

Лабораторная работа №3
студента группы ИТ – 32
Курбатовой Софьи Андреевны

Выполнение: _____ Защита _____

УПРАВЛЕНИЕ ХОДОМ ВЫПОЛНЕНИЯ ПРОГРАММЫ

Цель работы: .

Содержание работы

1. Напишите при помощи предиката «сократить» составной запрос к базе данных «путешествие», который будет находить только один город, в который можно добраться из Белгорода, а затем будет отыскивать все города, в которые можно отправиться из найденного города на автобусе.

2. Напишите обратимую версию процедуры, вычисляющей площадь прямоугольника. Используйте предикаты `var(X)` и `nonvar(X)`.

3. Напишите с использованием предиката «`repeat`» составной запрос, который спрашивает у пользователя имена школьных товарищей и добавляет каждое имя в базу данных в виде факта: `школьный_товарищ(Имя)`.

После того, как пользователь введет слово «конец», запрос должен прекратить задавать вопросы и выдать на экран все только что введенные имена.

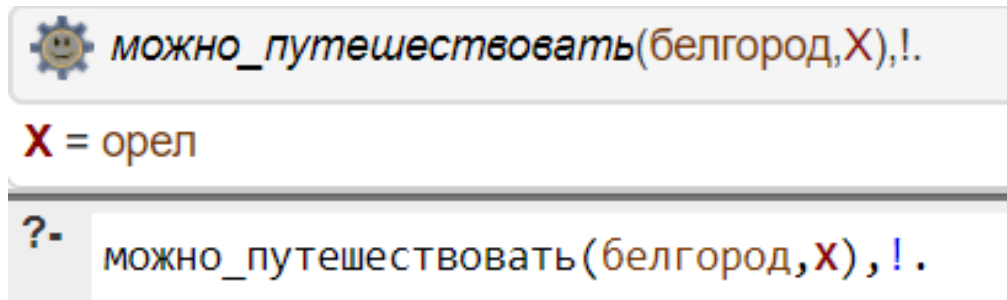
4. Выберите некоторую форму представления базы данных, в которой содержатся сведения об операциях с кредитными карточками. Каждая запись должна содержать сведения об имени лица, тратящего деньги, о типе операции и о сумме денег. Напишите процедуру, которая будет выдавать значение итоговой суммы всех операций для конкретного лица.

Ход работы

1. С применением ! стало возможным ограничить поиск по базе знаний с помощью указания вида интересующего нас транспорта. В данном случае это «автобус». Если в качестве первой цели указать можно_путешествовать(белгород, X), то будет возвращен первый известный город, из которого можно уехать, независимо от вида транспорта. Вторая цель позволяет продолжить поиск и указать, что уехать можно только в тот город, до которого есть маршрут на автобусе. То есть, так как из города Орёл можно уехать или в Брянск, или в Курск, программа вернет ответ: Курск.

```
1 путешествие_(белгород,орел,поезд).
2 путешествие_(орел,брянск,поезд).
3 путешествие_(орел,курск,автобус).
4 путешествие_(брянск,дятьково,автобус).
5 путешествие_(губкин,белгород,автобус).
6
7 можно_путешествовать(A,B):-
8     путешествие_(A,B,_),
9     A \= B.
10 можно_путешествовать(A,B):-
11     путешествие_(A,C,_),
12     путешествие_(C,B,_).
13 можно_путешествовать(A,B):-
14     путешествие_(C,B,_), можно_путешествовать(A,C).
```

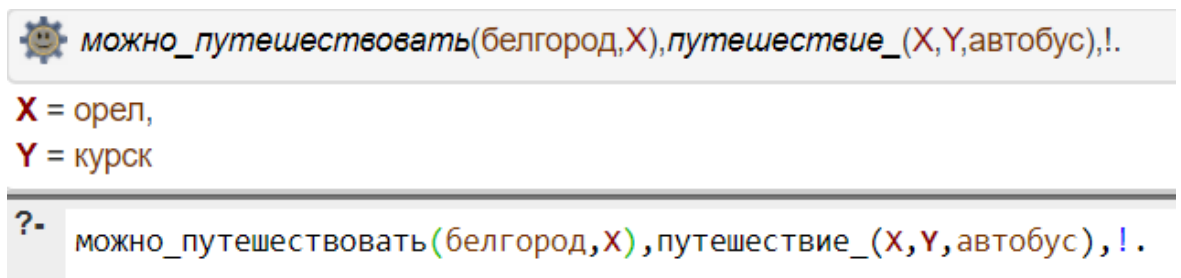
Рис. 3.1. База фактов



⚙️ можно_путешествовать(белгород,X),!.
X = орел

?- можно_путешествовать(белгород,X),!.
X = орел

Рис. 3.2. Только один город



⚙️ можно_путешествовать(белгород,X),путешествие_(X,Y,автобус),!.
X = орел,
Y = курск

?- можно_путешествовать(белгород,X),путешествие_(X,Y,автобус),!.
X = курск, Y = белгород

Рис. 3.3. Если есть ограничение на вид транспорта

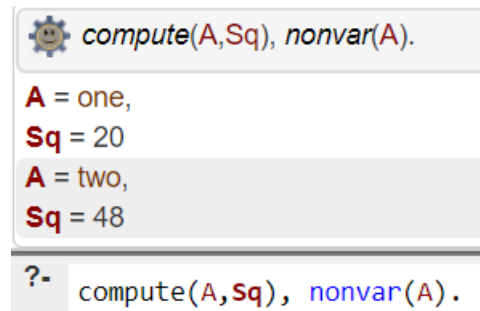
2. Предикат `var` и его отрицание `nonvar` проверяют, является ли терм свободной переменной. Так как терм `A` уже была конкретизирована в базе знаний для данной программы. Поэтому использование предиката `nonvar` вернул значение `площади` для всех известных в базе значений.

```

1 data(rectangle(one,4,5)).
2 data(rectangle(two,6,8)).
3 compute(A,Sq):-data(rectangle(A,B,C)),
4   Sq is (B*C).
5 compute(A,B,Sq):- Sq is (A*B).

```

Рис. 3.4. База знаний



```

?- compute(A,Sq), nonvar(A).

```

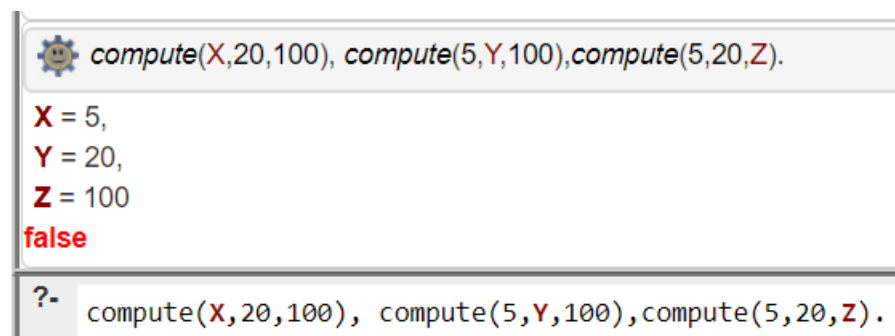
Рис. 3.5. Использование предиката `nonvar`

```

1 data(rectangle(one,4,5)).
2 data(rectangle(two,6,8)).
3 compute(A,Sq):-data(rectangle(A,B,C)),
4   Sq is (B*C).
5
6 compute(X,Y,Z):-integer(X),integer(Y),integer(Z),
7   S is X*Y, S=Z.
8 compute(X,Y,Z):-integer(X),integer(Y),var(Z),
9   Z is X*Y.
10 compute(X,Y,Z):-var(X),integer(Y),integer(Z),
11   X is Z/Y.
12 compute(X,Y,Z):-integer(X),var(Y),integer(Z),
13   Y is Z/X.

```

Рис. 3.6. Измененные правила в базе знаний



```

?- compute(X,20,100), compute(5,Y,100), compute(5,20,Z).

```

Рис. 3.7. Использование предиката `var`

3. Написала с использованием предиката «repeat» составной запрос, который спрашивает у пользователя имена школьных товарищей и добавляет каждое имя в базу данных в виде факта: школьный_товарищ(Имя). После того, как пользователь введет слово «конец», запрос прекращает задавать вопросы и выдает на экран все только что введенные имена.

```

1 :- dynamic школьный_товарищ/1.
2 cl :- школьный_товарищ(X), retract(школьный_товарищ(X)), fail.
3 in :- repeat, write("Введите имя"), nl, read(X), assert(школьный_товарищ(X)),
4      X = конец, !, retract(школьный_товарищ(X)), out.
5 out :- школьный_товарищ(Y), write(Y), nl, fail.

```

Рис. 3.8. Правила

Рис. 3.9. Заполнение базы знаний

4. Каждая запись должна описанной базы содержит сведения об имени лица, тратящего деньги, о типе операции и о сумме денег. Также написана процедура, которая будет выдавать значение итоговой суммы всех операций для конкретного лица.

```

1 :- dynamic cash/3.
2 :- dynamic s/2.
3 %инициализация суммы для конкретного лица X
4 init(X):-s(X,Y),retract(s(X,Y)), fail. % удаляем старое значение
5 init(X):-assert(s(X,0)). % добавляем сумму = 0
6
7 %добавление или вычет из общей суммы в зависимости от типа операции
8 sum(X,'in',C):- s(X,A), B is A + C, retract(s(X,A)), assert(s(X,B)),fail.
9 sum(X,'out',C):- s(X,A), B is A - C, retract(s(X,A)), assert(s(X,B)),fail.
10
11 % сама процедура, которая осуществляет подсчёт остатка на счету клиента
12 oper(X):-init(X), cash(X,Y,C), sum(X,Y,C).
13 oper(X):-s(X,S), write(S).

```

Рис. 3.10. Описание порядка работы с базой знаний

Рис. 3.11. Демонстрация работы программы

Вывод: В ходе выполнения лабораторной работы были реализованы механизмы позволяющие управлять программой написанной на языке Prolog. Так была создана заполняемая именами одноклассников база данных. Это стало возможным потому, что Prolog позволяет реализовывать циклы с помощью предиката repeat, реализующий вид цикла типа do...while.

Таким образом, используя данный язык программирования можно не только обращаться к заранее определенным базам данных, но и заполнять их по ходу выполнения программы, а также анализировать связи между данными этих баз.