Дәріс № 15

Операторлар, мәліметтер қорымен жұмыс

Операторларды анықтау:

:- op(Приоритет, Спецификатор, предикат атауы).

Приоритет – 1 мен 1200 аралығындағы сан. Приоритет көп болған сайын сан кіші. Спецификатор **x, y** және **f қолданады.**

f – біздің предикат, біз анықтаймыз.

x – f приоритетінен міндетті түрде көп предикат.

y – f приоритетінен міндетті түрде көп немесе тең предикат

Берілу әдістері:

үш типті инфиксті операторлар: **xfx xfy yfx**

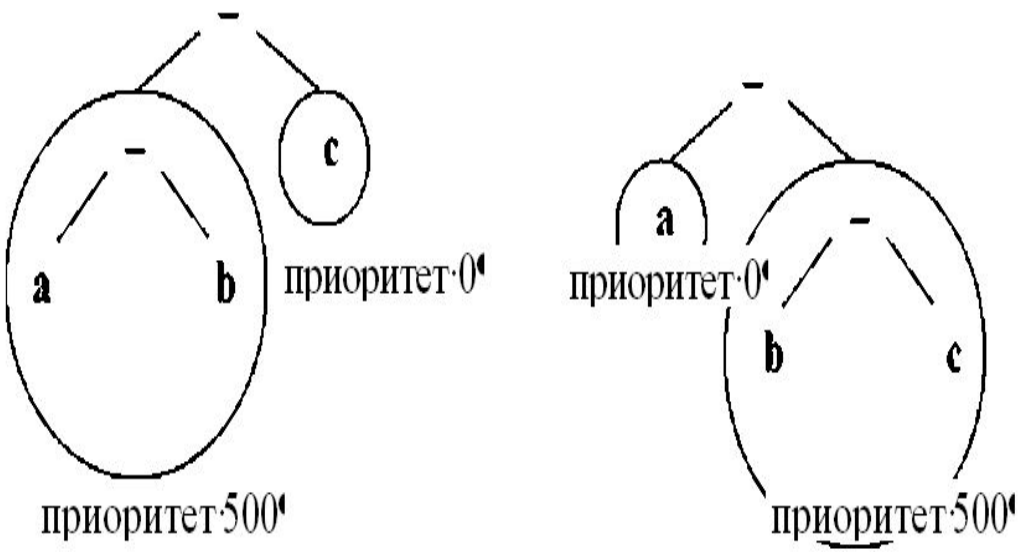
екі типті префиксті операторлар: **fx fy**

екі типті постфиксные операторлар: **xf yf**

Мысал:

:- op(500, yfx, -).

Анықтаудың мұндай әдісі айтады: предикат «минус» приоритет 500 бар және оның сол жағына оған тең предикат орналаса алады, ал оң жағынан тек одан кіші. Сонда, a – b – c жазбасы (a – b) – c ретінде интерпретацияланады. Сол жақтағы суретте көрсетілгендей.



Орындарымен ауыстырса:- op(500, xfy, -).

Онда оң жақтағы сурет дұрыс боладыда алу логикасы сіз мектепте үйренген математикамен үйлеспейді.

Прологта математикалық амалдар дұрыс орындалуы үшін амалдардың келесі приоритеттері қолданылады

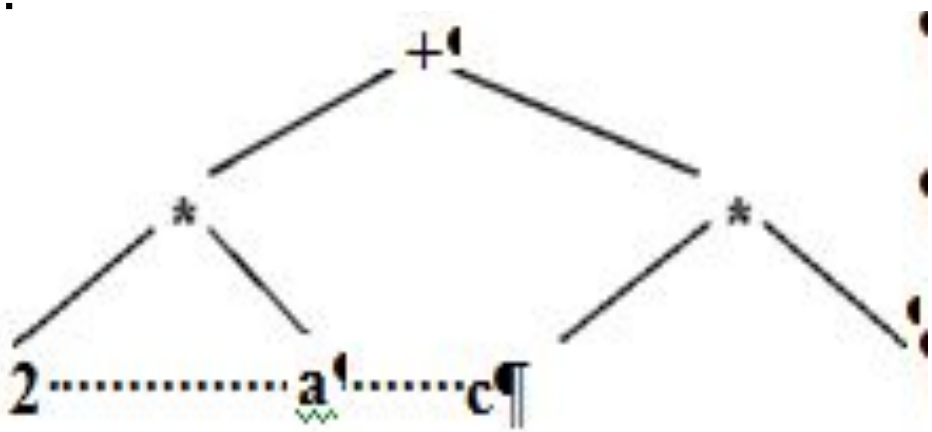
:- op(500, yfx, -).

:- op(500, yfx, +).

:- op(400, yfx, *).

- op(400, yfx, /).

Сонда 2*a+b*c өрнегі келесі суреттегідей интерпретацияланады:



Яғни алдымен приоритеті көп операциялар орындалады, соңынан приоритеті аз операциялар. Спецификаторлар операторлардың тізбектеле орындалу ретін көрсетеді.

Ендігі жерде біз бағдарламада анықтасақ:

:- op(1000, fx, not).

Онда мына сұранысты орындай аламыз:

?- X = 2, not X == 3.

X = 2

yes

not предикатының тік жақшаға алынбағанына назар аударыңыз**.**

Үй тапсырмасы: келесі белгілі ереже үшін барлық предикаттарды анықтау керек:

~ (A & B) <===> ~A v ~B

Ол эквивалентті түрде оқылуы керек (not(және(A, B)), немесе(not(A, not(B)))

assert(X) – X дәйегін бағдарламаға қосады.

retract(X) – X дәйегін бағдарламадан жояды.

Қосудың екі модификациясы бар **assertz** – бағдарламаның соңына қосу, **asserta** – бағдарламаның басына қосу.

Қолдану мысалы:

?- assertz(data(1)).

yes

?- data(X).

X = 1

yes

?– listing.

data(1).

yes

?– retract(data(_)).

yes

?– listing.

Yes

Динамикалық предикатқа дұрыс жүгіну үшін оны динамикалық ретінде анықтау керек. Ол үшін бағдарламада шақыру керек **dynamic(предикат_атауы/Арность_предиката).**

Фибоначчи сандарының мысалдары. Бағдарлама мәтіні:

:–dynamic(fibon/2).

fib(0, 1).

fib(l, 1).

fib(N, V) :– N1 is N–1, N2 is N–2, (fibon(N1, V1); fib(N1, V1)), (fibon(N2, V2); fib(N2, V2)), V is V1+V2, asserta(fibon(N,V)).

Шешімді осылайша орындауда рекурсивті шақырулар саны **fib** айтарлықтай кемиді. Неге **fib** орнына **fibon** мәліметтерді сақтау үшін? fibon қолданылады, себебі fib статикалық предикат ретінде анықталады, ал Пролог статикалық предикатқа өзгеріс енгізуге рұқсат етпейді. **Өзгерістерді тек қана динамикалық предикаттарға енгізуге болады!**

Предикат **abolish(предикат_атауы_Арность_предиката)** осы атаудағы және осы арндағы предикаттардың барлық кірулерді жояды.

retract арқылы осыған ұқсас предикаттың жоюлуын жазыңыз:

retractAll(X).

retractAll(X) :– retract(X), retractAll(X).

retractAll(_).

X тің бағдарламаға барлық кірулерін жояды.

Тапсырма: Прологтағы статикалық айнымалыларды assert және retrtact қолдану арқылы анықтау.

Жұмыс тәртібі:

init(айнымалы атауы, мәні) – берілген мәні бар статикалық айнымалыны инициализациялау.

set(ИмяПеременной, Значение) – айнымалыға мәнді беру.

get(ИмяПеременной, Значение) – айнымалыдан мәнді алу.

Шешім:

init(Var, Val) :– assertz(variables(Var, Val)).

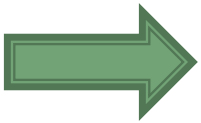
set(Var, Val) :– retract(variables(Var, _)),

Санның квадратын есептеу бағдарламасы:

```
square :– repeat, nl, write('Enter X = '), read(X), (X = end, !; Y
is X*X, write('X*X = '), write(Y), fail).
```

Пайдаланушы санды енгізеді – олардың квадратын есептейді. Бұл жағдай пайдаланушы end енгізбейінше орындала береді сол кезде бағдарлама дұрыс аяқталады немесе санның орнына басқа нәрсе енгізгенде exception орындалады.

```
?– square.
Enter X = 23.
X*X = 529
Enter X = 45.
X*X = 2025
Enter X = end.
yes
```



Дұрыс емес енгізуден қорғау үшін санды енгізгенде тексеру керек:

```
square :– repeat, nl, write('Enter X = '), read(X), (X = end, !;
number(X), Y is X*X, write('X*X = '),write(Y), fail).
```

Сонда аламыз:

```
?– square.
Enter X = er.
Enter X = 345.
X*X = 119025
Enter X = end.
Yes
```

Мұнда: **number(X)** – қосымша предикат, X сан болып табылатынын тексереді. Пролог тілінде оны қалай жазуға болады. (fail мен repeat орындаған секілді)?

Дұрыс емес енгізуден қорғау үшін санды енгізгенде тексеру керек:

```
square :– repeat, nl, write('Enter X = '), read(X), (X = end, !;
number(X), Y is X*X, write('X*X = '),write(Y), fail).
```

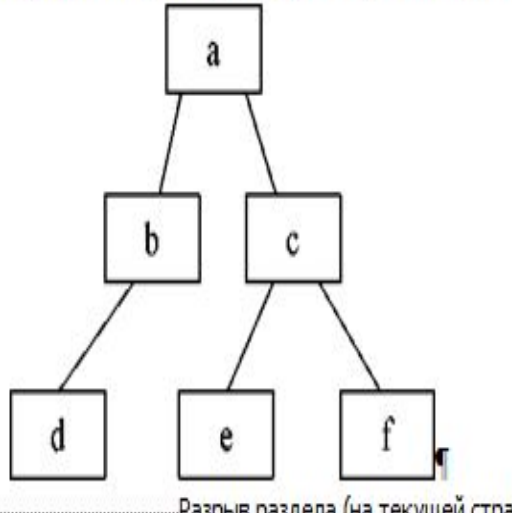
Сонда аламыз:

```
?– square.
Enter X = er.
Enter X = 345.
X*X = 119025
Enter X = end.
Yes
```

Мұнда: **number(X)** – қосымша предикат, X сан болып табылатынын тексереді. Пролог тілінде оны қалай жазуға болады. (fail мен repeat орындаған секілді)?

Бинарлы ағаштар.

Бинарлы ағаш – бұл әр түйінінде максимум екі бұтағы бар ағаш.



Прологта бинарлы ағаштарды беру нұсқалары:

1 a(b(d),c(e,f)).

2 nil түсінігін енгіземіз – п бос ағаш. btree предикатымен ағашты белгілейміз, үш параметрлі: бірінші – түбірі, екінші – сол жақтағы ағаш, үшінші – оң жақтағы ағаш.

btree(**a**, btree(**b**, btree(**d**, *nil*, *nil*), *nil*), btree(**c**, btree(**e**, *nil*, *nil*), btree(**f**, *nil*, *nil*)))

Бағдарламаны жазудың екінші нұсқасы бұдан күрделірек, бірақ бағдарламаны жазуға ыңғайлы.

Ағаштағы элементті іздеу бағдарламасын қарастырайық – member бағдарламасының аналогы, бірақ ағаштарға арналған.

Оны **in(Item, Tree)** деп атайық**.**

Бірінші параметр ретінде – ізделінуші элемент, екінші параметр – btree(...) ағаш түрі

```
in(Item, btree(Item, _, _)).
in(Item, btree(_, Left, _)) :– in(Item, Left).
in(Item, btree(_, _, Right)) :– in(Item, Right).
```

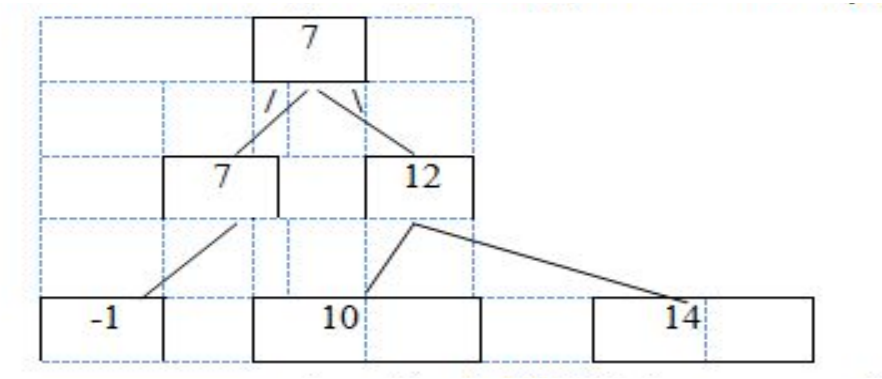
Бірінші ереже – индукция базасы – бойынша: ізделінуші элемент ағаштың түбірінде орналасқан.

Екінші ереже бойынша: ізделінуші элемент ағаштың сол жақтағы бұтағында орналасқан. Үшінші ереже бойынша : ізделінуші элемент ағаштың оң жақтағы бұтағында орналасқан.

Сұраныс үшін трассировкасын жазыңыз:

?– in(b, btree(a, btree(b, nil, nil)), nil).

Келтірілген бағдарлама тереңнен іздеуді орындайды. Бізде жай бинарлы ағаш емес, бинарлы сөздік деп ойлаңыз: ағаштың түйіндерінде сандар орналасқан, әрі сол жақ ағаштағы сандар түбірдегілердегі сандарға тең немесе олардан кем, ал оң жақ түйіндеріндегі сандар міндетті түрде үлкен. Ағашқа элементтің кіруін іздеу бағдарламасын жетілдіруіміз керек.



Айта кету керек, индексациялау кезінде Oracle МББЖ сі дәл осындай ағаштарды тұрғызады.

1 inS(Item, btree(Item, _, _)).

2 inS(Item, btree(Root, Left, _)) :– Item =< Root, in(Item, Left).

3 inS(Item, btree(Root, _, Right)) :– Item > Root, in(Item, Right).

Бинарлы ағашты мына тізім түрінде көрсетуге болады: **btree2list(Tree, List)**.

1 btree2list(Tree, List):– btree2list(Tree, [], List).

2 btree2list(nil, List, List).

3 btree2list(btree(Root, Left, Right), List, [Root | RList]) :– btree2list(Left, List, List1), btree2list(Right, List1, RList).

Бірінші ереже үш параметрлі предикатты шақыруды бағыттайды, оның параметрлерінің бірі бос тізім болып табылады. Екіншісі – индукция базасы. Үшіншісі – алдымен бір тізімнен нәтижені жинақтайды, одан кейін келесіден, соңынан оларға ағаш түбірін қосады.

Трассировка режіміндегі Прологтағы жауап пен сұранысты қарастырамыз:

?- btree2list(btree(a, nil, nil), R).

1	1	Call :	btree2list(btree(a, nil, nil), _20) ?
2	2	Call :	btree2list(btree(a, nil, nil), [], _20) ?
3	3	Call :	btree2list(nil, [], _117) ?
3	3	Exit:	btree2list(nil, [], []) ?
4	3	Call :	btree2list(nil, [], _77) ?
4	3	Exit:	btree2list(nil, [], []) ?
2	2	Exit:	btree2list(btree(a, nil, nil), [], [a]) ?
1	1	Exit:	btree2list(btree(a, nil, nil), [a]) ?

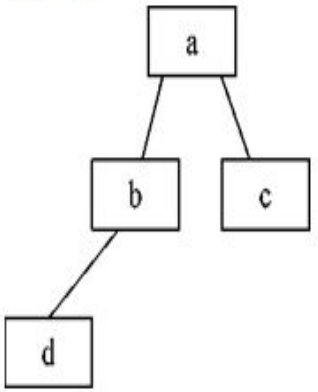
R=[a]

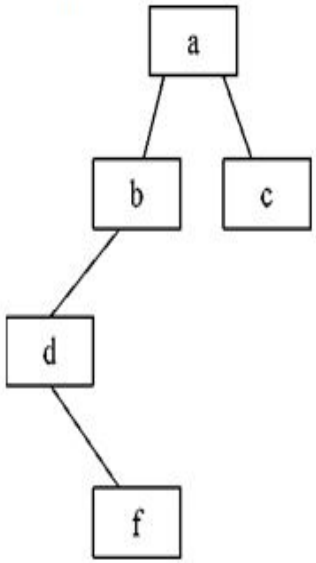
yes

Әрбір сұрақ белгісінің соңынан «Enter» бағдарламасы басылады.

Үй тапсырмасы: теңдестірілген бинарлы ағашты, яғни ағаштардың тереңдігінің айырмашылығы 1 ден көп емес бағдарламаны жазу керек.

Мысал:





Сол жақ ағаш теңдестірілген, ал оң жақтағы ағаш теңдестірілмеген, яғни «c» бұтағы мен «f» бұтағының айырмасы 2 тең.