsMut.*
- *Borrow, BorrowMut.*
- *Primer:*
  - *08\_array\_traits.rs*

# Funkcije

- Unutar svakog binarnog crate-a postoji *main* funkcija:

```
fn naziv_funkcije([param1, param2, ...]) ->  
tip_povrtne_vrednosti {  
  
Statements;  
  
}
```

- Primer:
  - *09\_function.rs*

# Iskaz i izraz (*statement and expression*)

- *Statement* (iskaz):
  - Instrukcije koje izvršavaju neku radnju i ne vraćaju vrednost.
    - Kreiranje promenljive i dodeljivanje vrednosti pomoću let.
    - Definisavanje funkcije.
- *Expression* (izraz):
  - Izračunava se do rezultujuće vrednosti

# Selekcija - *IF*

- if ~~statement~~-expression.
- if; else if; else.
- Primer:
  - *10\_if.rs*

# Selekcija - *MATCH*

- Izraz (vraća vrednost).
- Iscrpna pretraga.
- Analogan *switch* iskazu u C/C++.
- Koristi se i za *pattern matching*.
- Primer:
  - `16_match.rs`

# Iteracija - *LOOP*

- Takođe *expression*.
- Izvršava deo koda beskonačno puta ili dok se eksplicitno ne zaustavi (CTRL+C):
  - *break* ili *continue*.
- Primer:
  - *11\_loop.rs*

# Iteracija - *WHILE*

- *Statement.*
- Primer:
  - *12\_while.rs*

# Iteracija - *FOR*

- *Statement.*
- Primer:
  - *13\_for.rs*

# Vector

- Dinamička, sekvencijalna struktura.
- Sve vrednosti istog tipa.
- Neke od operacija:
  - *push*,
  - *get*.
- Primer:
  - *14\_vector.rs*

# HashMap

- Nelinearna, dinamička struktura.
- (Ključ, Vrednost).
- Neke od operacija:
  - *insert*,
  - *entry*,
  - *or\_insert*,
  - *get*.
- Primer:
  - *15\_hashmap.rs*

# Zadatak 1.

- Napisati algoritam u Rust programskom jeziku koji pretvara temperaturu iz °F u °C.
  - a) Zaokružiti vrednost na dve decimale.
  - b) Kao vrednost °F koristiti vrednost sa standardnog ulaza.

## Zadatak 2.

- Napisati algoritam u Rust programskom jeziku koji generiše prvih N-ti fibonačijev broj.
  - a) Generisati prvih N.

## Zadatak 3.

- Napisati algoritam u Rust programskom jeziku koji ispisuje da li je uneti broj prost.
  - Ispisati prvih N fibonačijevih brojeva.

## Zadatak 4.

- Napisati algoritam u Rust programskom jeziku koji omogućava korisniku da unese veličnu vektora, a zatim:
  - Dinamički popuniti vektor.
  - Ispisati sve elemente vektora.
  - Ispisati elemente vektora u obrnutom redosledu.
  - Ispisati element na indeksu koji je korisnik uneo. Ako nema elementa na traženom indeksu onda korisnika obavestiti o tome.
  - Ispisati sve elemente koji se nalaze na indeksu koji je deljiv sa 3.
  - Prebrojati koliko se elemenata nalazi na parnim, a koliko na neparnim indeksima.
    - Napomena: koristiti HashMap-u.

# Izvori

- Rust Community. “The Rust Programming Language - the Rust Programming Language.” Rust-Lang.org, 2018, [doc.rust-lang.org/book/](https://doc.rust-lang.org/book/).
- Rust Team. “Rust Programming Language.” Rust-Lang.org, 2018, [www.rust-lang.org/](https://www.rust-lang.org/).
- Rust Community. “Tour of Rust - Let’s Go on an Adventure!” Tourofrust.com, [tourofrust.com/](https://tourofrust.com/).

# Osnovni koncepti

Paralelne i distribuirane arhitekture i jezici  
Zimski semestar, školska 2025./26.  
Branislav Ristić