Stilovi i tehnike programiranja

Opšti principi dobrog programiranja

- Correctness
- User-friendliness
- Efficiency
- Readability
- Modifiability
- Robustness
- Documentation

Prološki programi

- Proceduralno i deklarativano rezonovanje
- Korišćenje rekurzije
  - Trivijalan (granični) slučaj
  - Opšti slučaj

```
maplist([],_,[]).
maplist([X|Tail],F,[NewX|NewTail]):-
        G =.. [F,X,NewX],
       call(G),
       maplist(Tail,F,NewTail).
```

```
square(X,Y):-Y is X*X.
```

```
?- maplist([2,6,5,12],square,Squares).
Squares = [4, 36, 25, 144].
```

Prološki programi

- Generalizacija
  - eightqueens(Pos) vs. nqueens(Pos,N)
  - donald(…) vs. puzzle(L1,L2,L3)
- Korišćenje slika
  - Grafička reprezentacija problema – relacije među objektima
  - Strukture u obliku stabla
  - Tumačenje deklarativnog značenja pomoću slika

Stil programiranja

Pravila dobrog stila

- Kratke klauze sa par ciljeva
- Kratke procedure
- Mnemonička imena
- Izgled programa(beline, prazne linije, uvlačenja…)
- Obazrivo korišćenje **cut** i **not** operatora
- Korišćenje **assert** i **retract** smanjuju transparentnost ponašanja programa
- Korišćenje **;** prikriva značenje kaluze

```
merge( List1, List2, List3) :-
   List1 = [ ], !, List3 = List2;
   List2 = [ ], !, List3 = List1;
   List1 = [X | Rest1],
   List2 = [Y | Rest2],
   ( X < Y, !,
     Z = X,
     merge( Rest1, List2, Rest3);
     Z = Y,
     merge( List1, Rest2, Rest3) ),
  List3 = [Z | Rest3].
```

```
merge( [ ], List, List)  :-  !.

merge( List, [ ], List).

merge( [X | Rest1], [Y | Rest2], [X | Rest3] ) :-
   X < Y, !,
   merge( Rest1, [Y | Rest2], Rest3).

merge( List1, [Y | Rest2], [Y | Rest3] ) :-
   merge( List1, Rest2, Rest3).
```

Stil programiranja

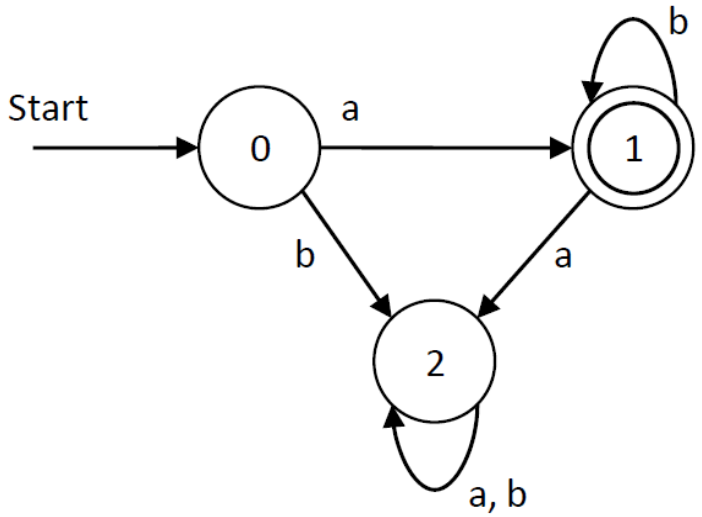
- Tabularna organizacija dugačkih procedura
- Komentarisanje - %, /* */
- Debagovanje
  - Testiranje malih delova programa
  - Dodaci za debagovanje
  - *tracing*
- Unapređenje efikasnosti
  - Dublje poznavanje problema
  - Korišćenje odgovarajućih struktura
  - Definisanje akumulatora

```
?- trace,conc([a,b],[c,d],L).
   Call: (11) conc([a, b], [c, d], _24162) ? creep
   Call: (12) conc([b], [c, d], _24672) ? creep
   Call: (13) conc([], [c, d], _24722) ? creep
   Exit: (13) conc([], [c, d], [c, d]) ? creep
   Exit: (12) conc([b], [c, d], [b, c, d]) ? creep
   Exit: (11) conc([a, b], [c, d], [a, b, c, d]) ? creep
L = [a, b, c, d].

[trace]  ?- notrace.
true.
```

Složene strukture

Deterministički konačni automat

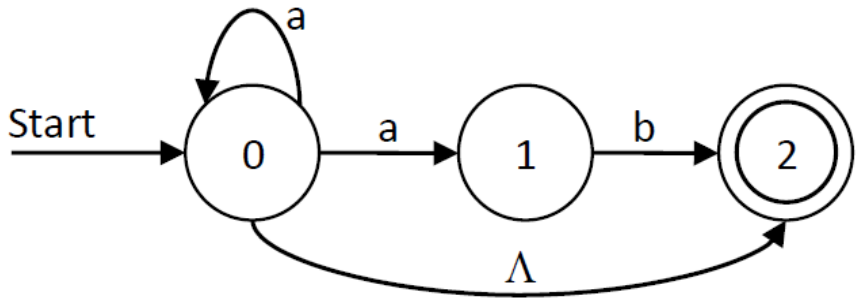
```
start(0).

t(0,a,1). t(0,b,2).
t(1,a,2). t(1,b,1).
t(2,a,2). t(2,b,2).

final(1).

accept(S):-start(I),path(I,S).

path(K,[]):-final(K).
path(K,[H|T]):-t(K,H,N),path(N,T).
```




```
accept([a,b,b]).
```

Nedeterministički konačni automat

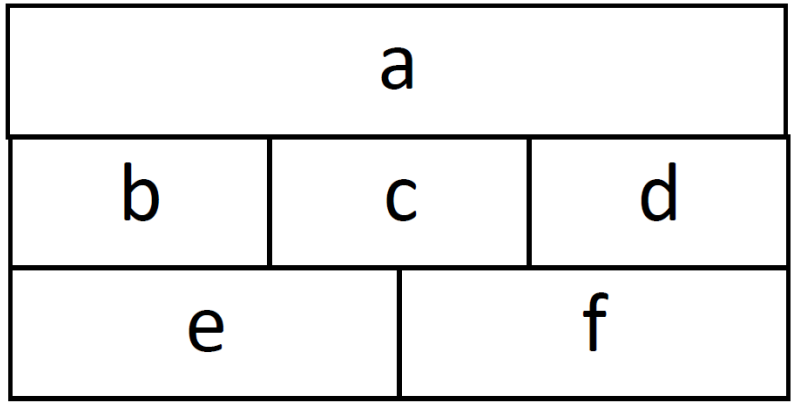
```
start(0).

t(0,a,0). t(0,a,1).
t(0,[],2). t(1,b,2).

final(2).

accept(S):-start(I),path(I,S).

path(K,[]):-final(K).
path(K,[H|T]):-t(K,H,N), path(N,T).
path(K,X):-t(K,[],N),path(N,X).
```




```
accept([a,b,b]).
```

Problem 4 boje

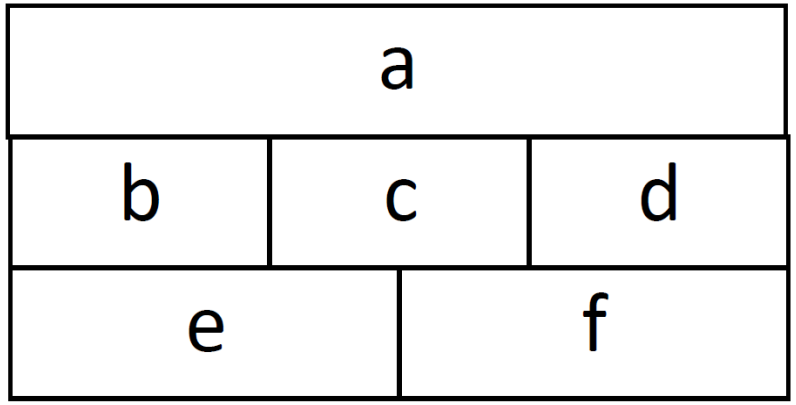
```
sused(zuta,zelena). sused(zuta,crvena).
sused(zuta,plava).

sused(zelena,zuta). sused(zelena,crvena).
sused(zelena,plava).

sused(crvena,zuta). sused(crvena,zelena).
sused(crvena,plava).

sused(plava,zuta). sused(plava,zelena).
sused(plava,crvena).

oboj(A,B,C,D,E,F):-sused(A,B), sused(A,C),
       sused(A,D), sused(B,C), sused(B,E),
       sused(C,D), sused(C,E), sused(C,F),
       sused(D,F), sused(E,F).
```




```
oboj(A,B,C,D,E,F).
```

Problem 4 boje

```
boji_mapu([Zem|Zem1],Boje):-
      boji_zemlju(Zem,Boje),
      boji_mapu(Zem1,Boje).
boji_mapu([],Boje).

boji_zemlju(zem(Ime,Boja,Susedi),Boje):-
      izvadi(Boja,Boje,Boje1),
      podskup(Susedi,Boje1).

izvadi(X,[X|Xs],Xs).
izvadi(X,[Y|Ys],[Y|Zs]):-izvadi(X,Ys,Zs).

podskup([X|Xs],Ys):-clan(X,Ys),
      podskup(Xs,Ys).
podskup([],_).

clan(X,[X|_]).
clan(X,[_|V]):-clan(X,V).
```




```
Zemlje = [zem(a,A,[B,C,D]), zem(b,B,[A,C,E]),
zem(c,C,[A,B,D,E,F]), zem(d,D,[A,C,F]),
zem(e,E,[B,C,F]), zem(e,F,[C,D,E])],
Boje = [bela,zuta,plava,zelena ],
boji_mapu(Zemlje,Boje).
```

Matrice

$$A = \begin{bmatrix} 3 & 5 & 12 \\ 4 & 9 & 18 \end{bmatrix}$$

Lista čiji elementi su liste

```
A = [[3, 5, 12], [4, 9, 18]]
```

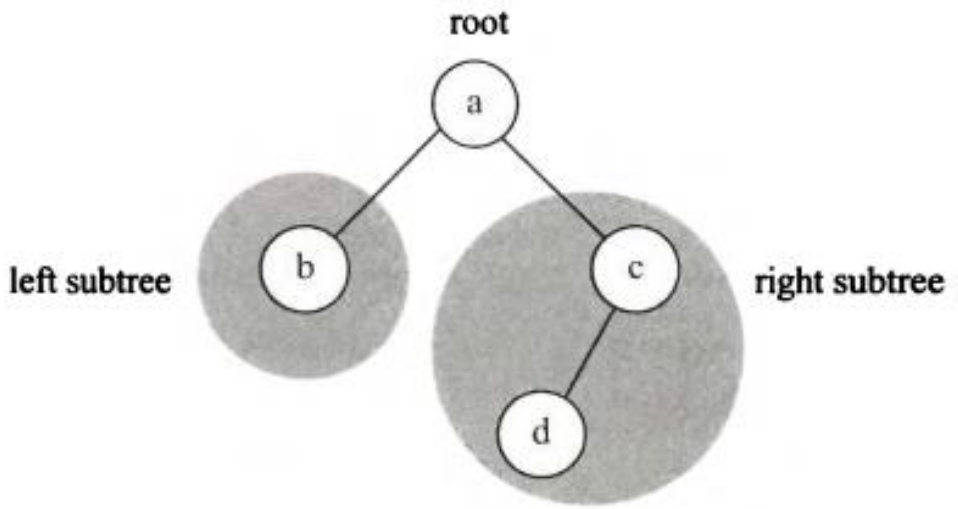
Predikat matrice

```
mat(a,1,1,3). mat(a,1,2,5). … mat(a,2,3,18).
```

Binarna stabla

Binarno stablo je ili prazno (**nil**) ili sadrži tri elementa

- Čvor
- Levo podstablo
- Desno podstablo




```
in(X,t(_,X,_)).
in(X,t(L,_,_)):-in(X,L).
in(X,t(_,_,D)):-in(X,R).
```

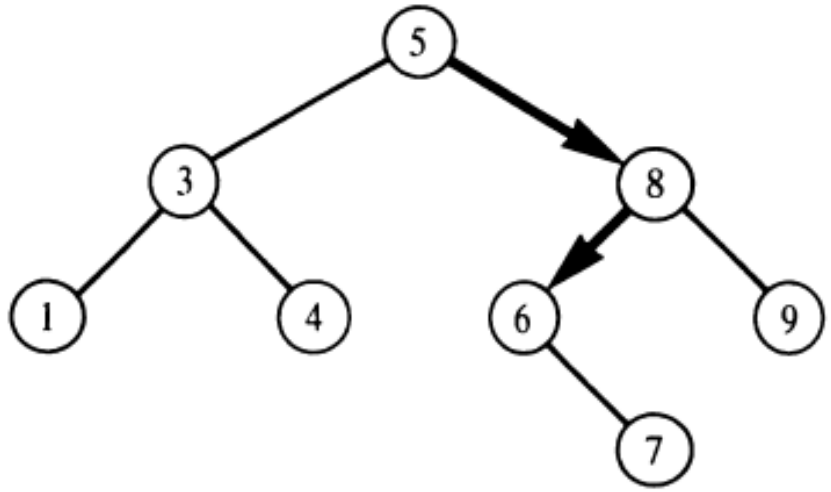
```
?- in(X,nil).
false.

?- in(b,t(t(nil,b,nil),a,t(t(nil,d,nil),c,nil))).
true .

?- in(X,t(t(nil,b,nil),a,t(t(nil,d,nil),c,nil))).
X = a ;
X = b ;
true ;
```

a(b,c(d))

t(t(nil,b,nil),a,t(t(nil,d,nil),c,nil))

Uređeno binarno stablo

```
in(X,t(_,X,_)).
in(X,t(Left,Root,Right)):-
                Root>X, in(X,Left).
in(X,t(Left,Root,Right)):-
                Root<X, in(X,Right).
```



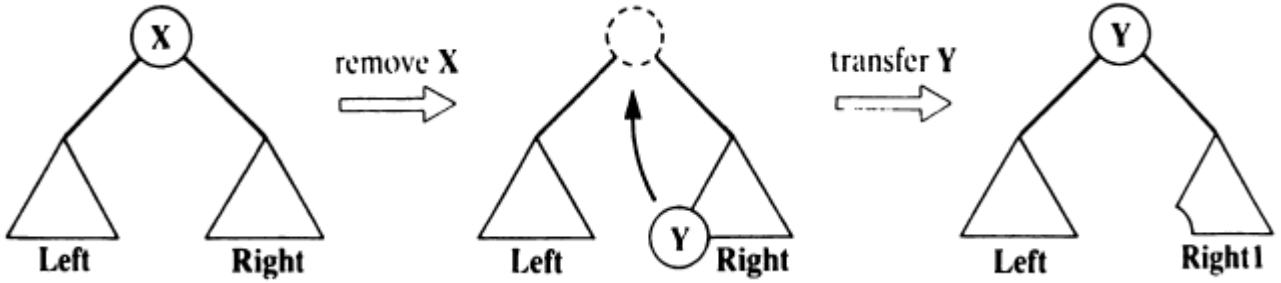

```
?- in(6,t(t(t(nil,1,nil),3,t(nil,4,nil)),5,t(t(nil,6,t(nil,7,nil)),8,t(nil,9,nil)))).
true .
```

Dodavanje čvora u binarno stablo

```
addleaf(nil,X,t(nil,X,nil)).
addleaf(t(Left,X,Right),X,t(Left,X,Right)).
addleaf(t(Left,Root,Right),X,t(Left1,Root,Right)):-
     Root>=X, addleaf(Left,X,Left1).
addleaf(t(Left,Root,Right),X,t(Left,Root,Right1)):-
     Root<X, addleaf(Right,X,Right1).
```

```
?- addleaf(nil,5,D),addleaf(D,3,D1),addleaf(D1,12,D2),
|    addleaf(D2,8,D3),addleaf(D3,15,T).
D = t(nil, 5, nil),
D1 = t(t(nil, 3, nil), 5, nil),
D2 = t(t(nil, 3, nil), 5, t(nil, 12, nil)),
D3 = t(t(nil, 3, nil), 5, t(t(nil, 8, nil), 12, nil)),
T = t(t(nil, 3, nil), 5, t(t(nil, 8, nil), 12, t(nil, 15, nil))) .
```

Brisanje čvora iz binarnog stabla



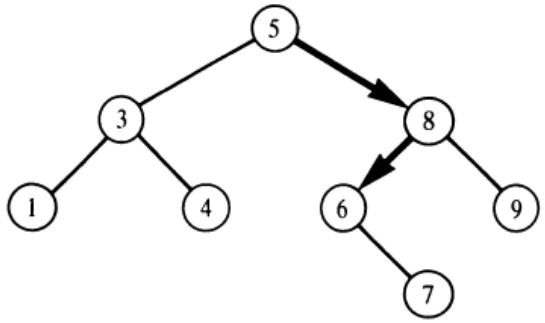

```
del(t(nil,X,Right),X,Right).
del(t(Left,X,nil),X,Left).
del(t(Left,X,Right),X,t(Left,Y,Right1)):-delmin(Right,Y,Right1).
del(t(Left,Root,Right),X,t(Left1,Root,Right)):-Root>=X, del(Left,X,Left1).
del(t(Left,Root,Right),X,t(Left,Root,Right1)):-Root<X, del(Right,X,Right1).
```

```
delmin(t(nil,Y,Right),Y,Right).
delmin(t(Left,Root,Right),Y,t(Left1,Root,Right)):-delmin(Left,Y,Left1).
```

```
?- del(t(t(t(nil,1,nil),3,t(nil,4,nil)),5,t(t(nil,6,t(nil,7,nil)),8,t(nil,9,nil))),5,D).
D = t(t(t(nil, 1, nil), 3, t(nil, 4, nil)), 6, t(t(nil, 7, nil), 8, t(nil, 9, nil))) .
```

Ispis stabla

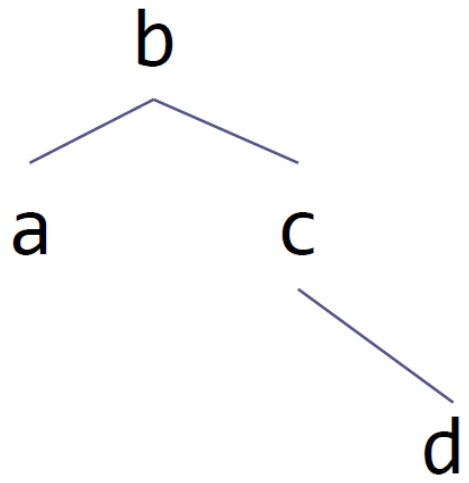
```
show(Tree):-show2(Tree,0).
show2(nil,_).
show2(t(Left,X,Right),Indent):-
      Ind2 is Indent + 2, show2(Right,Ind2),
      tab(Indent),write(X),nl, show2(Left,Ind2).
```



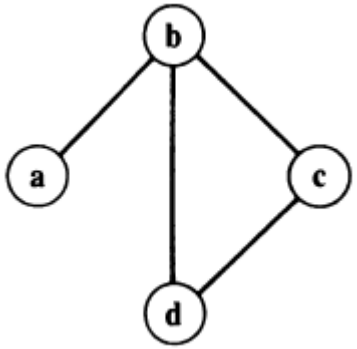

```
?- show(t(t(t(nil,1,nil),3,t(nil,4,nil)),5,t(t(nil,6,t(nil,7,nil)),8,t(nil,9,nil)))).
     9
   8
       7
     6
5
     4
   3
     1
true.
```

Stablo preko liste predikata

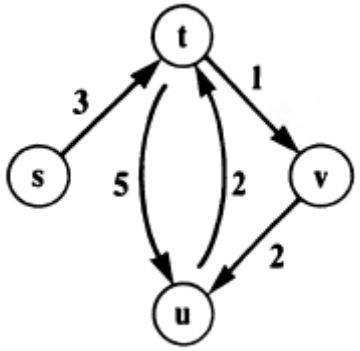
```
drvo(a,b,c).
drvo(nil,a,nil).
drvo(nil,c,d).
drvo(nil,d,nil).
```



Grafovi



G1 = graph([a,b,c,d],[e(a,b),e(b,d),e(b,c),e(c,d)])



G2 = digraph([s,t,u,v],[a(s,t,3),a(t,v,1),a(t,u,5),a(u,t,2),a(v,u,2)])

- Put
- Hamiltonov put
- Razapinjuće stablo

Ostale strukture

Balansirana stabla

AVL stabla

Polinomi – lista ili lista parova ili predikat polinoma

…

Algoritmi

Pretrage - BFS, DFS,…

Heuristike – Best-First, A*,

Ekspertni sistemi

Mašinsko učenje

Obrada prirodnih jezika

Igranje igara

…