13. Ievade un izvade

Nodaļas saturs

Nodaļā ievietotie attēli, programmu piemēri un tabulas

13.1. Ievades un izvades pamatprincipi

Standarta veids saziņai ar Prolog sistēmu ir interaktīvā vide, kurā var ievadīt vaicājumus un kurā tiek izvadīti vaicājumu rezultāti. Tomēr atsevišķos gadījumos ir nepieciešami citi līdzekļi datu apmaiņai ar sistēmu:

- ievadāmajiem/izvadāmajiem datiem ir cits formāts, nekā piedāvā Prolog sistēma,
- ievade/izvade jāveic no faila.

Valoda Prolog piedāvā divus galvenos veidus ievadei un izvadei:

- **pa simbolam** vai baitam (piemēram, predikāti *get/1*, *get0/1*, *put/1*),
- **pa termam** (atomam, sarakstam utt.) (piemēram, predikāti *read/1*, *write/1*).

No plūsmu viedokļa, ievades/izvades avots/mērķis ir vai nu fails vai nu lietotājs (*user*), t.i., standarta ievade/izvade.

Pēc noklusēšanas ievade/izvade notiek lietotāja (*user*) režīmā.

13.2. Ievade un izvade pa termam

Ievadi un izvadi pa termam veic predikāti *read/1* un *write/1*. Bez tam izvadei tiek lietoti arī predikāti *tab/1* un *nl/0*.

Tab. 13-1. Prolog ievade un izvades predikāti pa termam.

Nosaukums	Komentārs
`read/1`	Nolasa vienu termu.
`write/1`	Izdrukā vienu termu.
`nl/0`	Izdruka *newline* simbolu.
`tab/1`	Izdrukā tukšumus (*space*) skaitā N.

Daži piemēri bez predikātu izmantošanas (lietotāja ievads dots **trekināti**):

```
?- X=[X1,X2], read(X1), read(X2).
alpha.
beta.

X = [alpha,beta]
X1 = alpha
X2 = beta

yes
```

Ievērojiet, ka ievadot pēc terma jāliek punkts!

Tālāk seko izvade:

```
X=alpha, Y=beta, write(Y), nl, write(X), nl, write(end).
beta
alpha
end

X = alpha
Y = beta

yes
```

Tukšumu izmantošana:

```
?- tab(1), write(1), tab(1), write(viens), nl,
write(12), tab(1), write(divpadsmit).
 1 viens
12 divpadsmit

yes
```

Prologam var uzdot arī lielāku termu izdrukāšanu – ja vien apmierina Prolog piedāvātais formāts:

```
?- A=[a,b,c,d], write(A).
[a,b,c,d]

A = [a,b,c,d]

yes
```

13.3. Speciālu predikātu veidošana ievadei un izvadei

šeit ir parādīti divi piemēri – predikāts saraksta izvadei un predikāts saraksta ievadei.

Predikāts saraksta elementu izvadei *printlist/1*, katru liekot savā rindā:

```
printlist([]).
printlist([A|X]):-write(A),nl,printlist(X).
```

Predikāts darbībā:

```
?- printlist([a,b,c,d]).
a
b
c
d

yes
```

Saraksta nolasīšanas predikāts *readlist/1*:

```
readlist(L):-read(X), (X=end-> L=[]; readlist(M), L=[X|M]).
```

Predikāts darbībā (lietotāja ievads **trekināti**):

```
?- readlist(L).
alpha.
beta.
gamma.
delta.
end.

L = [alpha,beta,gamma,delta]

yes
```

13.4. Ievade un izvade failā

Ievade un izvade failā notiek tieši tāpat, kā lietotāja režīmā, vienīgi pirms tam ir jānovirza plūsma faila virzienā. Nākošajā tabulā doti predikāti un vērtības, kas ir specifiski faila apstrādei.

Tab. 13-2. Prolog predikāti un vērtības failu apstrādei.

Nosaukums	Komentārs
`see/1`	Atver failu **lasīšanai** (faila padošanas princips līdzīgi predikātam *consult/1*). No šī brīža *read* un citas nolasīšanas komandas strādās ar failu.
`seen/0`	Faila lasīšanas beigas. Pārslēgšanās uz lietotāja režīmu.
`see(user)`	Pārslēgšanās uz lietotāja režīmu lasīšanā, neaizverot failu.
`tell/1`	Atver failu **rakstīšanai**. No šī brīža *write* un citas rakstīšanas komandas strādās ar failu.
`told/0`	Faila rakstīšanas beigas. Pārslēgšanās uz lietotāja režīmu.
`tell(user)`	Pārslēgšanās uz lietotāja režīmu rakstīšanā, neaizverot failu.
`end_of_file`	Atoms, kas norāda, ka nolasītas faila beigas.

13.4.1. Piemērs 1

Dots fails “input”:

```
a. b. c. d.
```

Nākošais vaicājums uz ekrāna parāda (ievads no klaviatūras treknināti):

```
?- see(input),read(X),seen,read(Y),tell(output),write(X),
                  told,write(Y).
abrakadabra.
abrakadabra

X = a
Y = abrakadabra

yes
```

Rezultātā iegūtais fails "output" (sastāv no viena simbola):

```
a
```

13.4.2. Piemērs 2

Dots tas pats fails "input". Sekojošais piemērs parāda pārslēgšanos starp lietotāja un faila režīmu lasīšanā (lietotāja ievads **treknināti**):

```
see(input),read(X),see(user),read(Y),see(input),read(Z),seen.
myinput.

X = a
Y = myinput
Z = b

yes
```

13.4.3. Piemērs 3

Faila nolasīšanas predikāts, kas ielasa termus sarakstā *readlist/2*:

```
readlistfromfile(L):-read(X), (X=end_of_file-> L=[];
                  readlistfromfile(M), L=[X|M]).
readlist(F,L):-see(F), readlistfromfile(L), seen.
```

Predikāta izmantošana faila "input" nolasīšanai:

```
?- readlist(input,A).

A = [a,b,c,d]

yes
```

13.5. Failu apstrāde pa simbolam

Nākošajā tabulā parādīti predikāti failu apstrādē pa simbolam. Tie der arī informācijas ievadei/izvadei lietotāja režīmā.

Tab. 13-3. Prolog predikāti plūsmu apstrādei pa simbolam.

Nosaukums	Komentārs
`get/1`	Nolasa vienu simbolu, izlaižot tukšumus. Atgriež simbola kodu.
`get0/1`	Nolasa vienu simbolu, ieskaitot tukšumus. Atgriež simbola kodu.
`put/1`	Izdrukā vienu simbolu pēc tā koda.

Nākošais piemērs parāda predikāta *get/1* izmantošanu faila "input" nolasīšanā. Trešā nolasītā simbola kods 98 atbilst 'b', tātad tukšums ir izlaists:

```
?- see(input),get(X),get(Y),get(Z),seen.

X = 97
Y = 46
Z = 98

yes
```

Predikāts *get0/1* neizlaiž tukšumus, tāpēc trešais nolasītais simbols ir tukšums (kods 32):

```
?- see(input),get0(X),get0(Y),get0(Z),seen.

X = 97
Y = 46
Z = 32

yes
```

Predikāta *put/1* izmantošana izdrukai uz ekrāna (simboli 'a' un '.'):

```
?- put(97),put(46),told.
a.

yes
```

13.6. Vingrinājumi

Vingrinājums #1.

Definēt predikātu *printlist/2* sarakstu izvadei, kur otrais arguments norāda, pa cik elementiem vienā rindā drukāt (starp elementiem rindā drukāt tukšumu):

```
?- printlist([a,b,c,d,e],2).
a b
c d
e

no
?- printlist([a,b,c,d,e],3).
a b c
d e

no
```

Predikāta izveidošanai būs nepieciešams papildus predikāts, kurā būs papildus arguments izdrukāto rindas elementu uzskaitīšanai.

Vingrinājums #2.

Definēt predikātu *printlist2d/1*, kas izdrukā sarakstu no sarakstiem, katru apakšsarakstu liekot jaunā rindā:

```
?- printlist2d([[a,b,c],[2,3],[w,x,y,z]]).
a b c
2 3
w x y z

yes
```