## Programski jezik Lua

Jovana Pejkić, Jana Jovičić, Katarina Rudinac, Ivana Jordanov

Prezentacija seminarskog rada u okviru kursa Metodologija stručnog i naučnog rada Matematički fakultet

jov4ana@gmail.com, jana.jovicic755@gmail.com, rudinackatarina@gmail.com, ivanajordanov47@gmail.com

# Sadržaj

- Nastanak jezika
  - Mesto nastanka i autori
  - Prethodnici i osnovni ciljevi
- Primena Lue
  - Primena Lue kao skript jezika
- Podržane paradigme
- Programska okruženja
- Osnovni koncepti
  - Tabele
  - Meta-tabele i meta-metodi
  - Zatvorenja
  - Iteratori
- Zaključak
- Literatura

### Mesto nastanka i autori

- Nastao 1993. na Katoličkom univerzitetu u Rio De Žaneiru
- Na portugalskom znači "mesec"
- Autori jezika: Roberto Jeruzalimski, Luiz Henrike de Figereido i Valdemar Keles





Slika: Katolički univerzitetu u Rio De Žaneiru

## Prethodnici i osnovni ciljevi

- Prethodnici Lue: jezici DEL i SOL
- Jednostavna sintaksa i semantika
- Opis podataka kao u SOL-u
- Portabilnost
- Mnogi koncepti pozajmljeni iz drugih programskih jezika

### Primena Lue

#### Lua može da se koristi na 3 načina:

- Kao skript jezik u sastavu aplikacija pisanih na drugom jeziku
  - Lua-C api za konfigurisanje
- Zajedno sa C-om
  - Najveći deo programa napisan u C-u
  - Lua se importuje kao biblioteka
- Kao samostalan jezik
  - Standardna biblioteka Lue (čine je biblioteke za rad sa stringovima, tabelama, fajlovima, modulima, matematičkim funkcijama, itd.)

# Primena Lue kao skript jezika

- CGILua
  - Alat za pravljenje dinamičkih veb stranica
  - Apstrakcija za Veb server
- Razvoj softvera zasnovan na komponentnom programiranju
  - Lua se koristi za spajanje komponenti
  - Ubrzava proces razvoja softvera
- Igrice
  - 2003. godine proglašena za najpopularniji jezik za pravljenje igrica
- Adobe Photoshop Lightroom

## Podržane paradigme

#### Proceduralna

- Svi mehanizmi Lue rade nad standardnom proceduralnom semantikom
- Većina programa napisanih u Lui su proceduralni

#### Funkcionalna

Biblioteka Lua Fun (funkcije map(), filter(), zip(), ...)

#### Objektno-orijentisana

 Sistemi klasa i objekata se kreiraju pomoću tabela i meta-tabela

# Programska okruženja

- Lapis HTML templating, jednostavno uvođenje middleware-a, upravljanje ORM modelima
- Sailor mogućnost pisanja klijentskog koda u Lui, prednost izvršavanje na raznim serverima
- Luvit nalik Node.js-u, koriste istu biblioteku za asinhrone I/O operacije
- Fengari implementacija Lua virtuelne mašine pisana u JavaScriptu

### **Tabele**

- Sastoje se iz parova ključ-vrednost
  - table[key] = value
- Kreiraju se uz pomoć konstruktora {}
  - rgb = {"red", "green", "blue"}
- Nakon kreiranja, treba ih dodeliti promenljivoj
  - da bi mogle da budu referisane
- Ako ne postoji referenca na tabelu
  - upravljač memorije briše tabelu
  - oslobađa memoriju koju je ona zauzimala

### Meta-tabele i meta-metodi

- Meta-tabela
  - je standardna tabela u Lui
  - sadrži skup meta-metoda
  - postavlja se pomoću funkcije setmetatable()
- Meta-metodi
  - menjaju ponašanje tabela
  - pozivaju se kada Lua izvršava određene operacije

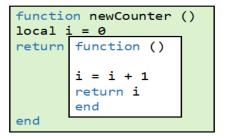
### Primer za operaciju sabiranja

```
-- meta i container su prethodno kreirane tabele
meta.__add = function (left, right )
  return left.value + right
end

setmetatable (container, meta)
result = container + 4
```

### Zatvorenja

- Omogućuju pristup lokalnim promenljivama funkcije nakon njenog izvršenja
- Primer zatvorenja je anonimna funkcija unutar funkcije
  - ona "vidi" lokalne promenljive funkcije kojom je okružena
  - može da nadživi postojeću funkciju



Slika: Primer zatvorenja

#### Iteratori

- Konstrukcije koje omogćuju prolazak kroz kolekciju
- Iteratori sa stanjem
  - Koriste zatvorenja kako bi zapamtili prethodno stanje u kom su bili
  - Čuvaju svoja stanja u okviru spoljašnje funkcije
- Iteratori bez stanja
  - Ne čuvaju sami svoja stanja
  - Isti iterator se može iskoristiti u više petlji, bez potrebe za pravljenjem novih zatvorenja

### Primer iteratora sa stanjem

```
function values ( t )
  local i = 0
  return function () i = i + 1; return t [ i ] end
end
t = {10 , 20 , 30}
for element in values ( t ) do
  print ( element )
end
```

### Primer iteratora bez stanja

```
a = {"one ", "two ", "three "}
for i , v in ipairs ( a ) do
  print (i , v )
end
```

### Zaključak

- lako je nastala za lokalne potrebe, Lua danas ima primenu u svetu
- Lua podržava objektno-orijentisanu, funkcionalnu i proceduralnu paradigmu
- Lua raspolaže malim brojem mehanizama, ali efikasnim

#### Literatura



Roberto Ierusalimschy (2003-2004) **Lua.org**, on-line at: https://www.lua.org/pil/7.1.html.

Etiene Dalcol (2014-2015) **Sailor Project**, on-line at: http://sailorproject.org/main/about.