

# Dizajn programskih jezika

Milena Vujošević Jančić

`www.matf.bg.ac.rs/~milena`

Dizajn programskih jezika  
Beograd, 18. oktobar, 2016.

# Pregled

- 1 Skript programiranje
- 2 Programiranje ograničenja

# Pregled

- 1 Skript programiranje
  - Namena skript jezika
  - Karakteristike skript jezika
  - Domeni upotrebe skript jezika
  - Literatura
- 2 Programiranje ograničenja

# Skript programiranje

- Skript jezici su u ekspanziji
- Najviše događanja na polju razvoja programskih jezika sada je u okviru razvoja skript jezika.
- Tradicionalni programski jezici su namenjeni za razvoj samostalnih aplikacija koje imaju za cilj da prime neku vrstu ulaza i na osnovu nje generisu odgovarajućí izlaz.
- Međutim, upotreba računara često zahteva manipulaciju i koordinaciju različitih programa.
- Ručno sprovođenje ovih poslova je naporno i sklono greškama.

# Skript programiranje

- Primer 1: sistem za obračun plata. Obračunavanje vremena sa kartica, papirnih izveštaja i unosa sa tastature, manipulacija bazama podataka, poštovanje pravnih i institucionalnih regulativa, priprema poreza, doprinosa i medicinskog osiguranja, pravljenje papirnih evidencija za arhivu...
- Primer 2: fotografija. Fotograf treba da skine fotografije sa digitalnog fotoaparata, konvertuje u odgovarajući format, rotira slike po potrebi, napravi manje koji su pogodne za brzo razgledanje, indeksira ih po vremenu, temi, napravi bekap na udaljenoj arhivi i ponovo inicijalizuje memoriju...
- Primer 3: kreiranje dinamičkih veb stranica. Autentikacija i autorizacija, komunikacija sa udaljenim uređajem, manipulacija sa slikama, komunikacija sa serverom, čitanje i pisanje HTMLa

# Skript programiranje

- Koordinaciju drugim programima moguće je ostvariti i u tradicionalnim programskim jezicima, kao što su Java ili C, ali to nije lako.
- Ovi jezici adresiraju efikasnost, lako održavanje, portabilnost i statičko otkrivanje grešaka. Sistem tipova je obično izgrađen oko koncepta kao što su celobrojne vrednosti fiksnih veličina, brojevi u pokretnom zarezu, karakteri i nizovi.
- S druge strane, skript jezici imaju za cilj da adresiraju fleksibilnost, brz razvoj, lokalnu prilagodljivost i dinamičke provere. Njihovi sistemi tipova, zbog toga teže da podrže više programske koncepte, kao što su tabele, katalozi, liste, datoteke...

# Skript jezici

- Skript jezik je programski jezik koji služi za pisanje skriptova.
- Skript je spisak (lista) komandi koje mogu biti izvršene u zadatom okruženju bez interakcije sa korisnikom.
- U prvobitnom obliku pojavljuju se kao komandni jezici operativnih sistema (npr Bash), danas imaju najrazličitije primene.
- Skript jezici imaju veliku primenu na Internetu (php, JavaScript, Dart, Perl...).
- Skript jezici mogu imati specifičan domen primene, ali mogu biti i jezici opšte namene (npr Python).
- Pošto se često koriste za povezivanje komponenti, nazivaju se „glue languages”

# Karakteristike skript jezika

- Skript paradigma je često specifična kombinacija drugih paradigmi, kao što su: objektno-orijentisana, proceduralna, funkcionalna (pa je to razlog što se skript paradigma ne prepoznaje uvek kao posebna paradigma).
- Nije uvek lako napraviti razliku između skript-jezika i drugih programskih jezika ali ipak imaju određene karakteristike na osnovu kojih se mogu izdvojiti od ostalih programskih jezika.
- Unix Shell (sh), Bash, JavaScript, PHP, Perl, Python, XSLT, VBScript, Lua, Ruby...

# Karakteristike skript jezika

- Interaktivno korišćenje/serijska obrada
- Skraćen zapis
- Deklaracije i pravila dosega
- Dinamičko tipiziranje
- Sistemske funkcije
- Manipulacija stringovima i poklapanje obrazaca
- Tipovi podataka visokog nivoa

## Interaktivno korišćenje/serijska obrada

- Većina skript jezika omogućava interaktivno korišćenje
- Neki skript jezici zahtevaju da se sve komande učitaju pre nego što počne obrada
- Većina skript jezika je interpretatorskog tipa

# Skraćen zapis

- Skraćen zapis zarad brzog razvoja i interaktivnog korišćenja
- Primer **Perl**, **Python**, **Ruby**

```
print "Hello, world\n"
```

**Java:**

```
class Hello {  
    public static void main(String[] args) {  
        System.out.println("Hello, world!");  
    }  
}
```

## Deklaracije i pravila dosega

- Promenljive se obično ne deklariraju, postoje jednostavna pravila dosega
- U nekim jezicima, sve promenljive su globalne (Perl), u drugim sve su lokalne (php)
- Python — svaka promenljiva je lokalna za blok u kojoj joj je dodeljena vrednost

## Dinamičko tipiziranje

- Usled nedostatka deklaracija, većina skript jezika dinamički određuje tipove podataka
- U nekim jezicima, tip se proverava neposredno pred korišćenje (php, Ruby, Python)
- U nekim jezicima, tip će se interpretirati drugačije u različitim kontekstima (Perl)

```
$a = "4"
```

```
print $a . 3 . "\n" # '.' je nadovezivanje: 43
```

```
print $a + 3 . "\n" # '+' je sabiranje: 7
```

# Sistemske funkcije

- U skript jezicima obično je omogućen lak pristup funkcionalnostima operativnog sistema
- Funkcije za ulaz/izlaz, manipulacija fajlovima i direktorijumima, upravljanje procesima, pristup bazama podataka, soketi, interprocesna komunikacija i sinhronizacija, zaštita i autorizacija, datum i sat, komunikacija preko mreže
- I u drugim programskim jezicima je to moguće, ali ne na tako jednostavan način

# Manipulacija stringovima i poklapanje obrazaca

- Skript jezici imaju svoje pretke u jezicima za procesiranje teksta i za generisanje izveštaja.
- Zbog toga imaju bogatu podršku za rad sa stringovima, za poklapanje obrazaca, pretragu i slično.
- Ovo se obično bazira na proširenim regularnim izrazima.
- Odrediti pravo ime funkciji osobina (tj naći koju osobinu broja  $n$  ispituje naredna funkcija u Python-u)

```
def osobina(n):  
    return not re.match(r'^.?$|^(..+?)\1+$', '1'*n)
```

# Tipovi podataka visokog nivoa

- Koriste se i osnovni tipovi podataka ali u okviru same sintakse i semantike skript jezika postoji i direktna podrška za više tipove podataka
- Podrška za skupove, rečnike, liste, torke...
- Sakupljač otpadaka

- Komandni jezici
- Procesiranje teksta
- Matematika i statistika
- Jezici opšte namene
- Jezici proširenja
- Jezici za www

## Domeni upotrebe skript jezika

- Komandni jezici (Shell) - eksanzija imena fajlova i varijabli, petlje, uslovi, cevi i redirekcija, funkcije, #! konvencija
- Procesiranje teksta i generisanje izveštaja (Sed, Awk) — orijentisani na rad sa stringovima: pretraga, zamena, ubacivanje, brisanje, poklapanje zagrada, Perl nastao sa idejom da kombinuje sed, awk i sh ali je izrastao u mnogo više od toga.

## Domeni upotrebe skript jezika

- Matematika (moderni naslednici jezika APL: Maple, Mathematica, Mathlab) - podrška numeričkim metodama, simboličkoj matematici, vizuelizaciji podataka, matematičkom modelovanju  
Statistika (S, R - open source) - podrška višedimenzionalnim nizovima, listama, mogu se proširivati infiksni operatorima, funkcionalno programiranje,
- Jezici opšte namene (Python, Ruby...)

## Domeni upotrebe skript jezika

- Jezici proširenja (Adobe grafički skup - Illustrator, Photoshop... dopuštaju dopune različitim skript jezicima: JavaScript, Visual Basic ili AppleScript; AutoCAD i Flash imaju svoje skript jezike za proširenje, skript jezik Lua se često koristi u razvoju skriptova za igrice, Majkrosoftovi alati obično koriste PowerShell, GIMP koristi Scheme...) - proširuje korisnost neke aplikacije dozvoljavajući korisniku da dodaje nove komande koristeći postojeće komande kao gradivne blokove
- Jezici za programiranje veba - skriptovi na strani servera (php, Ruby, PowerShell, Java servlets) i skriptovi na strani klijenta (JavaScript)

# Literatura

- Programming language pragmatics, Michael L. Scott, fourth edition, Elsevier.

# Pregled

## 1 Skript programiranje

## 2 Programiranje ograničenja

- Primene
- Podrška za programiranje ograničenja
- python-constraint
- Literatura

# Programiranje ograničenja

- Programiranje ograničenja je deklarativno programiranje
- Programiranje ograničenja (constraint programming) je programiranje u kojem se relacije između promenljivih zadaju u obliku različitih ograničenja.
- Za razliku od imperativne paradigme, gde je postupak pronalaženja rešenja dat u koracima, kod programiranja ograničenja, nije dat postupak već su postavljeni uslovi koje promenljive moraju da ispunjavaju.

# Programiranje ograničenja

- Od sistema se zatim očekuje da izračuna rešenje, tj da izračuna vrednosti promenljivih koje zadovoljavaju data ograničenja.
- Biblioteke programiranja ograničenja mogu imati različite pristupe za rešavanje problema, ali nije neophodno poznavanje algoritama koje ove biblioteke koriste.
- Rešavanje ograničenja vrši se različitim rešavačima, npr SAT i SMT

# Programiranje ograničenja

- Ograničenja se razlikuju od ograničenja u imperativnoj paradigmi. Na primer,  $x < y$  u imperativnoj paradigmi se evaluira u tačno ili netačno, dok u paradigmi ograničenja zadaje relaciju između objekata  $x$  i  $y$  koja mora da važi.
- Ograničenja mogu da budu različitih vrsta, na primer, ograničenja iskazne logike ( $A$  ili  $B$  je tačno), linearna ograničenja ( $x \leq 15$ ), ograničenja nad konačnim domenima
- Programiranje ograničenja ima primene pre svega u operacionim istraživanjima (tj u rešavanju kombinatornih i optimizacionih problema)

# Primene

- Programiranje ograničenja ima primene pre svega u operacionim istraživanjima (tj u rešavanju kombinatornih i optimizacionih problema)

# Programiranje ograničenja — primeri

- Kriptaritmetike su matematičke igre u kojima se rešavaju jednačine kod kojih su cifre brojeva zamenjene određenim slovima.
- Primer

```
SEND  
+MORE  
----  
MONEY
```

# Kriptoaritmetika

GREEN + ORANGE = COLORS  
MANET + MATISSE + MIRO + MONET + RENOIR = ARTISTS  
COMPLEX + LAPLACE = CALCULUS  
THIS + IS + VERY = EASY  
CROSS + ROADS = DANGER  
FATHER + MOTHER = PARENT  
WE + WANT + NO + NEW + ATOMIC = WEAPON  
EARTH + AIR + FIRE + WATER = NATURE  
SATURN + URANUS + NEPTUNE + PLUTO = PLANETS  
SEE + YOU = SOON  
NO + GUN + NO = HUNT  
WHEN + IN + ROME + BE + A = ROMAN  
DONT + STOP + THE = DANCE  
HERE + THEY + GO = AGAIN  
OSAKA + HAIKU + SUSHI = JAPAN  
MACHU + PICCHU = INDIAN  
SHE + KNOWS + HOW + IT = WORKS  
COPY + PASTE + SAVE = TOOLS

# Kriptoaritmetika

THREE + THREE + ONE = SEVEN  
NINE + LESS + TWO = SEVEN  
ONE + THREE + FOUR = EIGHT  
THREE + THREE + TWO + TWO + ONE = ELEVEN  
SIX + SIX + SIX = NINE + NINE  
SEVEN + SEVEN + SIX = TWENTY  
ONE + ONE + ONE + THREE + THREE + ELEVEN = TWENTY  
EIGHT + EIGHT + TWO + ONE + ONE = TWENTY  
ELEVEN + NINE + FIVE + FIVE = THIRTY  
NINE + SEVEN + SEVEN + SEVEN = THIRTY  
TEN + SEVEN + SEVEN + SEVEN + FOUR + FOUR + ONE = FORTY  
TEN + TEN + NINE + EIGHT + THREE = FORTY  
FOURTEEN + TEN + TEN + SEVEN = FORTYONE  
NINETEEN + THIRTEEN + THREE + TWO + TWO + ONE + ONE + ONE = FORTYTWO  
FORTY + TEN + TEN = SIXTY  
SIXTEEN + TWENTY + TWENTY + TEN + TWO + TWO = SEVENTY  
SIXTEEN + TWELVE + TWELVE + TWELVE + NINE + NINE = SEVENTY  
TWENTY + TWENTY + THIRTY = SEVENTY  
FIFTY + EIGHT + EIGHT + TEN + TWO + TWO = EIGHTY  
FIVE + FIVE + TEN + TEN + TEN + TEN + THIRTY = EIGHTY  
SIXTY + EIGHT + THREE + NINE + TEN = NINETY

# Programiranje ograničenja — primeri

- Rasporediti kraljice na šahovskoj tabli
- Rasporediti topove na šahovskoj tabli
- Rešiti sistem nejednakosti, npr  
 $x \in \{1, \dots, 100\}, y \in \{1, \dots, 100\}, x + y < 100, x > 10, y < 50$
- Odrediti položaje za najmanji broj predajnika tako da pokriva određeni prostor
- Definisati ponašanje semafora tako da protok saobraćaja bude najbolji

## Podrška za programiranje ograničenja

- Podrška za ograničenja su ili ugrađena u programski jezik (npr Oz, Kaleidoscope) ili su data preko neke biblioteke.
- Postoje različite biblioteke za programiranje ograničenja za jezike C, C++, JAVA, Python, za .NET platformu, Ruby
- Neke od biblioteka su IBM ILOG CPLEX, Microsoft Z3
- Programiranje ograničenja je često raspoloživo u okviru sistema za logičko programiranje
- Programiranje ograničenja u logici — Constraint logic programming (CLP)

# Koreni programiranja ograničenja

- Programiranje ograničenja je nastalo u okviru logičkog programiranja (Prolog II, Jaffar i Lassez, 1987)
- Logičko programiranje i programiranje ograničenja imaju puno zajedničkih osobina
- Većina Prolog implementacija uključuje jednu ili više biblioteka za programiranje ograničenja
- B-Prolog, CHIP V5, Ciao, ECLiPSe, SICStus, GNU Prolog, Picat, SWI Prolog

## Neke biblioteke

- Artelys Lakis (C++, Java, Python)  
<https://www.artelys.com/en/optimization-tools/kalis>
- Cassowary (C++, Java, JavaScript, Ruby, Smalltalk, Python)  
<https://constraints.cs.washington.edu/cassowary/>
- CHIP V5 (C++, C) [http:](http://www.cosytec.com/production_scheduling/chip/chip_technology.htm)  
[//www.cosytec.com/production\\_scheduling/chip/chip\\_technology.htm](http://www.cosytec.com/production_scheduling/chip/chip_technology.htm)
- Choco (Java) <http://choco-solver.org/>
- Cream (Java) <http://bach.istc.kobe-u.ac.jp/cream/>
- Disolver (C++)  
<http://research.microsoft.com/apps/pubs/default.aspx?id=64335>
- Gecode (C++, Python) <http://www.gecode.org/>
- Google or-tools (Python, Java, C++, .NET)  
<https://github.com/google/or-tools>
- JaCoP (Java) <http://jacop.osolpro.com/>
- JOpt (Java) <http://jopt.sourceforge.net/>
- Numberjack (Python) <http://numberjack.ucc.ie/>
- Minion (C++) <http://constraintmodelling.org/minion/>
- python-constraint (Python) <http://labix.org/python-constraint>
- Z3 (C++, Java, Python, C, C#) <https://github.com/Z3Prover/z3>

# Direktna podrška za programiranje ograničenja

- Claire <http://www.claire-language.com/>
- Curry(zasnovan na Haskell-u)  
<http://www-ps.informatik.uni-kiel.de/currywiki/>
- Kaleidoscope  
<https://constraints.cs.washington.edu/cip/kaleidoscope-asi.html>
- Oz <http://strasheela.sourceforge.net/strasheela/doc/01-Basics.html>
- Wolfram language <https://www.wolfram.com/language/>

# Programiranje ograničenja u Python-u

- Modul python-constraint podržava programiranje ograničenja na konačnom domenu.
- Programiranje ograničenja nad konačnim domenom sastoji se od tri dela
  - 1 Generisanje promenljivih i njihovih domena
  - 2 Generisanje ograničenja nad promenljivama
  - 3 Obeležavanje (labeling) — instanciranje promenljivih
- U okviru Pythona, rešenja se daju za sve promenljive, tako da je instanciranje podrazumevano

# Programiranje ograničenja nad konačnim domenom

Neka je:

$$\mathbf{a} = (a_1, a_2, \dots, a_n) \in D,$$

$$D = D_1 \times D_2 \times \dots \times D_n$$

gde su  $D_1, D_2, \dots, D_n$  neprazni skupovi sa konačno mnogo elemenata.

Nad promenljivom  $\mathbf{a} = (a_1, a_2, \dots, a_n)$  se može definisati skup ograničenja

$$\mathbf{C} = (C_1, C_2, \dots, C_m)$$

gde su sva ograničenja oblika:  $C_i(a_1, a_2, \dots, a_n)$

Pronalaženje jedne vrednosti (ili svih) promenljive  $\mathbf{a}$  koja ispunjava zadata ograničenja  $\mathbf{C}$  se naziva problem zadovoljivosti ograničenja (sa konačnim domenom) (finite domain) constraint satisfaction problem, **CSP**.

# Programiranje ograničenja u Python-u

```
import constraint

problem = constraint.Problem()

problem.addVariable("a", [1,2,3])
problem.addVariable("b", [4,5,6])

problem.getSolutions()

print resenja

[{'a': 3, 'b': 6}, {'a': 3, 'b': 5}, {'a': 3, 'b': 4},
 {'a': 2, 'b': 6}, {'a': 2, 'b': 5}, {'a': 2, 'b': 4},
 {'a': 1, 'b': 6}, {'a': 1, 'b': 5}, {'a': 1, 'b': 4}]
```

# Programiranje ograničenja u Python-u

```
import constraint

problem = constraint.Problem()

problem.addVariable("a", [1,2,3])
problem.addVariable("b", [4,5,6])

def o(a,b):
    if(2*a>b): return True

problem.addConstraint(o,"ab")
resenja = problem.getSolutions()

print resenja

[{'a': 3, 'b': 5}, {'a': 3, 'b': 4}]
```

# Opšta ograničenja

- `AllDifferentConstraint()` - različite vrednosti svih promenljivih
- `AllEqualConstraint()` - iste vrednosti svih promenljivih
- `constraint.MaxSumConstraint(s [,tezine])` - suma vrednosti promenljivih (pomnožena sa težinama) ne prelazi `s`
- `MinSumConstraint(s [,tezine])` - suma vrednosti promenljivih (pomnožena sa težinama) nije manja od `s`
- `ExactSumConstraint(s [,tezine])` - suma vrednosti promenljivih (pomnožena sa težinama) je `s`
- `InSetConstraint(skup)` - vrednosti promenljivih se nalaze u skupu `skup`
- `NotInSetConstraint(skup)` - vrednosti promenljivih se ne nalaze u skupu `skup`
- `SomeInSetConstraint(skup)` - vrednosti nekih promenljivih se nalaze u skupu `skup`
- `SomeNotInSetConstraint(skup)` - vrednosti nekih promenljivih se ne nalaze u skupu `skup`

# Primeri

SEND  
+MORE  
----  
MONEY

# SEND+MORE = MONEY

```
import constraint

problem = constraint.Problem()
# Definisemo promenljive i njihove vrednosti
problem.addVariables('SM',range(1,10))
problem.addVariables('ENDORY',range(10))

# Definisemo ogranicenje za cifre
def o(s,e,n,d,m,o,r,y):
    if(s*1000 + e*100 + n*10 + d + m*1000 + o*100 + r*10 + e)
       == (10000*m + 1000*o + 100*n + 10*e + y):
        return True

# Dodajemo ogranicenja za cifre na svim pozicijama
problem.addConstraint(o,"SENDMORY")

# Dodajemo ogranicenje da su sve cifre razlicite
problem.addConstraint(constraint.AllDifferentConstraint())
```

# SEND+MORE = MONEY

```
resenja = problem.getSolutions()

for r in resenja:
    print "  "+str(r['S'])+str(r['E'])+str(r['N'])+str(r['D'])
    print "  "+str(r['M'])+str(r['O'])+str(r['R'])+str(r['E'])
    print "="+str(r['M'])+str(r['O'])+str(r['N'])+str(r['E'])+str(r['Y'])
```

```
python sendmoremoney.py
9567
+1085
=10652
```

# SEND+MORE = MONEY — Prolog

```
sendmoremoney(Vars) :- Vars = [S,E,N,D,M,O,R,Y], %generisanje promenljivih
    Vars :: 0..9, %definisanje domena
    S #\= 0, %ograničenja
    M #\= 0,
    all_different(Vars),
    1000*S + 100*E + 10*N + D
+ 1000*M + 100*O + 10*R + E
#= 10000*M + 1000*O + 100*N + 10*E + Y,
    labeling(Vars). %instanciranje
```

# Literatura

- Concepts of programming languages, Robert W. Sebesta
- Programming Language Pragmatics, Michael L. Scott
- Slajdovi prof Dušana Tošića sa istoimenog kursa
- <http://labix.org/doc/constraint/>
- <https://pypi.python.org/pypi/python-constraint>
- [http://www.hakank.org/constraint\\_programming\\_blog/](http://www.hakank.org/constraint_programming_blog/)