2.10.4 Компоновка данных в список

1. bagof(X,P,L) - порождает список L всех объектов X, удовлетворяющих цели P.

Если bagof(X, P, L) не находит ни одного решения для P, то цель bagof не успешна.

Если один и тот же X найден многократно, то все его экземпляры будут занесены в L, что приведет к появлению в L повторяющихся элементов.

Пример 1:

Опишем предикат класс для разбиения букв из некоторого множества на гласные и согласные.

класс(т,согласная).
класс(а,гласная).
класс(б,согласная).
класс(м,согласная).
класс(и,гласная).

bagof(X,класс(X,согласная),L).
Формирует список из огласных.

2. setof(X,P,L) - работает аналогично предикату bagof. При этом, список L упорядочен и не содержит повторяющихся элементов. Упорядочение происходит по алфавиту или по отношению ‘@<‘.

Можно получить список пар вида Класс/Буква:

setof(X/Y,класс(Y,X),L).

3. findall(X,P,L) - аналогичен предикату bagof. Он тоже порождает список объектов, удовлетворяющих P. Отличие от bagof в том, что findall собирает в список все объекты X, не обращая внимание на возможно отличающиеся для них конкретизации тех переменных из P, которых нет в X.

Если не существует ни одного объекта X, удовлетворяющего P, то findall все равно успешен и возвращает L = [ ].

Пример 2:

Пусть имеются сведения о возрасте людей (предикат information). Требуется найти средний возраст людей старше 20 лет из этой группы.

```
information(a,25).
information(b,15).
information(c,40).
information(d,5).
information(e,10).
information(f,35).
goal:-findall(X,(information(_,X),X>20),L),
      s(L,S,N), Sr is S/N, writeln(Sr).
s([],0,0):-!.
s([H|T],S,N):-s(T,S1,N1),S is S1+H,N is N1+1.
str_list(S,S1):-string_chars(S,L), delete(L,' ',L1),
                    string_chars(S1,L1).
```

2.11 Строки

Строка – последовательность символов, заключенная в двойные кавычки.

Предикаты для работы со строками:

1.string_length(S, L)
Определяет длину строки S.

2. string_concat(S1,S2,S3)
Соединяет две строки S1 и S2 в третью S3.

3. sub_string(S,K,N,R,S1).
Выделяет в строке S подстроку S1, которая начинается с K-го элемента и содержит N символов. R – количество символов, стоящих в S после подстроки S1. Нумерация элементов строки начинается с 0.
Пример: sub_string("asdfgh dfg hhjk",3,4,_,S).

4. string_chars(S,L)
Преобразует строку S в список символов и наоборот.

5. name(S,L)

Преобразует строку S в список кодов символов и наоборот.

6. char_code(C,K)

Преобразует символ C в его код K и наоборот.

7. split_string(S,R,D,L)

Преобразует строку S в список подстрок L, используя R как разделитель, удаляя из начала и конца подстрок символы строки D.

Пример:

split_string("asdf, fghj. dfgk jkjlkkl!,   ;df"," ",",.! ;",L).

9. number_string(N,S)

Преобразует число N в строку S и наоборот.

10. atom_string(A,S)

Преобразует атом A в строку S и наоборот.

Пример 1:

Предикат, который преобразует строку S в строку S1, удаляя пробелы.

```
str_list(S,S1):-string_chars(S,L),
                delete(L,' ',L1),
                string_chars(S1,L1).
```

Пример 2:

Предикат, который считает кол-во вхождений символа C в строку S.

```
goal:-writeln(‘Введите строку’), read(S),
        writeln(‘Введите символ’), read(C),
        string_chars(S,L), count(L,C,N), writeln(N).
count([],_,0):-!.
count([C|T],C,N):-count(T,C,N1), N is N1+1,!.
count([_|T],C,N):-count(T,C,N).
```

2.12 Предикаты для работы с файлами

1. exists_file(<‘имя файла’>)

Завершается успешно, если файл с указанным именем существует (обратные слэши дублируются).

2. open(<'имя файла’>, <режим>,F)

Открытие файла для чтения, записи или добавления. Режимы:

- read (для чтения);
- write (для записи);
- append (для добавления).

F – файловая переменная

Предикаты для работы с файлами являются внелогическими.
Чтение и обработку данных следует выполнять отдельно!

3. set_input(F)

   set_output(F)

Перенаправление ввода из файла или вывода в файл.

4. close(F)

Закрытие файла.

5. see(<'имя файла‘>)

   tell(<'имя файла‘>)

Открытие и перенаправления ввода из файла или вывода в файл вместо 2 и 3.

При перенаправлении ввода на клавиатуру, а вывода на экран в качестве имени файла используют имя user.

6. seen

   told

Закрытие файлов, открытых с помощью see и tell.

7. seeing(F)

telling(F)

Связывает F с именем файла, являющегося текущим входным или выходным потоком.

8. at_end_of_stream

Успешно завершается, если найден конец файла.

9. read_line_to_codes(F,L)

Читает строку из входного потока F и преобразует ее в список кодов символов этой строки (без кода перевода строки).

10. read_stream_to_codes(F,L)

Читает содержимое из входного потока F (до конца файла) и преобразует его в список кодов символов (включая коды перевода строки 10).

Пример:

see('in.txt'), seeing(F), read_stream_to_codes(F,L), seen.

Пример 1:

Предикат, который выводит на экран строки из файла, начиная с некоторого номера.

Имя файла и номер строки вводятся с клавиатуры.

```
goal:-writeln(‘Введите имя файла’),read(Filename),
        check_exist(Filename),
        writeln(‘Введите номер строки’), read(N),
        open(Filename,read,F), set_input(F),
        read_file(F,N),
        format(‘Содержимое файла, начиная с ~w строки\n’,N),
        write_screen(F), close(F).
check_exist(Filename):-exists_file(Filename),!.
check_exist(_):-writeln(‘Такого файла нет’),fail.
read_file(_,_):-at_end_of_stream, !,
                        format('В файле меньше, чем ~w строк\n',N),  fail.
read_file(_,1):-!.
read_file(F,N):-read_line_to_codes(F,_), N1 is N-1,                    read_file(F,N1).
write_screen(_):-at_end_of_stream,!.
write_screen(F):- read_line_to_codes(F,L), string_to_list(S,L),
                          writeln(S), write_screen(F).
```

Пример 2:

Предикат, который записывает вводимые с клавиатуры строки в файл t.txt. Окончание ввода – строка “#” (Вводимые строки заключены в двойные кавычки).

```
write_to_file:-read(X), tell('t.txt’), write_s(X),
                told, write(‘Данные записаны в файл’).
write_s(“#”):-!.
write_s(X):-writeln(X), read(Y), write_s(Y).
```

2.13 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>.

2.13.1 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.

Пример:
asserta((отец(X,Y):-родитель(X,Y),мужчина(X))).

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

retract(F)
Удаления из динамической базы данных первого предложения, сопоставимого с F.

retractall(F)
Удаления из динамической базы данных всех предложений, сопоставимых с F

Пример 1: кризис

```
?- dynamic crisis/0.
 true.
?- crisis.
 false.
 ? - assert( crisis).
 true.
 ?- crisis.
 true.
 ? - retract( crisis).
 true.
 ?- crisis.
 false.
```

Пример 2: Программа о погоде.

:-dynamic солнце, дождь,туман.
хорошая:- солнце, not(дождь).
необычная:- солнце, дождь.
отвратительная:- дождь, туман.
дождь.
туман.

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

?- хорошая.
false.
?- отвратительная.
true.
?- retract(туман).
true.
?- отвратительная.
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.

Пример 3:

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

```
:-dynamic person/2.
person(a,25).
person(b,52).
person(c,54).
person(d,15).
person(e,35).
delete_person:-person(Family,Age), Age>50,
                writeln(Family),
                retract(person(Family,Age)), fail.
delete_person.
```

?- delete_person.
b
c
true.

?- listing(person/2).
:- dynamic person/2.

person(a, 25).
person(d, 15).
person(e, 35).

true.

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

?- delete_person.
b
c
true.

?- listing(person/2).
:- dynamic person/2.

person(a, 25).
person(d, 15).
person(e, 35).

true.
В динамической базе данных остались только факты, касающиеся сотрудников с возрастом до 50 лет.
?- retractall(person(_,_)).
true.

?- listing(person/2).
:- dynamic person/2.

true.

2.13.2 Заполнение динамической базы данных из файла, сохранение в файл

consult(‘<имя файла>’)
Считывает из файла предложения и добавляет их в конец динамической базы данных.

Сохранение из динамической базы данных в файл:
tell+listing

Пример 1:
Формирование динамической базы данных «Читатель библиотеки» с клавиатуры и сохранение ее в файле reader.txt.

```
:-dynamic reader/2.
goal:-repeat,
        writeln('будете вводить новые факты? y/n'),
        read(A), ответ(A),!,
        tell('reader.txt'), listing(reader/2), told.

ответ(n).
ответ(y):-запись,fail.

запись:-writeln('Фамилия читателя? '), read(Name),
            writeln('Дата последнего посещения библиотеки? '),
            read(Data),
           writeln('Месяц последнего посещения библиотеки? '),
           read(Mounth),
           assertz(reader(Name,дата_посещ(Data,Mounth))).
```

Пример 2:

Определение количества читателей, посетивших библиотеку в мае, по информации, находящейся в файле reader.txt, сформированном в примере 1.

```
:-dynamic счетчик/1.
goal:-consult('reader.txt'), retractall(счетчик(_)),
        asserta(счетчик(0)),
        счет, счетчик(N),writeln(N).
счет:-reader(_,дата_посещ(_,may)),
      счетчик(N), N1 is N+1, retract(счетчик(N)),
      asserta(счетчик(N1)), fail.
счет.
```

2.15 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 – процесс1'),
        writeln('2 – процесс2'),
        writeln('3 – выход'),nl,
        write('Введите Ваш выбор: (1-3) '),
        read(X), nl, X<4, process(X),nl, X=3,!.
process(3).
process(1):-writeln('Проработал процесс 1'),fail.
process(2):- writeln('Проработал процесс 2'),fail.
```