Vlasništvo

Paralelne i distribuirane arhitekture i jezici
Računarstvo visokih performansi
Zimski semestar, školska 2024/25.
Branislav Ristić

Stack i heap

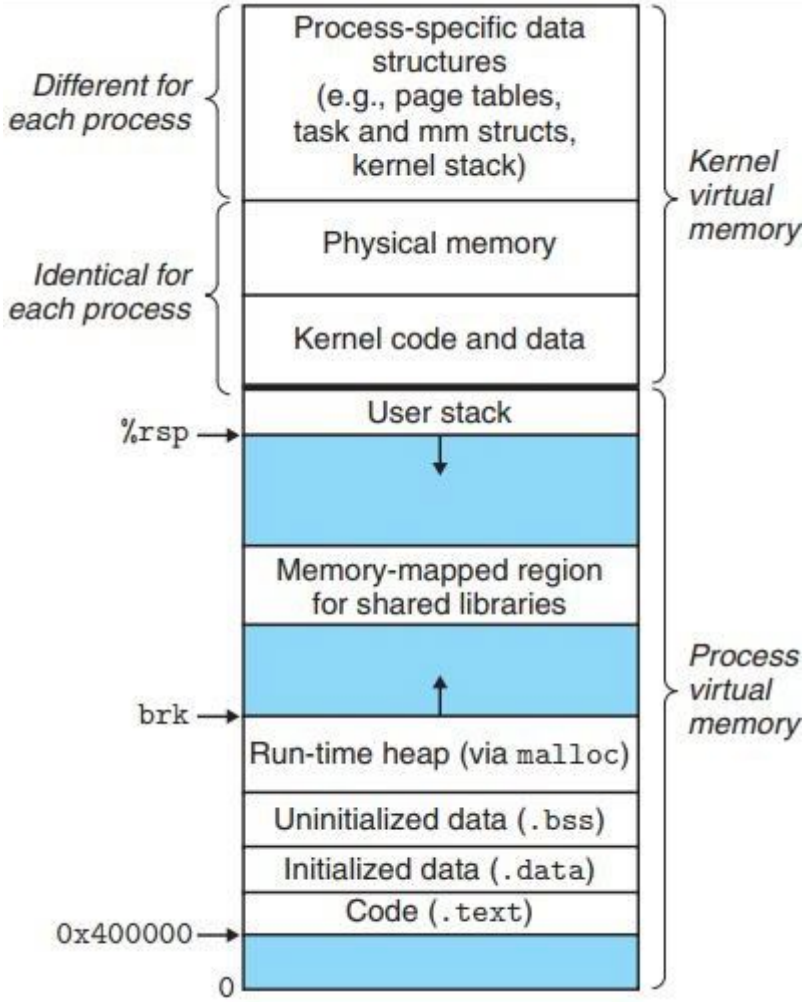
- Stack:
  - LIFO struktura
  - Mnogo manji u poređenju sa heapom
  - Alokacija:
    - Brza
    - Automatska
  - Životni vek varijable u okviru bloka gde je definisana
  - Brz pristup korišćenjem stack-pointera
  - Zaseban za svaku nit ponaosob
- Heap:
  - Nepostojeć poredak
  - Izrazito veći u odnosu na stack
  - Alokacija:
    - *malloc*
    - *new*
  - Varijabla živi dok se ne dealocira
  - Spor pristup preko pokazivača
  - Deljen između niti

Stack vs. Heap

Feature	Stack Memory	Heap Memory
Structure	LIFO (Last In, First Out)	No specific order, free-form memory
Size	Fixed, smaller size	Dynamic, larger size
Allocation	Automatic (fast)	Manual or garbage collected (slower)
Deallocation	Automatic when function exits	Manual or garbage collected
Lifetime	Limited to the scope of a function	Can persist as long as needed
Access Speed	Faster	Slower
Thread-Safety	Each thread has its own stack	Shared between threads, requires synchronization
Typical Use Cases	Function calls, local variables, recursion	Dynamic data structures, large allocations

Virtuelna memorija

- Sadrži stack i heap
- Memorija procesa
  - Stranice
  - Okviri
- Jedna od ključnih funkcionalnosti operativnog sistema
  - Izolovanost
  - Bezbednost



Vlasništvo

- Jedinstvena* funkcionalnost Rust programskog jezika.
- Primer:
  - *00_smart_pointers.cpp*
- Duboke implikacije na rad jezika.
- Omogućava memorijsku sigurnost bez GC-a.
- Predstavlja skup pravila koje govore kako upravljati memorijom.

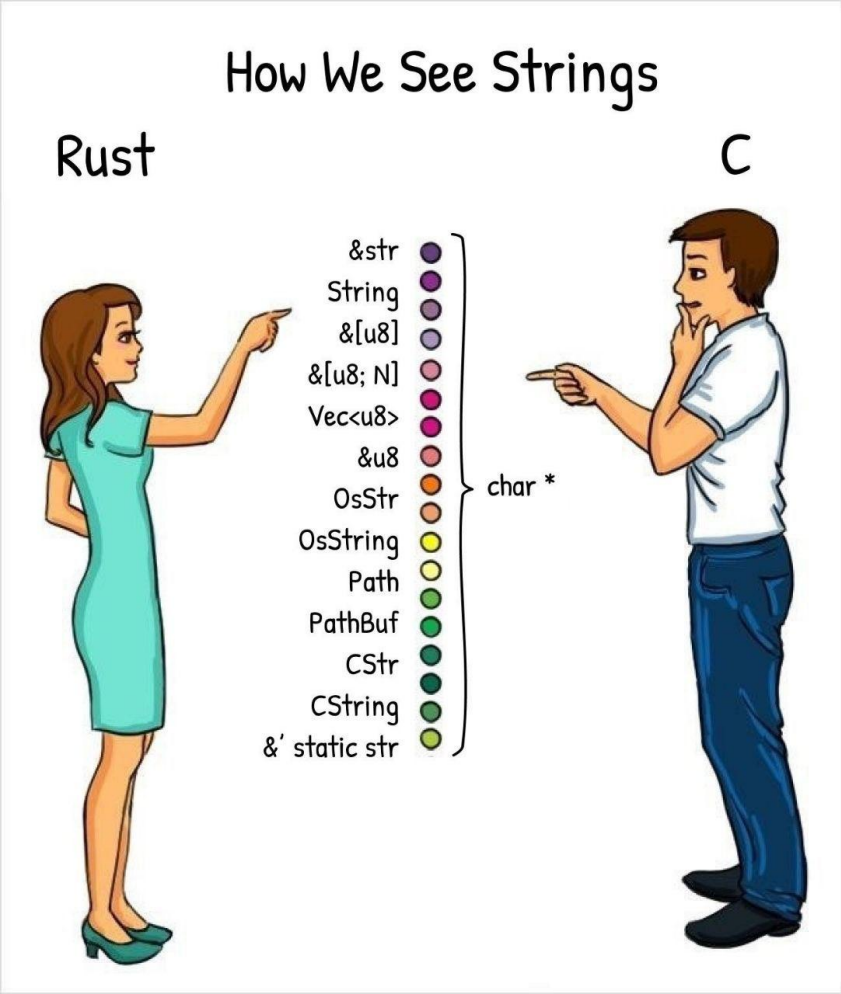
Pravila vlasništva

- Sve u Rust programu ima vlasnika
- Samo jedan vlasnik postoji za određeni resurs
- Ukoliko vlasnik izađe iz opsega, vrednost se dropuje

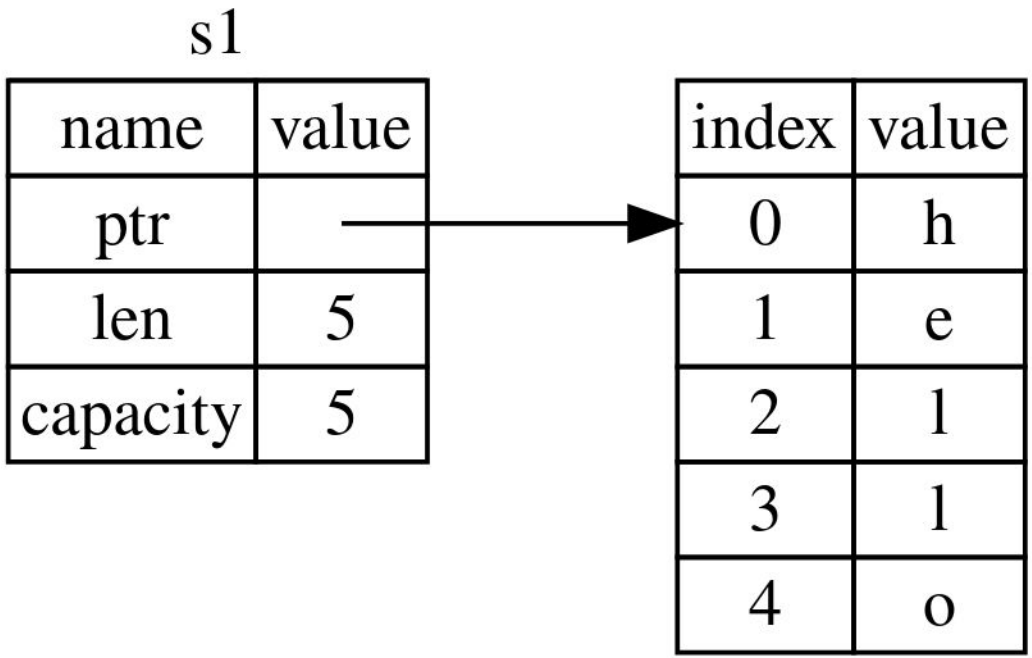
String literal i string struktura

- String literal
  - Sadržaj, te i veličina, je poznata u vreme kompajliranja
  - Hard-kodovana direkt u .rodata sekciju programa
  - Brzi, ali nepromenljive veličine
- String struktura
  - Sadržaj ne mora biti poznat u vreme kompajliranja
  - Spori, ali dinamičke veličine
- Primer:
  - *01_strings.rs*

Stringovi?

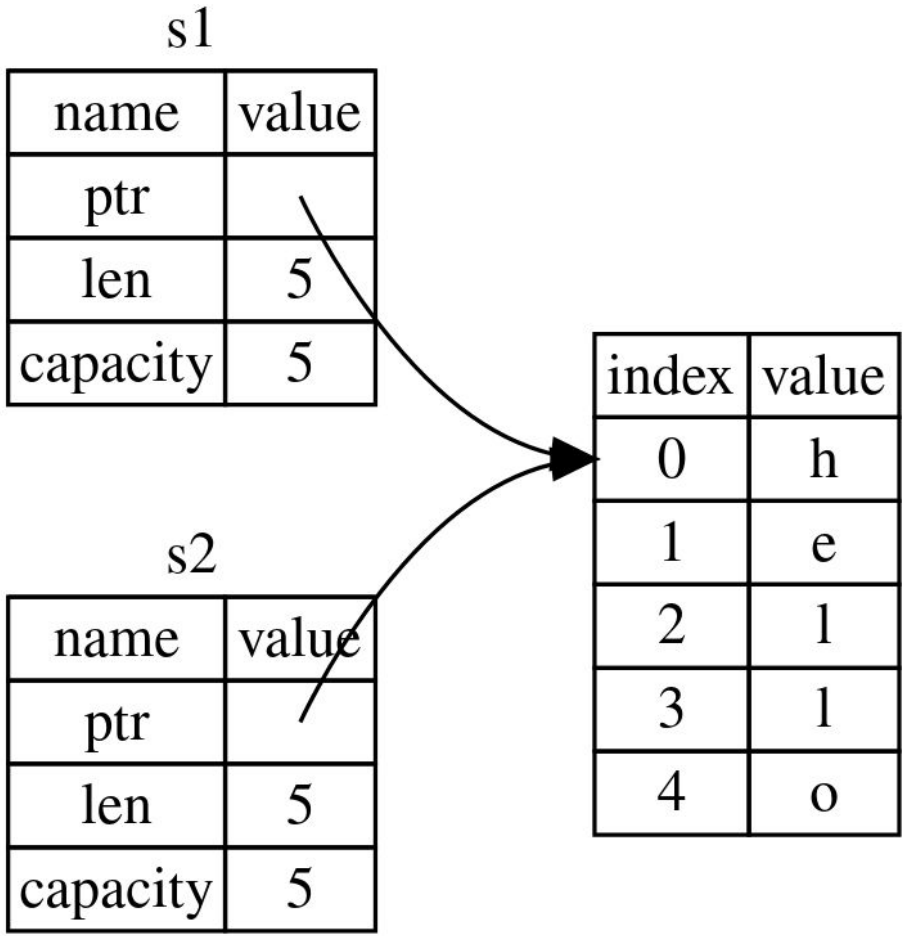

How We See Strings
Rust
C
&str
String
&[u8]
&[u8; N]
Vec<u8>
&u8
OsStr
OsString
Path
PathBuf
CStr
CString
&' static str
char *

Iza kulisa



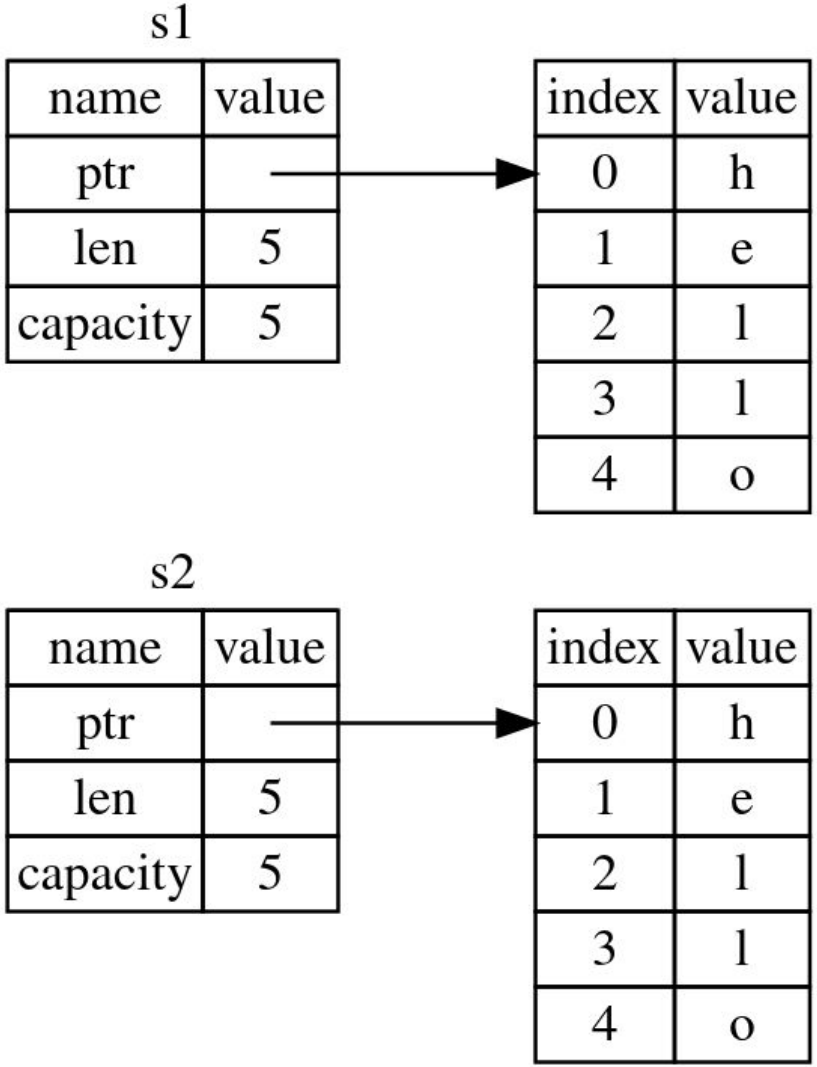
Iza kulisa

- Shallow copy



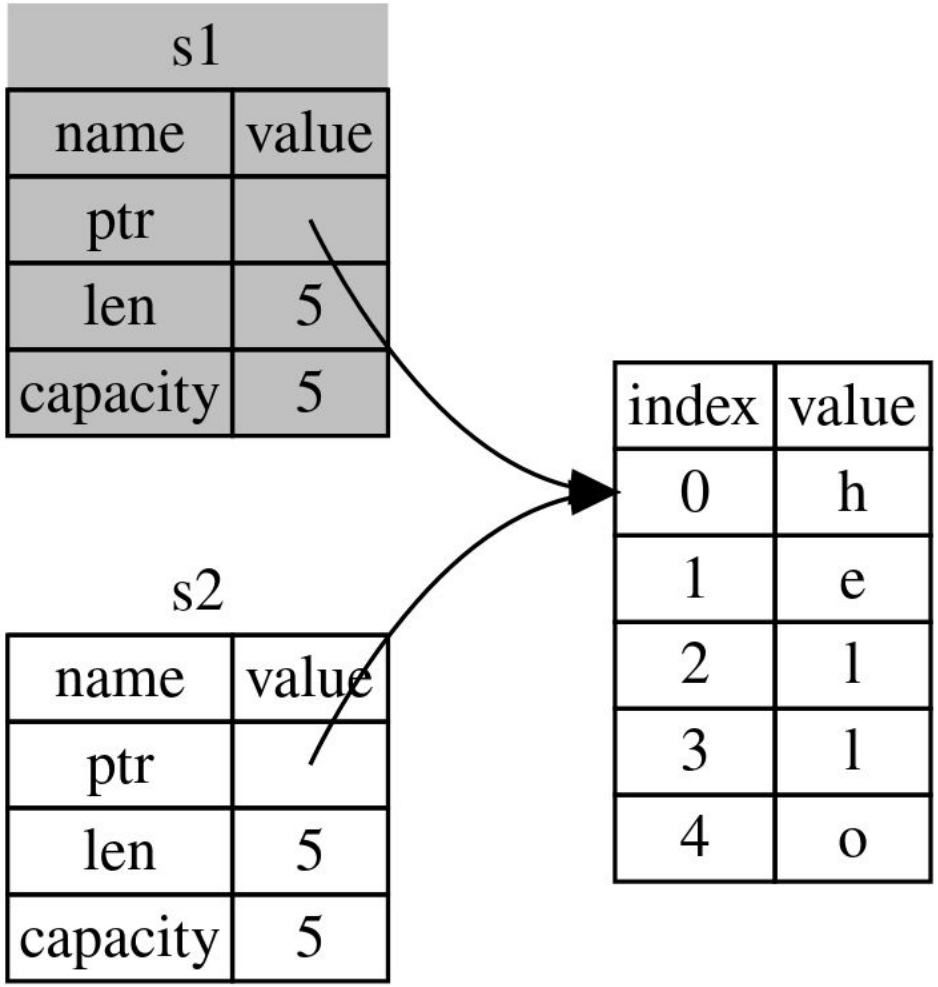
Iza kulisa

- Deep copy



Iza kulisa

- Primer:
- *02_strings_move.rs*



Stack-Only podaci

- Prilikom dodele, vlasništvo se ne prenosi
  - Razlog tome je poznavanje veličine prilikom kompajliranja
  - Lako se kopira bit po bit
  - Ne poseduje Drop osobinu koja ručno upravlja oslobađanjem resura na heap-u
- Podaci koji podležu ovome:
  - Prosti tipovi podataka (integer, floating-point, character, unit)
  - Torke (bez String podataka u torci)
  - Nizovi (bez String podataka u nizu)
  - Reference

Ne-Stack-Only podaci

- Prilikom dodele, vlasništvo se prenosi
- Podaci koji podležu ovome:
  - Heap-allocated tipovi (**String**, Vec<T>, Box<T>, poseduju custom Drop osobinu)
  - Custom tipovi (oni koji implementiraju sopstvenu Drop osobinu)
  - Ostali tipovi koji upravljaju vlasništvom (Rc<T>, Arc<T>, Cell<T>, RefCell<T>)

Vlasništvo i funkcije

- Primeri:
  - *03_ownership.rs*
  - *04_ownership_2.rs*
  - *05_ownership_3.rs*

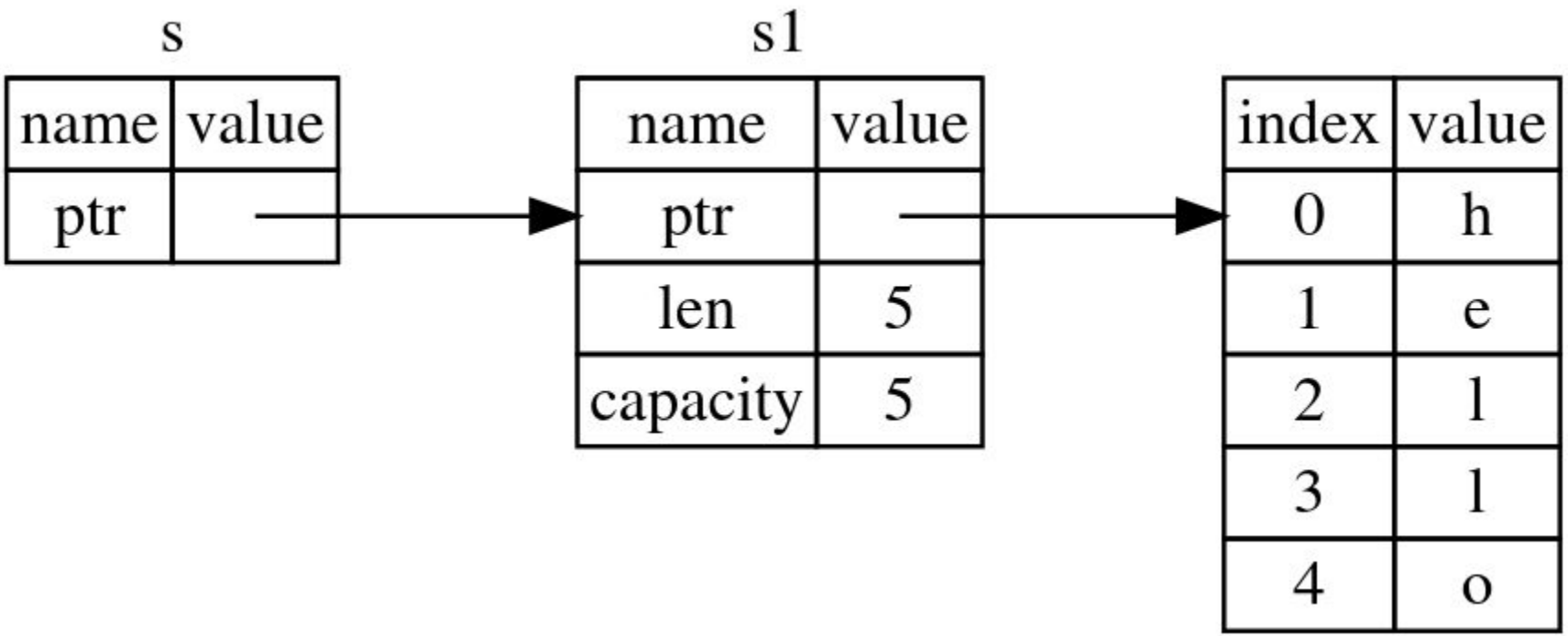
Reference i pozajmljivanje

- Problem prenosa vlasnistva
  - Lečiti vracanjem vrednosti iz funkcija
  - Sprečiti koriscenjem **referenci**

Reference

- Nalik pokazivačima u standardnim programskim jezicima
- Vlasnik je neko drugi, s tim da i ostali (oni kojima je pozajmljeno) mogu pristupiti vrednosti
- **Garantovano validna vrednost**
- Dozvoljeno
  - Ili 1 mutabilna referenca
  - Ili 1 do N nemutabilnih referenci
- Primer:
  - *06_borrow.rs*
  - *07_immutability.rs*
  - *08_mutability.rs*
  - *09_references.rs*
  - *10_references_2.rs*

Iza kulisa


s
name
value
ptr
s1
name
value
ptr
len
5
capacity
5
index
value
0
h
1
e
2
l
3
l
4
o

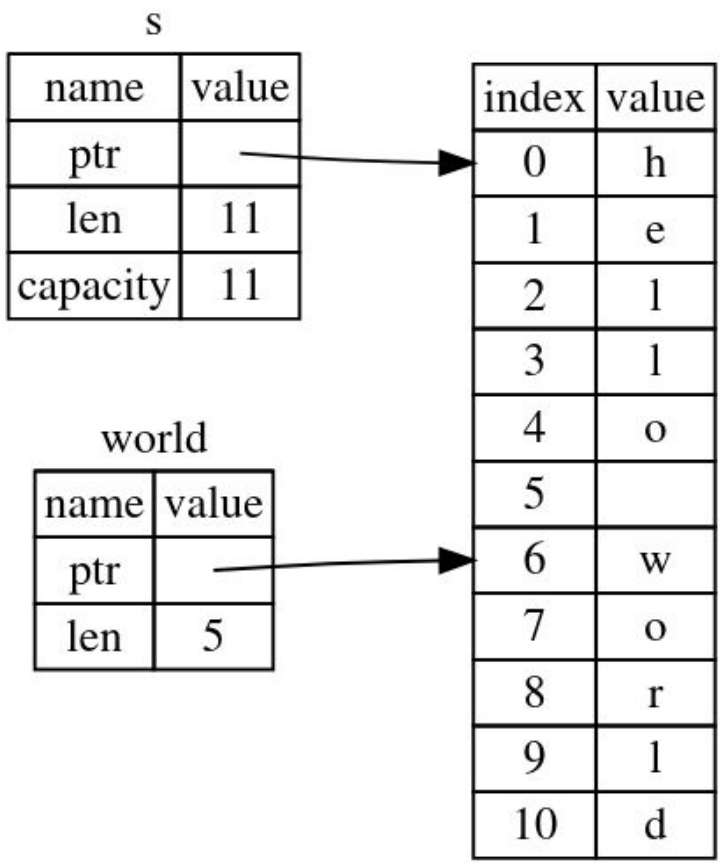
Viseći pokazivači

- Pokazivač koji pokazuje na memoriju koja je možda data nekome drugom
- Viseće reference u nije moguće napraviti
- Primer:
  - *11_dangling.rs*

Slice tip

- Omogućava referisanje na deo kolekcije (Vector/String/…)
  - Umesto na celu kolekciju
  - Mutabilni slice tip za String **ne postoji**
- Primer:
  - *12_manual_str.rs*
  - *13_manual_problems.rs*
  - *14_str_slice.rs*
  - *15_slices.rs*

Iza kulisa


s
name
value
ptr
len
11
capacity
11
world
name
value
ptr
len
5
index
value
0
h
1
e
2
l
3
l
4
o
5
6
w
7
o
8
r
9
l
10
d

Dodatni primeri

- Primeri:
  - *16_iterations.rs*
  - *17_tuple.rs*
  - *18_iterations_tuples.rs*
  - *19_vector.rs*
  - *20_hashmap.rs*

Zadatak 1.

- Napisati program u Rust programskom jeziku koji unetu rečenicu deli u vektor reči.

Zadatak 2.

- Napisati program u Rust programskom jeziku koji *inplace* obrće elemente u okviru niza.

Zadatak 3.

- Napisati program u Rust programskom jeziku koji sortira vektor označenih brojeva koristeći bubble sort algoritam.

Zadatak 4.

- Napisati program u Rust programskom jeziku koji sortira vektor označenih brojeva koristeći quick sort algoritam.

Zadatak 5.

- Napisati program u Rust programskom jeziku koji prima unos korisnika u obliku rečenice, a zatim:
  - Broji broj reči u rečenici
  - Pronalazi najdužu i najkraću reč u rečenici
  - Proverava da li je rečenica palindrom ili ne

Izvori

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

Vlasništvo

Paralelne i distribuirane arhitekture i jezici
Računarstvo visokih performansi
Zimski semestar, školska 2024/25.
Branislav Ristić