Функциональное и логическое программирование

Лекция 9

2.15 Динамические базы данных

Программа – реляционная база данных. В процессе работы может возникнуть необходимость изменить, удалить, добавить предложения. Такие предложения – часть динамической базы данных.

Директива
:-dynamic <имя предиката>/<арность>.

Если динамическими являются несколько предикатов, то они перечисляются через запятую.

2.15.1 Добавление и удаление предложений

asserta(<предложение>)
Добавление в начало базы данных.

assertz(<предложение>)
Добавление в конец базы данных.

assert(<предложение>)
Добавление в конец базы данных.

Предикаты, добавляемые с помощью asserta и assertz, становятся динамическими по умолчанию.

Пример 1: Пусть имеется дерево семейных отношений, описанное программой в лекции 4:

```
родитель(пам,боб).
родитель(том,боб).
родитель(том,лиз).
родитель(боб,энн).
родитель(боб,пат).
родитель(пат,джим).
мужчина(том).
мужчина(боб).
мужчина(джим).

?- asserta(родитель(джим,кэт)).
true.

?- родитель(X,кэт).
X = джим.
```

retract(F)
Удаление из динамической базы данных первого предложения, сопоставимого с F. Предикат всегда успешен.

retractall(F)
Удаление из динамической базы данных всех предложений, сопоставимых с F. Предикат всегда успешен.

Продолжим работу после добавления в динамическую базу данных факта родитель(джим,кэт):

```
?- retract(родитель(джим,кэт)).
true.

?- родитель(X,кэт).
false.
```

В динамическую базу можно добавлять правила:

```
?-  asserta((дед(X,Y):-мужчина(X),родитель(X,Z),родитель(Z,Y))).
true.

?- дед(X,джим).
X = боб ;
false.

?- retract((дед(X,Y):-мужчина(X),родитель(X,Z),родитель(Z,Y))).
true.

?- дед(X,джим).
false.
```

Предложения, добавленные к программе приведенным выше способом, ведут себя точно так же, как и те, что были в “оригинале” программы.

Пример 2:

```
?- dynamic crisis/0.
 true.

?- crisis.
 false.

 ? - assert( crisis).
 true.

 ?- crisis.
 true.

 ? - retract( crisis).
 true.

 ?- crisis.
 false.
```

Пример 3: Программа о погоде.

```
:-dynamic солнце/0,дождь/0,туман/0,man/2.

хорошая:- солнце, not(дождь).
необычная:- солнце, дождь.
плохая:- дождь, туман.
дождь.
туман.
```

Далее приведен диалог с программой, во время которого база данных постепенно меняется.

```
?- хорошая.
false.
?- плохая.
true.
?- retract( туман).
true.
?- плохая.
false.
?- assert( солнце).
true.
?- необычная.
true.
?- retract( дождь).
true.
?- хорошая.
true.
```

listing(<имя предиката>/<арность>)

Вывод всех предложений базы данных, относящихся к определенному предикату, в текущий выходной поток.

```
?- listing(хорошая/0).
хорошая :-
    солнце,
    not(дождь).
```

Предикат listing без аргументов выводит все содержимое базы данных. В том числе, там хранится и код программы (в конце базы данных).

Пример 4:

Предикат, выводящий на экран и удаляющий из базы данных фамилии людей, которые старше 50 лет.

```
man(a,34).
man(b,60).
man(c,51).
man(d,25).
man(e,19).
man(f,52).
delete_man:-man(Family,Age),Age>50,writeln(Family),
                    retract(man(Family,Age)),fail.
delete_man.
```

```
?- delete_man.
b
c
f
?- listing(man/2).
:- dynamic man/2.
man(a, 34).
man(d, 25).
man(e, 19).
```

В динамической базе данных остались только факты, касающиеся людей с возрастом до 50 лет.

```
?- retractall(man(_,_)).

?- listing(man/2).
:- dynamic man/2.
```

В динамической базе данных не осталось фактов.

2.15.2 Заполнение динамической базы данных из файла, сохранение в файл

consult(‘<имя файла>’)

Считывает из файла предложения и добавляет их в конец динамической базы данных (аналогично assert).

Сохранение из динамической базы данных в файл:

tell+listing.

Пример 5:

Формирование динамической базы данных «Читатель библиотеки» с клавиатуры и сохранение ее в файле reader.txt.

```
goal:-repeat,
        writeln('Будете вводить новые сведения о читателях? y/n'),
        read(Answer),ответ(Answer),!,
         tell('C:\\reader.txt'),listing(читатель/2),
         retractall(читатель(_,_)),told.
ответ(n).
ответ(y):-запись_в_базу_данных,fail.
ответ(_):-fail.
запись_в_базу_данных:-write('Фамилия? '),read(Family),
           write('Месяц последнего посещения библиотеки? '),
           read(Mounth),
          write('Число месяца последнего посещения библиотеки? '),
          read(Data),
     asserta(читатель(Family,последнее_посещение(Data,Mounth))).
```

Фрагмент ввода данных:

?- goal.
Будете вводить новые сведения о читателях? y/n
|: d.
Будете вводить новые сведения о читателях? y/n
|: y.
Фамилия? |: a.
Месяц последнего посещения библиотеки? |: 5.
Число месяца последнего посещения библиотеки? |: 23.
Будете вводить новые сведения о читателях? y/n
|: y.
Фамилия? |: b.
Месяц последнего посещения библиотеки? |: 4.
Число месяца последнего посещения библиотеки? |: 1.
Будете вводить новые сведения о читателях? y/n
……
Будете вводить новые сведения о читателях? y/n
|: n.

Содержимое файла reader.txt после работы программы:

```
:- dynamic читатель/2.

читатель(f, последнее_посещение(25, 4)).
читатель(e, последнее_посещение(30, 5)).
читатель(d, последнее_посещение(2, 5)).
читатель(c, последнее_посещение(10, 5)).
читатель(b, последнее_посещение(4, 1)).
читатель(a, последнее_посещение(23, 5)).
```

Пример 6:

Определение количества читателей, посетивших библиотеку в мае, по информации, находящейся в файле reader.txt, сформированном в примере 5.

```
:-dynamic счетчик/1.
goal:-consult('reader.txt’),
   asserta(счетчик(0)),
   счет,счетчик(N),
   format('Кол-во читателей, посетивших библиотеку в мае ~w',[N]),
   retractall(счетчик(_)).
счет:-читатель(_,последнее_посещение(_,5)),
        счетчик(N),N1 is N+1,
        retract(счетчик(N)),
        asserta(счетчик(N1)),
        fail.
счет.
?- goal.
Кол-во читателей, посетивших библиотеку в мае 4
```

2.16 Создание меню

Пример:

Напишем предикат menu, который создает окно с главным меню, состоящим из трех пунктов. Выбор пунктов главного меню происходит до тех пор, пока не будет выбран 3 пункт (выход).

```
menu:-repeat,
    writeln('1 - процесс 1'),
    writeln('2 - процесс 2'),
    writeln('3 - выход'),nl,
    writeln('Выберите пункт меню 1-3'),
    read(N),
    N<4,process(N),N=3,!.
process(3).
process(1):-writeln('Проработал процесс 1'),fail.
process(2):-writeln('Проработал процесс 2'),fail.
```

Пример работы программы:

```
?- menu.
1 - процесс 1
2 - процесс 2
3 - выход

Выберите пункт меню 1-3
|: 1.
Проработал процесс 1
1 - процесс 1
2 - процесс 2
3 - выход

Выберите пункт меню 1-3
|: 2.
Проработал процесс 2
1 - процесс 1
2 - процесс 2
3 - выход

Выберите пункт меню 1-3
|: 3.
```