Ministerul Educaţiei al Republicii Moldova
Universitatea Tehnică a Moldovei
Facultatea Calculatoare Informatică şi Microelectronică
Catedra Automatică şi Tehnologii Informaţionale

RAPORT

Lucrare de laborator nr 3

Disciplina: Programarea Funcțională și Inteligența Artificială

Tema: Structuri de date în prolog

A efectuat:

Vovc Artemie st. TI-133

A verificat:

Lazu Victoria lect. super.

Chișinău 2016

Cuprins

1 Scopul lucrării

Folosirea listelor, un instrument puternic al Prologului.

1.2 Sarcina

Scrieţi un program care utilizează predicatul listaPara, ce conţine două argumente: o listă de numere întregi, iar al doilea argument returnează o listă cu toate numerele pare din prima listă.

2 Noțiuni teoretice

Lista reprezintă unul dintre tipurile cele mai utilizate de structurile de date atât în Prolog, cât şi în alte limbaje declarative. O listă este o secvenţă ordonată de obiecte de acelaşi tip. În Prolog, elementele unei liste se separă între ele prin virgulă şi întreaga secvenţă este închisă între paranteze drepte.

Exemple:

[] – listă vidă;

[X, Y, Y] – listă ale cărei elemente sunt variabilele **X, Y şi Z**;

[[0, 2, 4], [1, 3]] – listă de liste de numere întregi.

Tipurile de liste utilizate într-un program Prolog trebuie declarate în secţiunea domains sub forma: tip_lista = tip* unde tip este un tip standard sau definit de utilizator. O listă este compusă conceptul din două părţi:

• ***cap*** (head), care desemnează primul element din listă;

• ***rest*** (tail), care desemnează lista elementelor rămase după eliminarea primului element.

Restul unei liste se mai numeşte corpul sau coada unei liste şi este întotdeauna o listă. Exemplele următoare ilustrează modul în care se structurează o listă (tabelul 3.1).

Tabelul 3.1. Exemple de structurare a listelor

Lista	Cap	Coada
['a','b','c']	'a'	['b','c']
['a']	'a'	[]
[]	nedefinit	nedefinit
[[1,2,3],[2,3,4],[	[1,2,3]	[[2,3,4],[]

Această dihotomie este utilizată în predicatele care prelucrează liste folosindu-se de avantajul regulii de unificare (identificare, substituire):

• singurul termen care se identifică cu [] este [].

• o listă de forma [H1|T1] se va identifica numai cu o listă de

forma [H2|T2] dacă H1 se poate identifica cu H2 şi T1 se poate identifica cu T2.

În tabelul 3.2 sunt date câteva exemple care ilustrează această regulă de unificare.

Tabelul 3.2. Exemple ce ilustrează regula de unificare

Lista1	Lista2	Legarea variabilelor
[X,Y,Z]	[Ion, Maria, Vasile]	X=Ion, Y=Maria, Z=Vasile
[7]	[X,Y]	X=7, Y=[]
[1,2,3,4]	[X,Y|Z]	X=1, Y=2, Z=[3,4]
[1,2]	[3|X]	esec

3 Realizarea

```
inc(X,Y):- Y is X+1.
getElementById([Head|Tail],ValueIndex,IncFrom,Result):-
inc(IncFrom,LocalI), LocalI=\=ValueIndex-
>getElementById(Tail,ValueIndex,LocalI,Result);(Result=Head,!).
length_1(0,[]).
length_1(L+1, [_|T]) :- length_1(L,T).
ciclu([H|T],ArrSize,StartInc):-
     inc(StartInc,LocalI),
     getElementById([H|T],LocalI,0,Result),
     LocalI=<ArrSize->
     ((Pair is mod(Result,2),Pair==0->(write(Result),
         (LocalI<(ArrSize-1)->write(",");write("")))
      ;write("")),ciclu([H|T],ArrSize,LocalI));!.
result([H|T]):-write("["),length_1(X,[H|T]),ArrSize is
X,ciclu([H|T],ArrSize,0),write("]").
```

Rezultatul:

```
28 ?- result([1,2,3,4,5,6,7,8,9,10]).
[2,4,6,8,10]
true.

29 ?- result([1,2,3,4,5,6,7,8,9]).
[2,4,6,8]
true.

30 ?- result([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]).
[2,4,6,8,10,12,14,16,18,20]
true.

31 ?- result([0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20])
[0,2,4,6,8,10,12,14,16,18,20]
true.
```

Figura 3.1 – Rezultatul interogării

Regula inc incrementează o valoare de intrare X și înscrie rezultatul în Y. Regula GetElementById iterează în listă pînă la indexul introdus și cînd ajunge la el returnează valoarea în Result și iesă din recursie. Length returnează lungimea unei liste. Ciclu cilează sau procesează lista în care se analizează elementele și se aleg elementele acelea care satisfac condiția de paritate. Result – regula returnează lista cu elementele pare găsite

Concluzia

Lucrarea dată a avut ca scop să ne facă cunoscuți cu programarea logică la general și programarea în prolog esențial. În lucrarea dată am lucrat cu listele în prolog. Listele sunt o structură greu de procesat, dar în schimb e eficient pentru Prolog.

Bibliografia

1 **Prolog la general :** http://biblioteca.regielive.ro/referate/limbaje-de-programare/prolog-limbaj-de-programare-logica-114586.html

2 **Studierea SWI-Prolog :** http://www.swi-prolog.org/pldoc/index.html

Anexe A

Baza de cunoștințe

```
inc(X,Y):- Y is X+1.
getElementById([Head|Tail],ValueIndex,IncFrom,Result):-
inc(IncFrom,LocalI), LocalI=\=ValueIndex-
>getElementById(Tail,ValueIndex,LocalI,Result);(Result=Head,!).
length_1(0,[]).
length_1(L+1, [_|T]) :- length_1(L,T).
ciclu([H|T],ArrSize,StartInc):-
     inc(StartInc,LocalI),
     getElementById([H|T],LocalI,0,Result),
     LocalI=<ArrSize->
     ((Pair is mod(Result,2),Pair==0->(write(Result),
         (LocalI<(ArrSize-1)->write(",");write("")))
      ;write("")),ciclu([H|T],ArrSize,LocalI));!.
result([H|T]):-write("["),length_1(X,[H|T]),ArrSize is
X,ciclu([H|T],ArrSize,0),write("]").
```