Prolog

Реализация с использованием Turbo Prolog

1. Имена в Прологе (Prolog)

Пробелы в записи имени недопустимы, однако можно использовать **знак подчеркивания** (_) в качестве разделителя:

```
employee_name
color_of_box
```

Можно задавать имена, состоящие всего из одной буквы:

```
domains
  a,b = symbol
predicates
  s(a,b)
```

Имена переменных в Прологе начинаются с ПРОПИСНОЙ буквы.

2. Предикаты в Прологе (Prolog)

Программы на Прологе (Prolog) содержат объявления *логических взаимосвязей*, необходимых для решения задачи.

Рассмотрим предложение:

```
«Мэри любит яблоки».
```

То есть имеется **факт**, *утверждающий*, что Мэри любит яблоки. **Факты** и **правила** являются *утверждениями*, которые образуют данные программы на Прологе (Prolog).

Предикаты в Прологе (Prolog) определяют отношения между *объектами*, то есть используются для представления как *фактов*, так и *правил*.

Добавим еще одно предложение:

```
«Бет любит то же самое, что и Мэри».
```

Используя эти два предложения, можете придти к заключению, что *Бет тоже любит яблоки*.

Оба утверждения можно записать так:

```
«Мэри любит яблоки»
«Бет любит нечто, если Мэри любит (это же) нечто».
```

2. Предикаты в Прологе (Prolog)

Логика предикатов рассматривает отношения между *утверждениями* и *объектами*.

Отношение, соответствующее первому предложению будет иметь вид:

Объект	Отношение	Объект
Мэри	любит	яблоки

Можно поместить *имя отношения* перед объектами:

Отношение	Объект	Объект
любит	Мэри	яблоки

В предложении *«Бет любит нечто, если Мэри любит (это же) нечто»*, и заменим слово *«нечто»* на местоимение *«это»*:

```
«Бет любит это, если Мэри любит это».
```

Это предложение выражает два отношения «любит». Они соединены *условием*, выраженным словом «если». То есть это предложение является *условной предпосылкой* со связкой «если», указывающей на условное отношение. **Условие «если»** требует проверки предпосылки для вывода нового факта.

Таким образом, имеются следующие отношения:

Отношение	Объект	Объект	Условие
любит	Мэри	яблоки	
любит	Бет	это	если
любит	Мэри	это	

2. Предикаты в Прологе (Prolog)

Местоимение «это» может иметь переменное значение, изменяющиеся от предложения к предложению. Если неизвестно, что обозначает слово «это», то нельзя вывести новых фактов. Если значение слова «это» известно, то тогда можно вывести новые факты, связанные условным отношением с другими уже известными фактами.

Поэтому можно использовать предыдущую предпосылку «Мэри любит яблоки». Местоимение «это» имеет определённое значение - «яблоки».

Поэтому предложения могут быть переформулированы следующим образом:

Отношение(предикат)	Объект	Объект	Условие
любит	Мэри	яблоки	
любит	Бет	это (яблоки)	если
любит	Мэри	это (яблоки)	

Теперь может быть получен новый факт:

```
"Бет любит яблоки".
```

Отношение «любит» связывает объекты «Мэри» и «яблоки» в конструкцию, обладающую определенным смыслом.

«Любит» является *предикатом* (термом или символом предиката), а **«Мэри»** и **«яблоки»** - *объектами предиката*. Отношение между Мэри или Бет и яблоками называется **отношением связывания**. *Объектами* в этом отношении являются Мэри, Бет и яблоки.

Перечень объектов предиката в Прологе (Prolog) заключается в *круглые скобки* (), а в качестве разделителя используется *запятая*.

2. Предикаты в Прологе (Prolog)

Предикаты в Прологе (Prolog) описываются в разделе **predicates**. В этом же разделе вы должны задать *тип объектов* каждого из предикатов.

```
predicates
  likes(symbol,symbol)
```

Пример предикатов:

```
предикат(объект1,объект2).
любит (мэри,яблоки).
```

Пример программы:

```
predicates
 likes(symbol,symbol)
clauses
 likes(mary, apples).
 likes(beth, X) if likes(mary, X).
```

Для запуска программы нажмите Alt+R (Turbo Prolog). В ответ на запрос системы Цель, необходимо ввести цель – likes(beth,apples). Система даст ответ «Да».

Как это выглядит в Прологе:

```
predicates
  likes(symbol,symbol).
  vopros
clauses
  likes(mary,apples).
  vopros:-likes(mary,X),write("Beth любит ",X),nl.
goal
  vopros
```

2. Предикаты в Прологе (Prolog)

Предикаты и утверждения разных арностей

При трансляции утверждений Пролог контролирует правильность написания имени предиката, количество объектов и типы их доменов.

Термин *арность* в Прологе (Prolog) обозначает число объектов утверждения. Так утверждение likes(mary,apples) имеет арность 2.

Предикат	Утверждения	Арность
go_home	go_home	0
female(person)	female(betty) female(katty)	1
father(person,person)	father(john,kathy) father(john,tom)	2

В Прологе (Prolog) предикаты без объектов, то есть нулевой арности, называют ещё «голыми».

Предикаты с нулевой арностью в Прологе (Prolog) часто используются для построения правил, как, например:

```
go_home if (condition(sickness) and transportation(bus))
```

Предикаты арности 1 в Прологе (Prolog) полезны при сортировке объектов программы по доменам. В приведенном в таблице примере предикат female указывает, что имя betty относится к домену женских имен.

2. Предикаты в Прологе (Prolog)

Предикаты и утверждения разных арностей

Предикаты арности 2 в Прологе (Prolog) используются для установления отношения между двумя объектами. Например, предикат:

```
father(person,person)
```

и соответствующее ему утверждение:

```
father(john,kathy)
```

Предикаты арностей выше 2 в Прологе (Prolog) пригодны для установления связи нескольких объектов по какому-либо признаку. В утверждении:

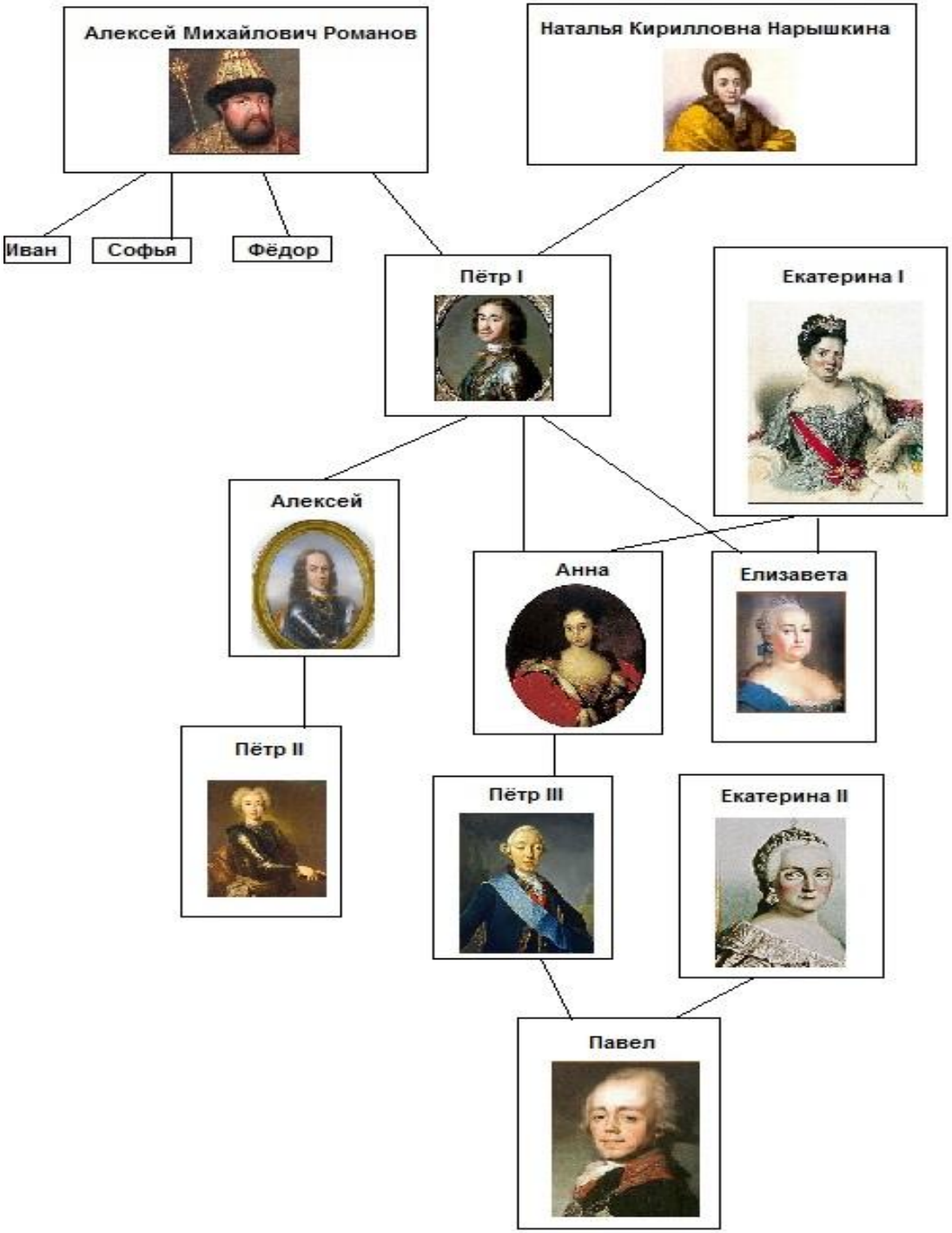
```
europe(“France”,”Germany”,”Spain”,”Italy”)
```

все используемые значения: France, Germany, Spain и Italy принадлежат домену country. Общим для них является то, что все они обозначают европейские страны.

Первая программа

```
/* Програнна синонины */

/* Описание предикатов */
predicates
  synonyn(string,string).
/* Утверждения*/
clauses
  synonyn("любить","иметь слабость").
  synonyn("уважать","чтить").
  synonyn("дорогой","яхонтовый").
  synonyn("красивый","щегольской").
goal
  synonyn("яхонтовый",X);synonyn(X,"яхонтовый"),
  write("Синоним для 'яхонтовый' - это ",X),nl.
```


Алексей Михайлович Романов
Наталья Кирилловна Нарышкина
Иван
Софья
Фёдор
Пётр I
Екатерина I
Алексей
Анна
Елизавета
Пётр II
Пётр III
Екатерина II
Павел

Поиск братьев и сестер в семье Романовых

```
predicates
  parent(string,string)
  male(string)
  female(string)
  brother(string,string)
  sister(string,string)
  print1
  print2
  vopros
```

Поиск братьев и сестер в семье Романовых

```
clauses
  parent("Aleksei Mihailovich","Petr I").
  parent("Natalya","Petr I").
  parent("Aleksei Mihailovich","Sofya").
  parent("Aleksei Mihailovich","Ivan").
  parent("Aleksei Mihailovich","Fyedor").
  parent("Petr I","Aleksei Petrovich").
  parent("Petr I","Anna").
  parent("Petr I","Elizaveta").
  parent("Ekaterina I","Anna").
  parent("Ekaterina I","Elizaveta").
  parent("Aleksei Petrovich","Petr II").
  parent("Anna","Petr III").
  parent("Petr III","Pavel").
  parent("Ekaterina II","Pavel").
  male("Aleksei Mihailovich").
  male("Ivan").
  male("Fyedor").
  male("Petr I").
  male("Aleksei Petrovich").
  male("Petr II").
  male("Petr III").
  male("Pavel").
  female("Natalya").
  female("Sofya").
  female("Ekaterina I").
  female("Anna").
  female("Elizaveta").
  female("Ekaterina II").
  brother(X,Y):-parent(Z,X),parent(Z,Y),male(X),X<>Y.
  sister(X,Y):-parent(Z,X),parent(Z,Y),female(X),X<>Y.
  print1:-brother(X,Y),write(X," - brat ",Y),nl,fail.
  print2:-sister(X,Y),write(X," -sestra ",Y),nl,fail.
  vopros:-parent(X,"Petr III"),write(X," - roditel Petr III"),nl,fail.
```

Задание на закрепление темы прошлого занятия

- Суть задания: задаем предикат «нравится» с 2 термами. Они будут означать кому нравится и что нравится.
- Далее зададим несколько личностей и их любимые виды спорта.

```
likes("Елена Исинбаева","Прыжки с шестом").
likes("Екатерина Гамова","волейбол").
likes("Мария Шарапова","Теннис").
likes("Евгений Плющенко","Фигурное катание").
likes("Аделина Сотникова","Фигурное катание").
likes("Елена Ильиных","Фигурное катание").
likes("Юлия Ефимова","плавание").
likes("Яна Кудрявцева","художественная гимнастика").
likes("Маргарита Мамун","художественная гимнастика").
likes("Алина Кабаева","художественная гимнастика").
likes("Евгения Канаева","художественная гимнастика").
```

Задание на закрепление темы прошлого занятия

- Теперь надо ответить на 3 вопроса:

1. Вопрос1: занимается ли Юлия Ефимова волейболом? Вопрос требует ответа «да» или «нет».
2. Вопрос2: Каким видом спорта занимается Елена Ильиных?
3. Вопрос3: Выведите всех спортсменов, которые занимаются художественной гимнастикой.

Как следовало выполнить задание:

```
vopros3:-likes(X,"художественная гимнастика"),write(X),
nl,fail.
vopros2:-likes("Елена Ильиных",Y),write(Y),nl.
vopros1:-likes("Юлия Ефимова","волейбол").
```

Попробуем создать некое подобие базы данных:

```
predicates
  mashina(string,string,integer,string,integer,real,integer).

clauses
  mashina("Toyota","Camry",2006,"белая",640,2.4,170).
  mashina("Toyota","Allion",2011,"серебристая",855,1.8,89).
  mashina("Toyota","Nadia",2000,"красная",275,2,170).
  mashina("Toyota","Caldina",2002,"черная",210,1.5,208).
  mashina("Mitsubishi","Fuso",1993,"белая",650,7.545,220).
  mashina("Toyota","Camry",2007,"черная",775,2.4,112).
  mashina("Toyota","Duet",2001,"бордовая",120,0.989,160).
  mashina("Lexus","RS350",2009,"белая",1400,9,80).
  mashina("Lexus","RS300",2000,"черная",550,3,160).
```

Теперь будем пробовать задавать разные цели:

1. Цель1 (дала ответ, что машины марки Toyota есть).

```
Цель: mashina("Toy
ota",_,_,_,_,_,_)
Да
```

2. Цель2 (дала ответ, что машины марки Toyota 2005 года выпуска нет).

```
Цель: mashina("Toy
ota",_,2005,_,_,_,
_)
Не
```

3. Цель3 (дала ответ, какие машины есть в заданном ценовом диапазоне).

```
Цель: mashina(X,Y,
_,_,Z,_,_),Z<500,Z
>120
X=Toyota, Y=Nadia,
 Z=275
X=Toyota, Y=Caldin
a, Z=210
2 решен.  s
```

Самостоятельное задание:

1. Вывести все старые машины (старше 2003 года), создав для этого специальный предикат old_mashins.

```
old_mashins:-mashina(X,Y,Z,_,_,_,_),Z<2003,write(X," ",Y," ",Z),nl,fail.
```

2. Вывести все машины нужного цвета, создав специальный предикат color(string).

```
color(X):-mashina(X1,Y,Z,X,X2,X3,X4),write(X1," ",Y," ",Z),nl,fail.
```

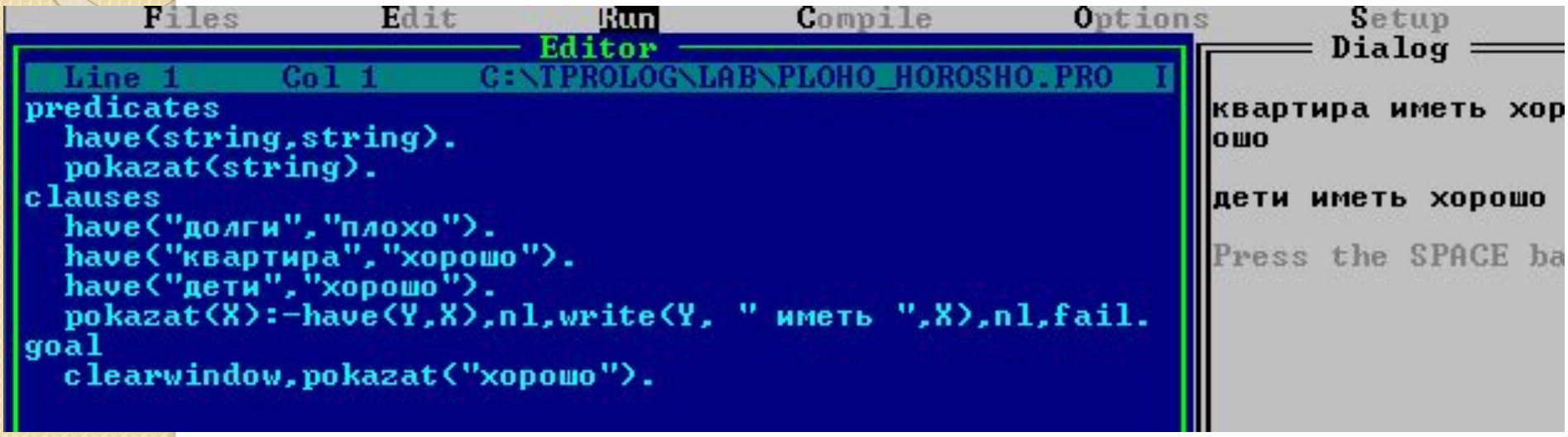
3. Вывести, какие есть машины для девушек (чтобы цвет был не белый, не черный и объем не более 1.5 л)

```
Dlya_devushek:-mashina(M,M1,G,Cv,Cena,O,P),Cv<>"белая",Cv<>"черная",O<1.5,
write(M," ",M1," ",G," Цена: ",Cena,"т.р."),nl,fail.
```

Самостоятельное задание

- Известны высказывания о том, что хорошо, что плохо. Например, иметь квартиру хорошо, детей - хорошо, а долги - плохо. Выведите все вещи, которые указаны в базе, как те, которые иметь хорошо.

Решение самостоятельной задачи



Задача 1: Нахождение корней уравнения $ax^2+bx+c=0$

DETERMIN.BAK — Блокнот

Файл Правка Формат Вид Справка

```
predicates
  solve(real,real,real).
  reply(real,real,real).
  zapros1
clauses
  solve(A,B,C):-D=B*B-4*A*C,reply(A,B,D),nl.
  reply(_,_,D):-D<0,write("Корней нет!").
  reply(A,B,D):-D=0,X=-B/(2*A),write("x1=x2=",X).
  reply(A,B,D):-D>0,SqrtD=sqrt(D),X1=(-B+SqrtD)/(2*A),
  X2=(-B-SqrtD)/(2*A),write("x1=",X1," and x2=",X2).
  zapros1:-solve(1,2,1).
```

Задача 2: Запутанные суждения

- Бутси – коричневая кошка. Корни – черная кошка. Мак – рыжая кошка. Флэш, Ровер, Спот – собаки, Ровер – рыжая, а Спот – белая. Все животные, которыми владеют Том и Кейт, имеют родословные. Том владеет всеми черными и коричневыми животными, а Кейт владеет всеми собаками небелого цвета, которые не являются собственностью Тома. Алан владеет Мак, если Кейт не владеет Бутси и если Спот не имеет родословной. Флэш – пятнистая собака. Определить, какие животные не имеют хозяев.

ZAPUT_SU.BAK — Блокнот

Файл Правка Формат Вид Справка

```
domains
  animal_name,color,name=string.
predicates
  cat(animal_name).
  dog(animal_name).
  color(animal_name,color).
  have(name,animal_name).
  rodoslovnaya(animal_name).
  animal(animal_name).
clauses
  cat("Butsi").
  cat("Korni").
  cat("Mak").
  dog("Flesh").
  dog("Rover").
  dog("Spot").
  color("Butsi","brown").
  color("Korni","black").
  color("Mak","red_headed").
  color("Rover","red_headed").
  color("Spot","white").
  color("Flesh","spotted").
```

Построение системы утверждений

```
/*Животное-это кошка или собака*/
animal(X):-cat(X);dog(X).

/*Животные? имеющие родословные*/
rodoslovnaya(X):-animal(X),have(Tom,X).
rodoslovnaya(X):-animal(X),have(Keit,X).

/*Том владеет всеми черными и коричневыми животными*/
have("Tom",X):-color(X,black);color(X,brown).

/*Кейт владеет собаками небелого цвета, не являющимися собственностью Тома*/
have("Keit",X):-dog(X),not(color(X,white)),not(have("Tom",X)).

/*Алан владеет Мак, если Кейт не владеет Бутси и Спот не имеет родословной*/
have("Alan","Mak"):-not(have("Keit","Butsi")),not(rodoslovnaya("Spot")).

goal
clearwindow,animal(X),not(have(_,X)),write(X),nl,fail.
```

Задача3: перевозка через реку волка, козы и капусты

- Крестьянин должен перевезти в лодке через реку волка, козу и капусту. Вместе с человеком в лодке может поместиться или волк, или коза, или капуста, причем человеку приходится охранять козу от волка и капусту от козы.

Решение задачи 3

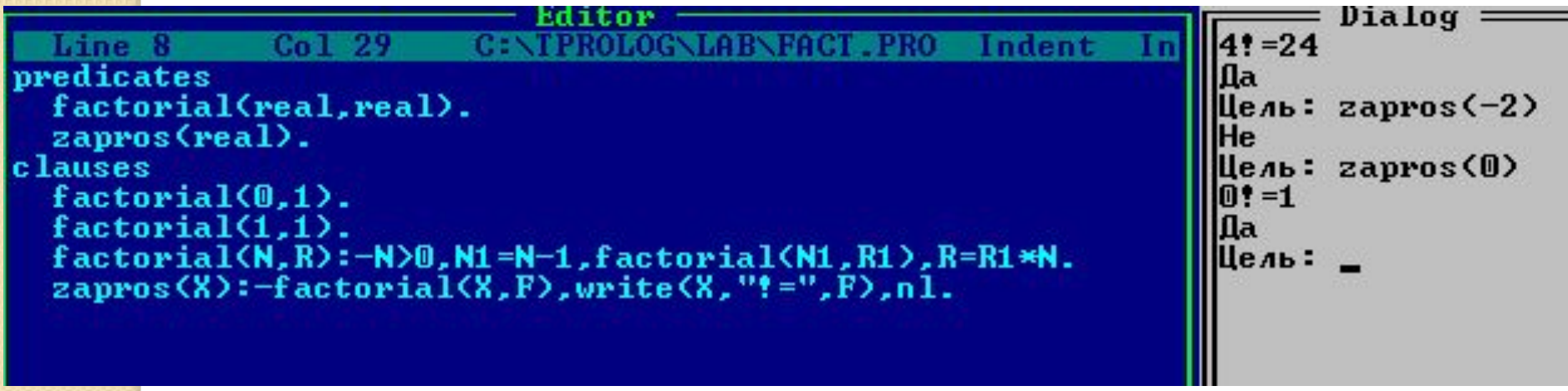
```
predicates
  move(string,string,string,string).
  move(string,string,string).
  move(string,string).
  move(string).
  move.
clauses
  move("Крестьянин","Волк","Коза","капуста"):-move("Волк","капуста"),write("Кр-н уплыл козой. ").
  move("Волк","капуста"):-move("Крестьянин","Волк","капуста"),write("Кр-н вернулся один. ").
  move("Крестьянин","Коза"):-move(),write("Никого не осталось.").
  move("Крестьянин","Волк","капуста"):-move("капуста"),write("Кр-н уплыл с волком. ").
  move("Крестьянин","Волк","капуста"):-move("Волк"),write("Кр-н уплыл с капустой. ").
  move("Крестьянин","Коза","Волк"):-move("Коза"),write("Кр-н забрал волка, а козу оставил. ").
  move("Крестьянин","Коза","капуста"):-move("Коза"),write("Кр-н капусту забрал,а козу оставил. ").
  move("Волк"):-move("Крестьянин","Коза","Волк"),write("Кр-н вернулся с козой. ").
  move("капуста"):-move("Крестьянин","Коза","капуста"),write("Кр-н вернулся с козой. ").
  move("Коза"):-move("Крестьянин","Коза"),write("Кр-н вернулся за козой. ").
  move().
goal
  clearwindow,move("Крестьянин","Волк","Коза","капуста").
```

Задача 4: вычисление факториала.

0!=1

1!=1

n!=1*2*3*…*(n-1)*n=(n-1)!*n



СПИСКИ в прологе

- Указываются через запятую в квадратных скобках:

```
[1,2,3]
```

```
[dog,cat,canary]
["valerie ann","jennifer caitlin","benjamin thomas"]
```

- Чтобы объявить список, надо

```
domains
    integerlist=integer*
```

Символ (*) означает "список чего-либо"; таким образом, **integer*** означает "список целых".

Голова и хвост списка

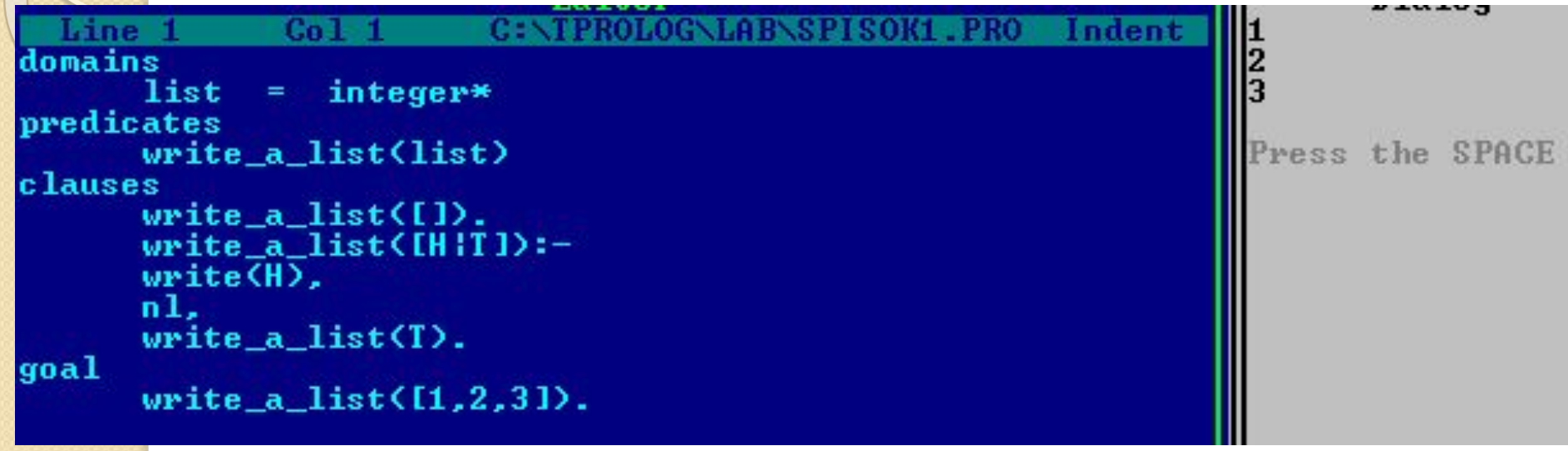
Список является рекурсивным составным объектом. Он состоит из двух частей - головы, которая является первым элементом, и хвоста, который является списком, включающим все последующие элементы. *Хвост списка - всегда список, голова списка - всегда элемент*. Например:

```
голова [a,b,c] есть a
хвост [a,b,c] есть [b,c]
```

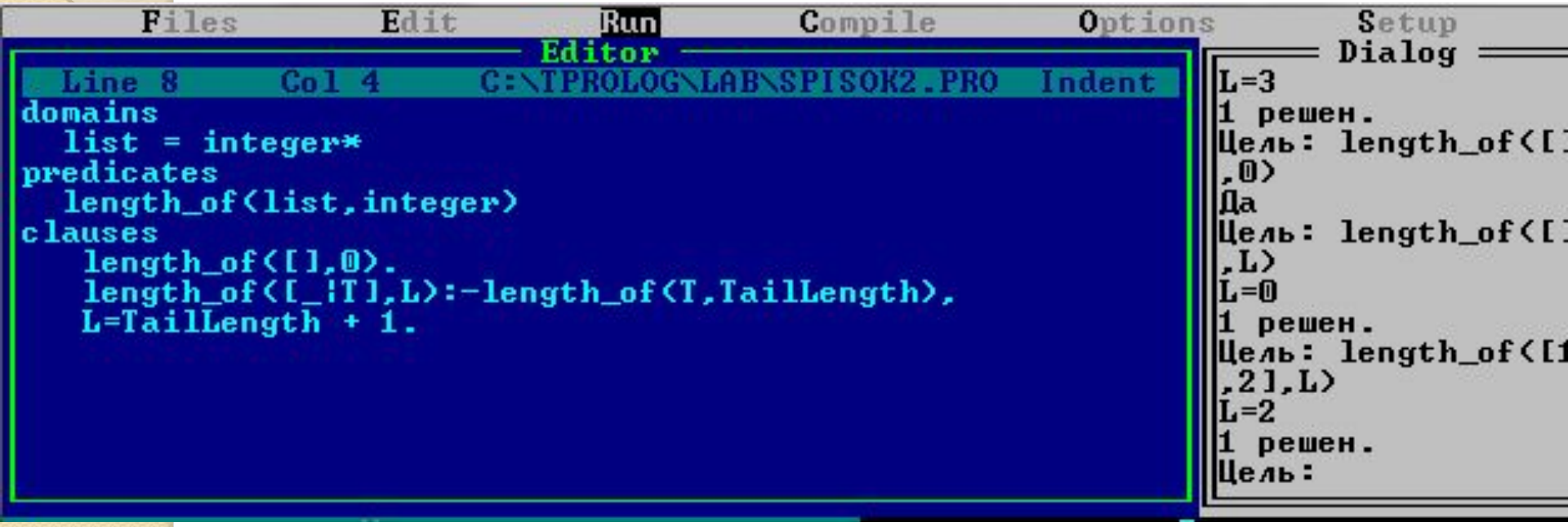
Что происходит, когда вы доходите до одноэлементного списка? Ответ таков:

```
голова[c] есть c
хвост [c] есть [ ]
```

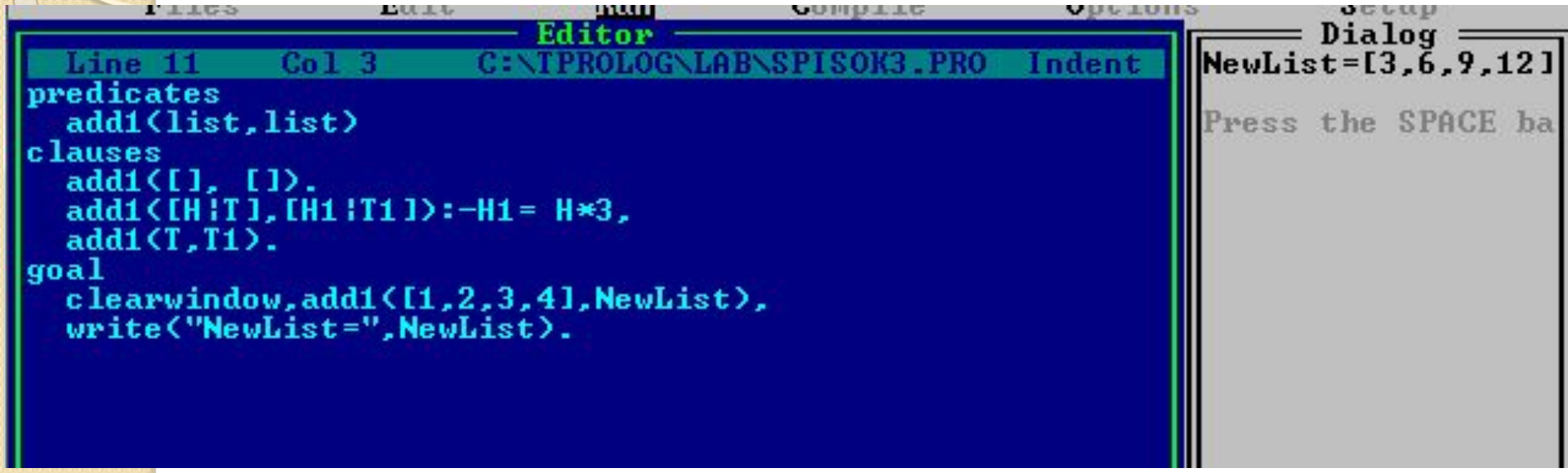
Как вывести все элементы списка



Подсчет количества элементов списка:

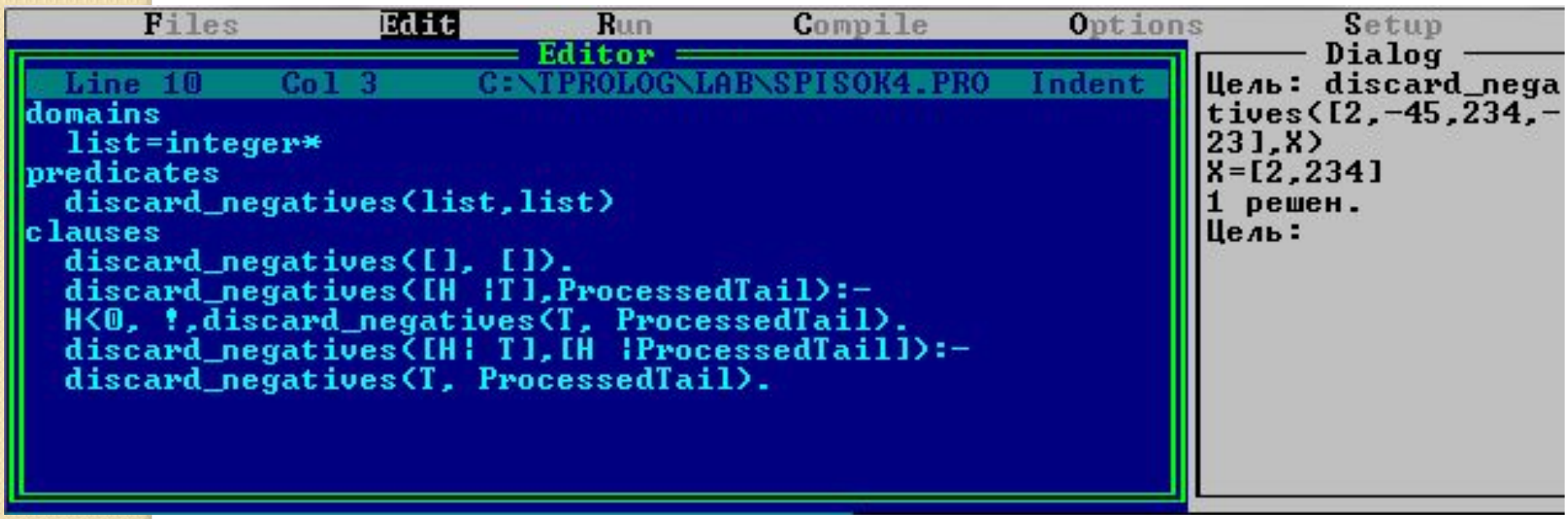


Получение нового списка из известного по правилу:

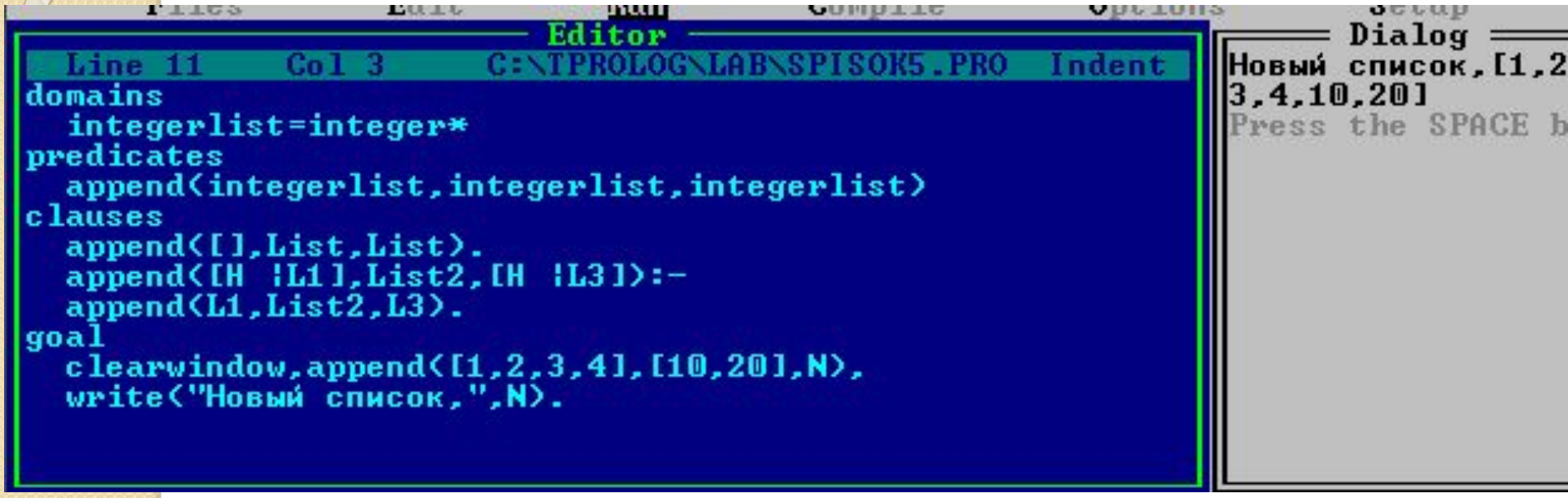


Самостоятельное задание

- Выведите новый список, который будет содержать все значения конкретного списка, удовлетворяющие какому-либо условию. Например, содержать только положительные элементы старого списка.



Объединение списков



Встроенный предикат findall

- Встроенный предикат использует целевые утверждения в качестве одного из своих аргументов и собирает все решения для этого целевого утверждения в единый список. У предиката **findall** три аргумента:
- **varName**(имя переменной) - определяет параметр, который необходимо собрать в список;
- **myPredicate** (мой предикат) - определяет предикат, из которого надо собрать значения;
- **ListParam** (список параметров) - содержит список значений, собранных методом поиска с возвратом. Заметьте, что должен быть определенный пользователем тип, которому принадлежат значения **ListParam**.

Определение среднего возраста группы людей

```
domains
  name,address=string
  age=integer
  list=age*
predicates
  person(name,address,age)
  sumlist(list,age,integer)
clauses
  sumlist([],0,0).
  sumlist([H |T],Sum,N):-
  sumlist(T,S1,N1),
  Sum=H+S1,
  N=1+N1.
  person("Яхина Асия Сергеевна","Центр",33).
  person("Служаева Ксения","КСК", 22).
  person("Фалейчик Андрей Анатольевич", "Забво", 62).
goal
  findall(Age,person(_, _, Age),L),
  sumlist (L,Sum,N), Ave=Sum/N,write("Average=", Ave),nl.
```

Создание окон

- **makewindow(<номер окна>, <атрибуты окна>, <атрибуты рамки>, <заголовок окна>, <начальный номер строки>, <начальный номер столбца>, <высота окна>, <ширина окна>).**

Цвет текста и фона

Таблица 1. **Цвета текста**

Значение	Цвет	Значение	Цвет
0	Черный	8	Серый
1	Синий	9	Светло-синий
2	Зеленый	10	Светло-зеленый
3	Голубой	11	Светло-голубой
4	Красный	12	Светло-красный
5	Фиолетовый	13	Светло-фиолетовый
6	Коричневый	14	Желтый
7	Белый	15	Интенсивно-белый

Таблица 2. **Цвета фона**

Значение	Цвет	Значение	Цвет
0	Черный	64	Красный
16	Синий	80	Фиолетовый
32	Зеленый	96	Коричневый
48	Голубой	112	Белый

Цвет рамки окна

Таблица 3. **Цвета рамки окна**

Значение	Вид рамки окна	Значение	Вид рамки окна
0	Нет рамки	-1	Мерцающая белая рамка
1	Синяя рамка	-2	Мерцающая желтая рамка
2	Зеленая рамка	-3	Мерцающая фиолетовая рамка
3	Светло-синяя рамка	-4	Мерцающая красная рамка
4	Красная рамка	-5	Мерцающая светло-синяя рамка
5	Фиолетовая рамка	-6	Мерцающая светло-зеленая рамка
6	Желтая рамка	-7	Мерцающая синяя рамка
7	Белая рамка	-8	Мерцающая серая рамка
8	Коричневая рамка	-	-

Размер и положение окна

- ***Определение размеров и положения окон.*** Аргумент начальный номер строки предиката **makewindow** есть целое число, определяющее верхнюю строку (линию) создаваемого окна. Максимальное количество строк, умещающихся на экране, 25. Значение 4 указывает, что окно начинается с четвертой строки. Для рассматриваемого аргумента можно использовать значения от 0 до 24.
- Аргумент ***Начальный номер столбца*** предиката **makewindow** есть целое число, указывающее крайний левый столбец окна. Максимальное число столбцов, умещающихся на экране, 80 (от 0 до 79).
- Аргумент ***Высота окна*** есть целое число, определяющее количество строк, занимаемых создаваемым окном. Максимально возможное значение аргумента 25.
- Аргумент ***Ширина окна*** есть целое число, указывающее число столбцов, занимаемых окном. Максимальное значение аргумента 80.

Как в прологе вывести окно

```
predicates
  hello
clauses
  hello:-makewindow(1,12,96,"Вопрос!",1,36,23,44),
  nl,write("Назовите, пожалуйста, свое имя: "),
  cursor(4,5),readln(X),write("Добро пожаловать, ",X).
goal
  hello
```

Самостоятельное задание

makewindow(1,7,7,"Полный экран",0,0,25,80) - данному окну присвоен номер 1. Символы в нем будут белые, фон черный, рамка белая и метка окна - "Полный экран". Верхний левый угол окна находится на строке 0, столбце 0, а само окно имеет 25 строк и 80 столбцов;

makewindow(2,4,1,"Меню",4,20,16,40) - этому окну присвоен номер 2. Оно имеет метку "Меню". Символы в нем красные на черном фоне, и ограничено оно синей рамкой. Окно начинается с 4-й строки и 20-го столбца, имеет 16 строк и 40 столбцов.

Предикаты для работы с окнами:

1. Предикат **shiftwindow** используется для смены текущего окна (переключения). Его форма:

shiftwindow(<номер окна>)

- Параметр номер окна является целым числом, приписываемым окну при его создании. Например, задание предиката **shiftwindow(3)** вызывает переназначение всех операций ввода и вывода в окно, с номером 3.

2. Предикат gotowindow позволяет выполнять быстрое переключение между двумя окнами, которые не перекрываются. Его форма:

gotowindow(<номер окна>)

- Этот предикат выполняется быстрее, чем **shiftwindow**, и поэтому его следует использовать для переключения между окнами, содержащими большое количество текста.

3. Предикат clearwindow удаляет из текущего окна все текстовые и графические изображения. Предикат не имеет аргументов:

clearwindow

- Окно и рамка окна, если она имеется, не разрушаются. Окно целиком закрашивается соответствующим цветом фона.

4. Предикат removewindow удаляет текущее окно с экрана. Предикат аргументов не имеет, поэтому его синтаксис прост:

removewindow

- Все текстовые и графические изображения, находящиеся в окне, также удаляются. Если за данным окном находится другое окно, то это окно и его содержимое становятся видимыми. Если удаляется последнее из заданных окон, на экране появляется то изображение, которое было на нем до создания окон.

Установка курсора в окне

- По умолчанию предполагается, что курсор расположен в верхнем левом углу окна. Однако можно использовать предикат **cursor** и помещать курсор в любую позицию текущего окна. Этот предикат имеет вид:
- **cursor(<номер строки>,<номер столбца>).**
- Аргументы предиката являются целыми числами, задающими номера строки и столбца, по отношению к верхней строке и крайнему левому столбцу экрана. Строки и столбцы нумеруются, начиная с нуля: предикат **cursor(0,0)**обеспечивает вывод текста, начиная с верхнего левого угла экрана. Если случайно указывается позиция курсора, выходящая за рамки текущего окна, то во время выполнения программы будет выдано сообщение об ошибке.

Пример вывода сообщений

Например, следующие три предиката могут быть использованы для вывода сообщения в центре экрана:

```
makewindow(1,7,7,"",1,1,8,28),
cursor(4,12),
write("Отличный день").
```

Если опустить предикат **cursor**, данное сообщение будет выведено, начиная с левого верхнего угла окна. Аргументами предиката **cursor** могут также быть переменные, которым присваиваются целочисленные значения. Другой формой подцели, размещающей сообщение в центре окна, будет:

```
makewindow(1,7,7,"",1,1,8,28),
Row=4,
Col=12,
cursor(Row,Col),
write("Отличный день").
```

Если в качестве аргументов предиката **cursor** используются неопределенные переменные, то этим переменным присваиваются текущие значения строки и столбца. Подцель:

```
makewindow(1,7,7,"",1,1,8,28),
Row=4,
Col=12,
cursor(Row,Col),
write("Отличный день "),
cursor(What_row,What_column).
```

присвоит переменной **What_row** значение 4, а переменной **What_column** - значение 27.

Создание перекрывающихся окон

```
predicates
        make_windows_write_text
clauses
        make_windows_write_text:-makewindow(1,7,7,"Жизнь звезды",3,12,10,40),
           cursor(3,8),
           write("ПОЯВИЛАСЬ ЧЕРНАЯ ДЫРА."),nl,
           makewindow(2,7,7," Жизнь звезды ",5,14,10,40),
           shiftwindow(2),
           cursor(3,12),
           write("ЗВЕЗДА ВЗОРВАЛАСЬ."),nl,
           makewindow(3,7,7," Жизнь звезды ",7,16,10,40),
           shiftwindow(3),
           cursor(3,11),
           write("ЗВЕЗДА СВЕТИТ."),nl,
           makewindow(4,7,7," Жизнь звезды ",9,18,10,40),
           shiftwindow(4),
           cursor(3,11),
           write("ЗВЕЗДА РОДИЛАСЬ."),nl,
           cursor(6,4),
           write("Нажмите пробел"),
           readchar(_),
           removewindow,
           cursor(6,2),
           write("Нажмите пробел "),
           readchar(_),
           removewindow,
           cursor(6,2),
           write("Нажмите пробел "),
           readchar(_),
           removewindow,
           cursor(7,2),
           write("Нажмите пробел "),
           readchar(_),
           removewindow,exit.
goal
        make_windows_write_text.
```

Создание меню с помощью окон

```
predicates
        repeat
        process(integer)
        show_menu
        run_file_utility
    goal
        run_file_utility.
    clauses
        run_file_utility:-
            show_menu,
            nl,write("Нажмите клавишу ПРОБЕЛ"),
            readchar(_),
            exit.
        repeat.
        repeat:-
            repeat.
        show_menu:-
            repeat,
            makewindow(1,7,7,"Главное меню",4,10,16,36),nl,
            write("          Выберите действие"),nl,nl,
            write("       0 выход из программы"),nl,
            write("       1 удалить файл"),nl,
            write("       2 переименовать файл"),nl,
            write("       3 создать файл"),nl,
            write("       4 выход в DOS"),nl,nl,
            write("Укажите номер:(0-4)  "),
            readint(X),
            X<5,
            process(X),
            X=0,
            !.
```

Создание меню с помощью окон

```
process(0):-
   nl,write(" Выбран выход из программы!"),nl.
process(1):-
   makewindow(2,7,7,"Удаление файла",12,36,10,36),
   write("Выбрано удаление файла."),nl,nl,
   write("Процесс будет реализован позднее."),nl, nl,
   write("Для продолжения нажмите пробел."),
   readchar(_),
   removewindow.
process(2):-
   makewindow(3,7,7,"Переименование файла",10,40,10,36),
   write("Выбрано переименование файла."), nl,nl,
   write("Процесс будет реализован позднее."),nl, nl,
   write("Для продолжения нажмите пробел."),
   readchar(_),
   removewindow.
process(3):-
   makewindow(4,7,7,"Создание файла",5,10,15,60),
   write("Выбрано создание файла."), nl,nl,
   write("Процесс будет реализован позднее."),nl, nl,
   write("Для продолжения нажмите пробел."),
   readchar(_),
   removewindow.
process(4):-
   makewindow(5,7,7,"Выход в DOS",10,40,10,35),
   write("Выбран временный выход в DOS."), nl,nl,
   write("Процесс будет реализован позднее."),nl, nl,
   write("Для продолжения нажмите пробел."),
   readchar(_),
   removewindow.
```

Файловая система в Прологе

- ***Пролог*** использует:

1. **current_readdevice** (текущее устройство чтения), с которого считывается ввод;
2. **current_write_device** (текущее устройство записи), на которое посылается вывод.

Как правило, текущим устройством чтения является клавиатура, а текущим устройством записи - экран дисплея.

Однако можно и назначить другие устройства. Например, ввод может читаться из файла, хранимого во внешней памяти (возможно, на диске).

Можно даже переопределить устройства текущего ввода и вывода во время исполнения программы.

Доступ к файлу

Для доступа к файлу надо сначала его открыть. Файл может быть открыт:

- для чтения;
- для записи;
- для добавления;
- для модификации.

Файл открытый для любого действия, отличного от чтения, должен быть закрыт после завершения операции.

В противном случае внесенные в файл изменения могут быть потеряны. Можно открыть несколько файлов одновременно. При этом и ввод вывод могут быть быстро переназначены между открытыми файлами. Открытие и закрытие файлов занимает намного больше времени, чем переназначение потоков данных между ними.

Описание файла

- Когда ***Пролог*** открывает файл, он связывает символическое имя с действительным именем файла операционной системы и использует это символическое имя для направления ввода и вывода.
- Символические имена файлов должны начинаться с маленькой буквы и должны быть объявлены в описании домена **file**. Например:
- **file=file1; source; auxiliary; somethingElse**
-
- В любой программе разрешен только один домен **file**.

Альтернативы file

- ***Пролог*** распознает пять встроенных альтернатив **file**, описанных в таблице:

Альтернатива	Описание
keyboard	Чтение с клавиатуры (по умолчанию)
screen	Запись в монитор
stdin	Чтение из стандартного ввода
stdout	Запись в стандартный вывод
stderr	Запись на стандартное устройство для вывода ошибок

- Эти встроенные альтернативы не должны встречаться в описании **file**. Открывать и закрывать их не требуется.

Путь к файлу в Прологе

- Во время открытия файла необходимо помнить, что обратный слэш (\), используемый для указания подкаталога диска, в **DOS**-ориентированных версиях **Пролога** является **ESC**-символом (управляющим).
- Поэтому при указании пути доступа файла в программе нужно всегда указывать два обратных слэша (\\). Например, строка:
- **"c:\\prolog\\include\\iodecl.con"** представляет путь доступа к файлу:
- **c:\prolog\include\iodecl.con**

Предикат openread/2

- Предикат **openread** открывает файл **OSFileName** дли чтения, используя формат:
- **openread(SymbolicFileName,OSFileName)** ***% (i, i)***
- ***Пролог*** обращается к открытому файлу по символическому имени **SymbolicFileName**, объявленному в домене **file**. Если файл не может быть открыт, ***Пролог*** выдаст сообщение об ошибке.

Предикат openwrite/2

- Предикат **openwrite** открывает файл **OSFileName** для записи, используя формат:
- **openwrite(SymbolicFileName,OSFileName)** ***% (i,i)***
- Если файл уже существует, то он уничтожается. В противном случае ***Пролог*** создает новый файл и помещает его в соответствующем каталоге. Если файл не может быть создан, ***Пролог*** выдаст сообщение об ошибке.

Предикат openappend/2

- Предикат **openappend** открывает файл **OSFileName** для записи в конец файла. При этом используется формат:

- **openappend(SymbolicFileName,OSFileName)** ***% (i,i)***
-
- Если файл не может быть открыт на запись, ***Пролог*** сообщит об ошибке.

Предикат openmodify/2

- Предикат **openmodify** открывает файл **OSFileName** и для записи, и для чтения; если файл уже существует, он не будет перезаписан, **openmodify** имеет формат:
- **openmodify(SymbolicFileName,OSFileName)** ***% (i,i)***
- Если система не может открыть **OSFileName**, ***Пролог*** сообщит об ошибке. Для заполнения файла с произвольным доступом предикат **openmodify** может использоваться вместе со стандартным предикатом **filepos**.

Предикат filemode/2

- При открытии файла в текстовом режиме предикат **filemode** устанавливает указанный файл в текстовый или двоичный режим, используя формат:
- **filemode(SymbolicFileName,FileMode)** ***% (i,i)***
- Если **FileMode=0**, файл **SymbolicFileName** устанавливается в двоичный режим; если **FileMode=1**, то он устанавливается в текстовый режим.
- В текстовом режиме при записи к новым строкам добавляются символы "возврат каретки"\"перевод строки", а при чтении эта пара символов интерпретируется как новая строка.
- **Carriage return (возврат каретки) = ASCII 13**
  Line feed (перевод строки) = ASCII 10
- В двоичном режиме никаких преобразований не производится. Для чтения двоичного файла вы можете использовать только предикат **readchar** или предикаты для доступа к двоичным файлам.

Предикат closefile/1

- Предикат **closefile** закрывает указанный файл; он использует формат:

- **closefile(SymbollcFileName)** ***% (i)***

- Этот предикат всегда завершается успешно, даже если файл не был открытым.

Предикат readdevice/1

- Предикат **readdevice** переопределяет **current _read_device** (текущее устройство чтения) или возвращает его имя. Предикат имеет формат:
- **readdevice(SymbolicFileName)** ***% (i), (o)***
- Предикат **readdevice** переопределяет текущее устройство чтения, если переменная **SymbolicFileName** определена, и файл открыт для чтения.
  Если **SymbolicFileName** является свободной переменной, то **readdevice** присвоит ей имя текущего активного устройства чтения.

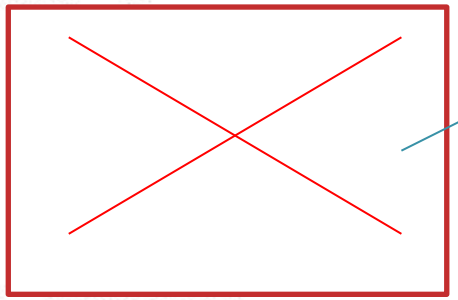
Предикат writedevice/1

- Предикат **writedevice** либо назначает, либо позволяет получить имя **current_ write_device** (текущего устройства записи). Он имеет формат:
- **writedevice(SymbolicFileName)** ***% (i), (o)***
- Предикат **writedevice** переопределит устройство записи, если указанный файл открыт для записи или добавления. Если переменная **SymbolicFileName** свободна, **writedevice** присвоит ей имя текущего активного устройства записи.

Пример работы с файлом

- Программа помещает символы, набранные на клавиатуре, в файл **TRYFILE.ONE** на текущем диске, использует стандартные предикаты **read** и **write**. Набираемые символы не выводятся на экран дисплея. Файл закрывается при нажатии клавиши #.

```
domains
      file=myfile
predicates
      readloop
clauses
```



```
readloop:-
   readchar(X),
  X<>'#',
  !,
  writedevice(myfile),
  write(X),
  writedevice(screen),
  write(X),
  readloop.
readloop.

goal
      write("Эта программа считывает все, что Вы вводите и записывает в файл"),nl,
      write("tryfile.one. Чтобы закончить ввод нажмите #"),nl,
      openwrite(myfile,"tryfile.one") ,
      writedevice(myfile),
      readloop,
      closefile(myfile),
      writedevice(screen),
      write("Все, что Вы ввели записалось в файл tryfile.one"),
      nl.
```

- Самостоятельное задание 1: допишите программу, чтобы она выводила эти символы и в файл, и на экран.
- Самостоятельное задание 2: допишите программу, чтобы она дописывала эти символы в файл, не удаляя содержимое файла.

Переопределение стандартного ввода-вывода

- Домен **file** имеет три дополнительные опции: **stdin, stdout, stderr**. Преимущество этих файловых потоков в том, что вы можете переназначить стандартный ввод/вывод в командной строке.

Файловый поток	Описание
stdin	Стандартный ввод является файлом, доступным только для чтения. По умолчанию это клавиатура, readdevice(stdin) назначает устройством ввода stdin
stdout	Стандартный вывод является файлом, доступным только для записи. По умолчанию это экран терминала, writedevice(stdout) назначает устройством ввода stdout
stderr	Стандартный вывод ошибок является файлом, который доступен только для записи. По умолчанию это экран терминала. Writedevice(stderr) назначает устройством для вывода сообщений об ошибках stderr

Предикаты работы со строками

Предикат filepos

- Предикат **filepos** может управлять позицией, где производится чтение или запись. Он имеет формат:
- **filepos(SymbolicFileName,FilePosition,Mode)**
- Если **FilePosition** - связанная переменная, предикат может изменить позицию чтения и записи для файла с именем **SymbolicFileName**. Если при вызове **FilePosition** является свободной переменной, то **SymbolicFileName** возвращает текущую позицию в файле.
- **Mode** является целой величиной и указывает, как должно интерпретироваться значение **FilePosition**.

Mode	FilePosition
0	Относительно начала файла
1	Относительно текущей позиции
2	Относительно конца файла (конец файла соответствует позиции 0)

- Когда возвращается **FilePosition**, то **filepos** возвращает позицию относительно начала файла независимо от значения **Mode**.

Предикат eof/1

- Предикат **eof** проверяет, является ли позиция, полученная в процессе чтения, концом файла. В этом случае **eof** успешен. В противном случае он терпит неуспех. Предикат **eof** имеет вид:
  - **eof(SymbolicFileName)** ***% (i)***
- **eof** выдает ошибку во время выполнения, если файл был открыт с правами только на запись.
- Пример, как предикат **eof** можно использовать для определения предиката **repfile**, который полезен при работе с файлами, **repfile** генерирует точку возврата до тех пока не будет достигнут конец файла.

```
predicates
    repfile(FILE)
clauses
    repflie(_).
    repfile(F) :-
        not(eof(F)),
        repfile(F).
```

Задача 1: символы из одного файла переписать в другой файл, заменив их все на такие же, но заглавные.

```
domains
        file=input; output
predicates
        convert_file
        repfile(FILE)
clauses
        convert_file:-
           repfile(input),
           readln(Ln), % Перевод букв строки в заглавные
           upper_lower(LnInUpper,Ln) ,
           write(LnInUpper),nl,
           fail.
      convert_file.
      repfile(_).
      repfile(F):-
         not(eof(F)),
         repfile(F).
goal
      write("Какой файл будем конвертировать?"),
      readln(InputFileName),nl,
      write("В какой файл запишем результат?"),
      readln(OutputFileName),nl,
      openread(input,InputFileName),
      readdevice(input),
      openwrite(output,OutputFileName),
      writedevice(output),
      convert_file,
      closefile(input),
      closefile(output).
```

Предикат flush/1

- Предикат **flush** записывает содержимое внутреннего буфера в именованный файл.
- Он имеет формат.
  - **flush(SymbolicFileName)** ***% (i)***

Предикат existfile/1

- Предикат **existfile** выполняется успешно, если файл **OSFileName** будет найден. Формат его следующий:
  - **existfile(OSFileName)** ***% (i)***
- Предикат **OSFileName** может содержать каталог, а само имя может содержать знаки подстановки, как **c:\psys*.cfg**. Предикат **existfile** завершается неуспешно, если имя файла не найдено в обозначенном пути каталога. Однако, заметьте, несмотря на то, что **existfile** находит все файлы, включая файлы с установленными атрибутами "**system**" (Системный) и "**hidden**" (Скрытый), он не находит каталоги.
- Для проверки того, что файл присутствует на диске (прежде чем открывать его), вы можете воспользоваться:

```
open(File, Name):-
   existfile(Name),
   !,
   openread(File,Name).
open(_, Name):-
   writeName("Error: the file",Name," is not found").
```

Предикат deletefile/1

- Предикат **deletefile** удаляет файл, заданный его аргументом, и имеет формат:
  - **deletefile(OSFileName)** ***% (i)***
- Предикат **deletefile** даст ошибку, если не сможет удалить файл.

Предикат renamefile/1

- Предикат **renamefile** переименовывает файл с именем **OldOSFileName** в **NewOSFileName**.
- Он имеет формат:
  - **renamefile(OldOSFileName, NewOSFileName)** ***%(i,i)***
- Предикат **renamefile** будет успешен, если файл с именем **NewOSFileName** не существует, и оба имени являются допустимыми файловыми именами. В противном случае будет выдана ошибка.

Предикат disk/1

- Предикат **disk** используется для изменения текущего диска и/или каталога/подкаталога и имеет формат:
  - **disk(Path)** ***%(i),(o)***
- При вызове со свободной переменной в качестве параметра, **disk** возвратит текущий каталог.

Предикат frontchar/3

- Предикат **frontchar** действует согласно равенству:
- **String1 = *объединение* Char и String2**
- и имеет следующий формат:
  - **frontchar(String1,Char,String2)**

- Предикат **frontchar** имеет три аргумента: первый из них - строка, второй - символ (первый символ первой строки), третий - остаток первой строки.
- Предикат **frontchar** можно использовать для расщепления строки в последовательность символов или для создания строки из последовательности символов, а также проверки символов в строке. Если аргумент **String1** связан со строкой нулевой длины, то предикат завершается неуспешно.

Задача 2: преобразовать строку в список символов.

```
domains
  charlist=char*
predicates
  string_chlist(string,charlist)
clauses
  string_chlist("",[ ]):-!.
  string_chlist(S,[H |T]):-
          frontchar(S,H,S1),
          string_chlist(S1,T).
goal
  string_chlist("Kacca",Z),write(Z).
```

Предикат fronttoken/3

- Предикат **fronttoken** выполняет три взаимосвязанные функции, в зависимости от типа аргументов, который используется для обращения к нему.
- **fronttoken(String1,Token,Rest)** ***%(i,o,o) (i,i,o) (i,o,i) (i,i,i) (o,i,i)***
- В случае потока **(i,o,o) fronttoken** находит первую лексему в **String1**, связывает ее с **Token**, а остаток **String1**связывает с **Rest**. Варианты потока **(i,i,o), (i,o,i)**, а также **(i, i, i)** служат для проверки: если связанные аргументы соответствуют частям **String1** (первой лексеме, всему, что находится после первой лексемы, или же и тому и другому), то **fronttoken** завершается успешно, в противном случае - неуспешно.
- В случае если использован поток **(o,i,i)**, предикат создает объединение **Token** и **Rest**, связывая **String1** с результатом.
- Последовательность знаков является лексемой, если она представляет собой:
- имя в соответствии с синтаксисом ***Пролога***;
- число (предшествующий ему знак является отдельной лексемой);
- отличный от пробела знак.
- Предикат **fronttoken** отлично приспособлен для разбиения строки на лексические символы.

Задача 3: разбить предложение на список слов.

```
domains
  namelist=name*
  name=symbol
predicates
  string_namelist(string,namelist)
clauses
  string_namelist(S,[H |T]):-
          fronttoken(S,H,S1),!,
          string_namelist(S1,T).
  string_namelist(_, []).
goal
  string_namelist("Мама Дама Сама Коля",X),write(X).
```

Предикат frontstr/4

- Предикат **frontstr** расщепляет **String1** на две части. Синтаксис предиката:
- **frontstr(NumberOfChars,String1,StartStr ,EndStr)** ***% (i,i,o,o)***
- где **StartStr** содержит **NumberOfChars** п ервых символов из **String1**, а **EndStr** содержит остаток. При обращении к **frontstr** первые два параметра должны быть связанными, а последние два - свободными.

Предикат concat/3

- Предикат **concat** устанавливает, что строка **String3** является результатом сцепления **String1** и **String2**. Он имеет форму:
- **concat(String1,Sring2,String3)** ***% (i,i,o), (i,o,i), (o,i,i), (i,i,i)***
- По крайней мере два параметра должны быть связаны перед тем, как вы вызываете **concat**; это означает, что этот предикат всегда дает только одно решение (другими словами, он - детерминированный). Например, мы вызываем
- **concat("croco","dile",In_a_while)**
- связывая **In_a_while** с **crocodile**.

Предикат str_len/2

- Предикат **str_len** решает следующие задачи: определяет или проверяет длину строки или возвращает строку пробелов заданной длины. Он имеет формат:
- **str_len(StringArg,Length)** ***% (i,o), (i,i), (o,i)***
- Предикат **str_len** связывает переменную **Length** с длиной строки **StringArg** или проверяет, имеет ли **StringArg**данную длину **Length**. **Length** - это беззнаковое целое. В версии предиката с третьим потоком **str_len** возвращает строку пробелов данной длины, что может быть использовано для распределения буферов и других операций.

Предикат isname/1

- Предикат **isname** проверяет, является ли аргумент допустимым именем согласно синтаксису ***Пролога***, и имеет формат:
- **isname(String)** ***% (i)***
- Имя начинается с буквы алфавита или символа подчеркивания, за которым следует любое число букв, цифр и символов подчеркивания. Предыдущие и последующие пробелы игнорируются.

Предикаты преобразования типов

Предикат char_int/2

- Предикат **char_int** преобразует символ в целое число или целое в символ и имеет формат:
- **char_int(Char,Integer)** ***% (i,o), (o,i), (i,i)***
- Если оба аргумента связаны, то **char_int** проверяет, соответствуют ли значения аргументов. Если один аргумент связан, а другой свободен, **char_int** выполняет преобразование и связывает выходной параметр с преобразованной формой входной параметра.

Предикат str_char/2

- Предикат **str_char** преобразует строку, содержащую один и только один символ, в символ или символ в строку из одного символа; предикат имеет формат:
- **str_char(String,Char)** ***% (i,o), (o,i), (i,i)***
- В случае если поток параметров - **(i,i)**, то предикат **str_char** завершается успешно, если при этом **String** связан со строкой из одного символа, который соответствует **Char**. Если длина строки - не единица, то **str_char** завершается неуспешно.

Предикат str_int/2

- Предикат **str_int** преобразует строку, содержащую целое число, в его текстовое представление и имеет формат:
- **str_int(String,Integer)** ***% (i,o), (o,i), (i,i)***
- В случае если поток параметров - **(i,i)**, то **str_int** завершается успешно, при условии, что **Integer** связан с целым эквивалентом числа, представленного с помощью **String**.

Предикат str_real/2

- Предикат **str_real** преобразует строку в вещественное число или вещественное число в строку и имеет формат:
  - **str_real(String,Real)** ***% (i,o), (o,i), (i,i)***
- В случае если поток параметров - **(i,i)**, то **str_real** завершается успешно, если **Real** связан с вещественным числом, равным числу, представленному в **String**.

Предикат upper_lower/2

- Предикат **upper_lower** преобразует строку, все символы (или часть символов) которой являются символами верхнего регистра, в строку соответствующих символов нижнего регистра, и наоборот. Формат предиката:
  - **upper_lower(Upper,Lower)** ***% (i,o), (o,i), (i,i)***

- Задача 4: определить предикат **scanner**, который преобразует строку в список лексем. Лексемы классифицируются с помощью связывания функтора с каждой лексемой. В этом примере используются предикаты **isname, str_int, str_char** для определения природы лексемы, полученной с помощью **fronttoken**.

```
domains
   tok=numb(integer); name(string); char(char)
   toklist=tok*
predicates
   scanner(string,toklist)
   maketok(string,tok)
clauses
   scanner("",[ ]).
   scanner(Str,[Tok |Rest]):-
           fronttoken(Str,Sym,Str1),
           maketok(Sym,Tok),
           scanner(Str1,Rest).

   maketok(S,name(S)):-isname(S).
   maketok(S,numb(N)):-str_int(S,N).
   maketok(S,char(C)):-str_char(S,C).
goal
   write("Введите текст:"),nl,
   readln(Text),nl,
   scanner(Text,T_List),
   write(T_List).
```

Динамическая база фактов

- Внутренняя база фактов состоит из фактов, которые вы можете непосредственно добавлять и удалять из вашей программы на ***Прологе*** во время ее исполнения. Вы можете объявлять предикаты, описывающие внутреннюю базу данных в разделе **database** программы и применять эти предикаты таким же образом, как используются предикаты, описанные в разделе **predicates**.
- Для добавления новых фактов в базу данных в ***Прологе*** используются предикаты **insert, asserta, assertz**, а предикаты **retract** и **retractall** служат для удаления существующих фактов. Вы можете изменить содержание вашей базы фактов, сначала удалив факт, а потом вставив новую версию этого факта (или совершенно другой факт). Предикаты **consult/1** и **consult/2** считывают факты из файла и добавляют их к внутренней базе данных, а **save/1** и **save/2** сохраняют содержимое внутренней базы фактов в файле.
- ***Пролог*** интерпретирует факты, принадлежащие к базе данных, таким же образом, как обычные предикаты. Факты предикатов внутренней базы фактов хранятся в таблице, которую можно легко изменять, тогда как обычные предикаты для достижения максимальной скорости компилируются в двоичный код.

Объявление внутренней базы фактов

- Ключевое слово **database** определяет начало объявления раздела **database**.
  Раздел **database** состоит из последовательности объявлений предикатов, описывающих соответствующую внутреннюю базу фактов.

```
domains
    name,address=string
    age=integer
    gender=male; female
database
    person(name,address,age,gender)
predicates
    male(name,address,age)
    female(name,address,age)
    child(name,age,gender)
clauses
    male(Name,Address,Age):-
        person(Name,Address,Age,male).
        ...
```

Доступ к внутренней базе фактов

- Предикаты, принадлежащие внутренней базе фактов, доступны точно так же, как и другие предикаты. Единственное видимое различие состоит в том, что объявления таких предикатов расположены в разделе **database** вместо раздела **predicates**.

- По своей природе предикаты в разделе **database** всегда недетерминированные. Так как факты могут быть добавлены в любой момент во время выполнения программы, компилятор всегда должен учитывать, что существует возможность найти альтернативные решения в ходе поиска с возвратом.

Пример использования базы

```
domains
   name=string
   sex=char
database
   person(name,sex)
predicates
   genchin
clauses
   person("Маша",'Ж').
   person("Николай",'М').
   person("Виктор",'М').
   person("Анна",'Ж').
   genchin:-person(X,'Ж'),write(X),nl,fail.
goal
   clearwindow,genchin.
```

Удаление фактов во время выполнения программы

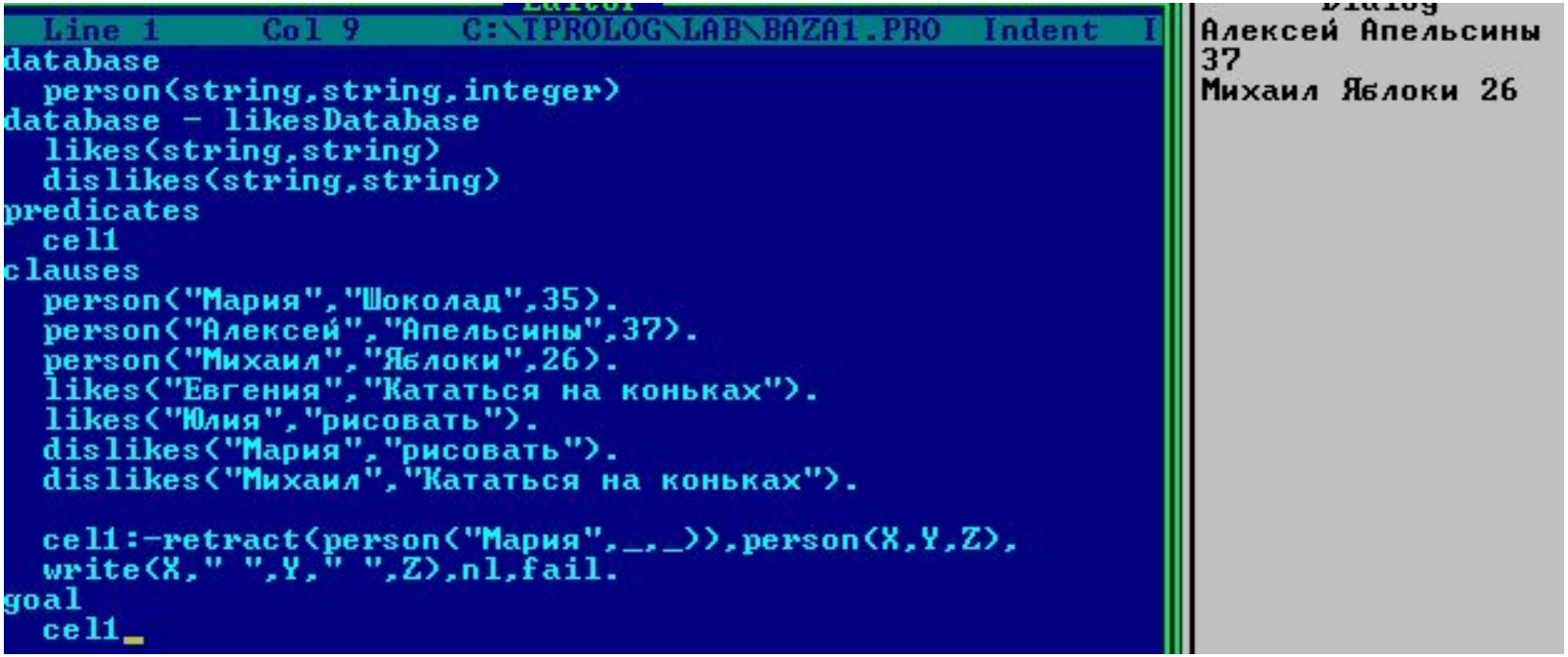
- Предикат **retract** унифицирует факты и удаляет их из внутренней базы фактов. Он имеет следующий формат:
  - **retract(the fact)** ***% (i)***
  - **retract(the fact,databaseName)** ***% (i,i)***

- Предикат **retract** удаляет первый факт из базы данных, который совпадает с фактом **the fact.**

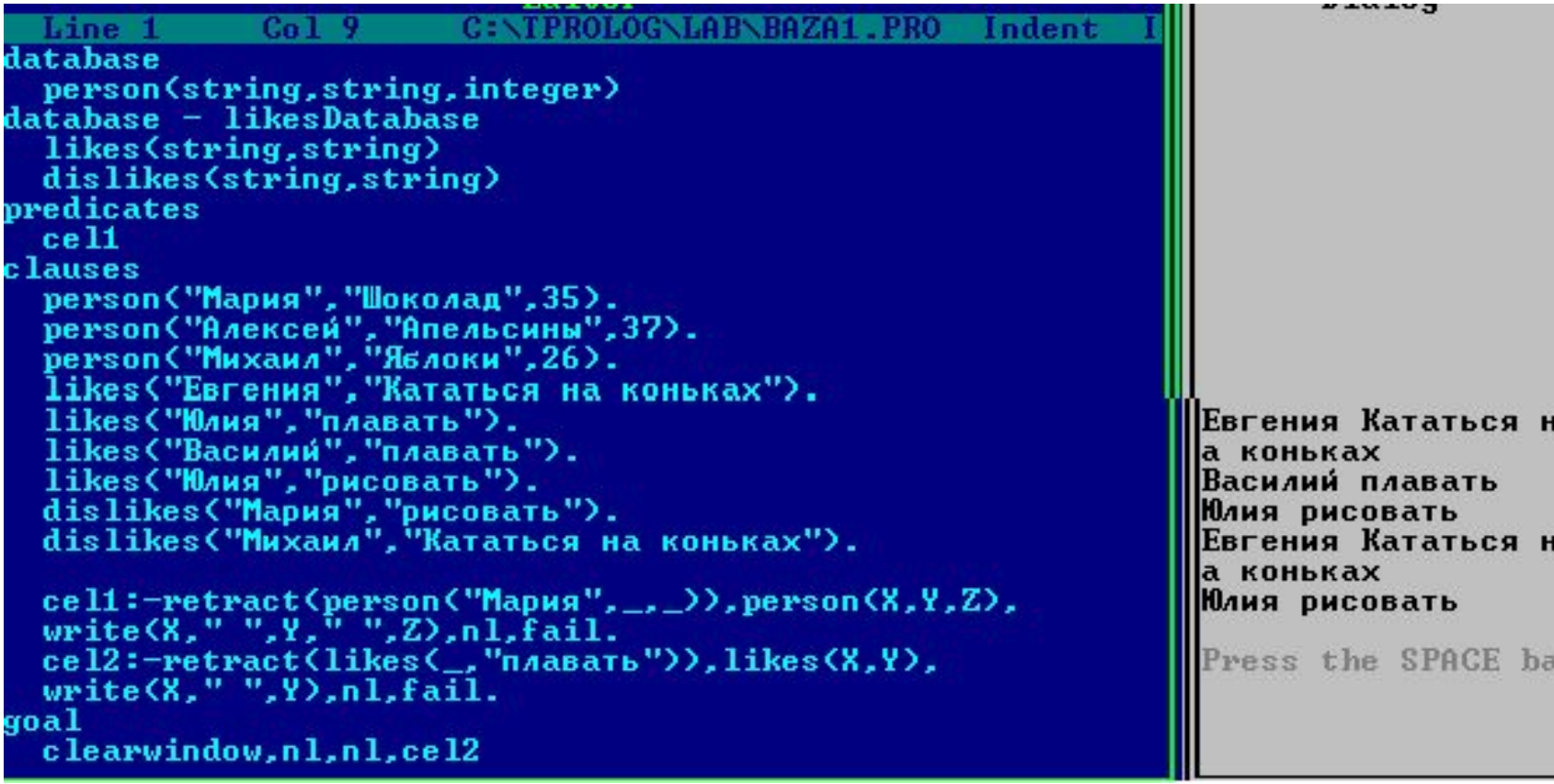
Удаление нескольких фактов сразу

- Предикат **retractall** удаляет из вашей базы фактов все факты, сопадающие с образцом **the fact**. Предикат **retractall**имеет следующий формат:
  - **retractall(the fact)**
  - **retractall(the fact,databaseName)**
- Следующая цель удаляет все факты о мужчинах из базы фактов с фактами **person**:
  - **retractall(person(_,_,_,male)).**
- Следующая цель удаляет все факты из базы **mydatabase**.
  - **retractall(_,mydatabase).**

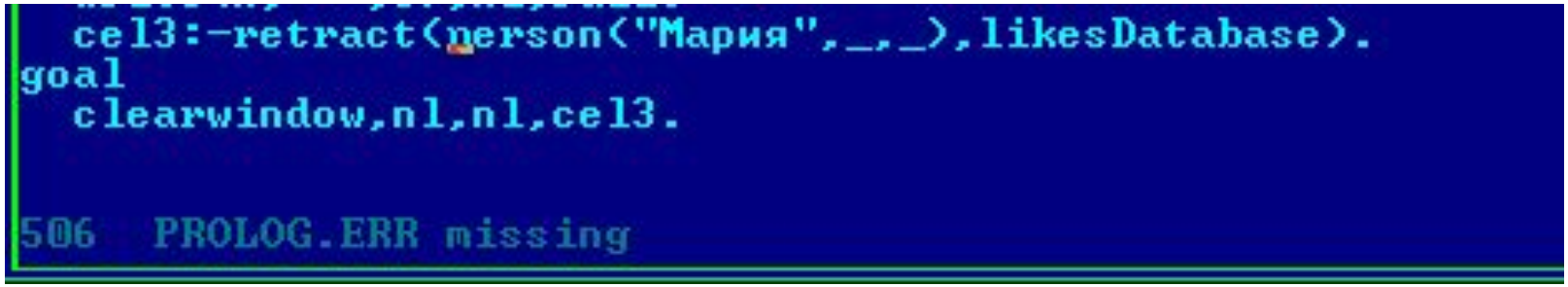
Пример удаления фактов из базы



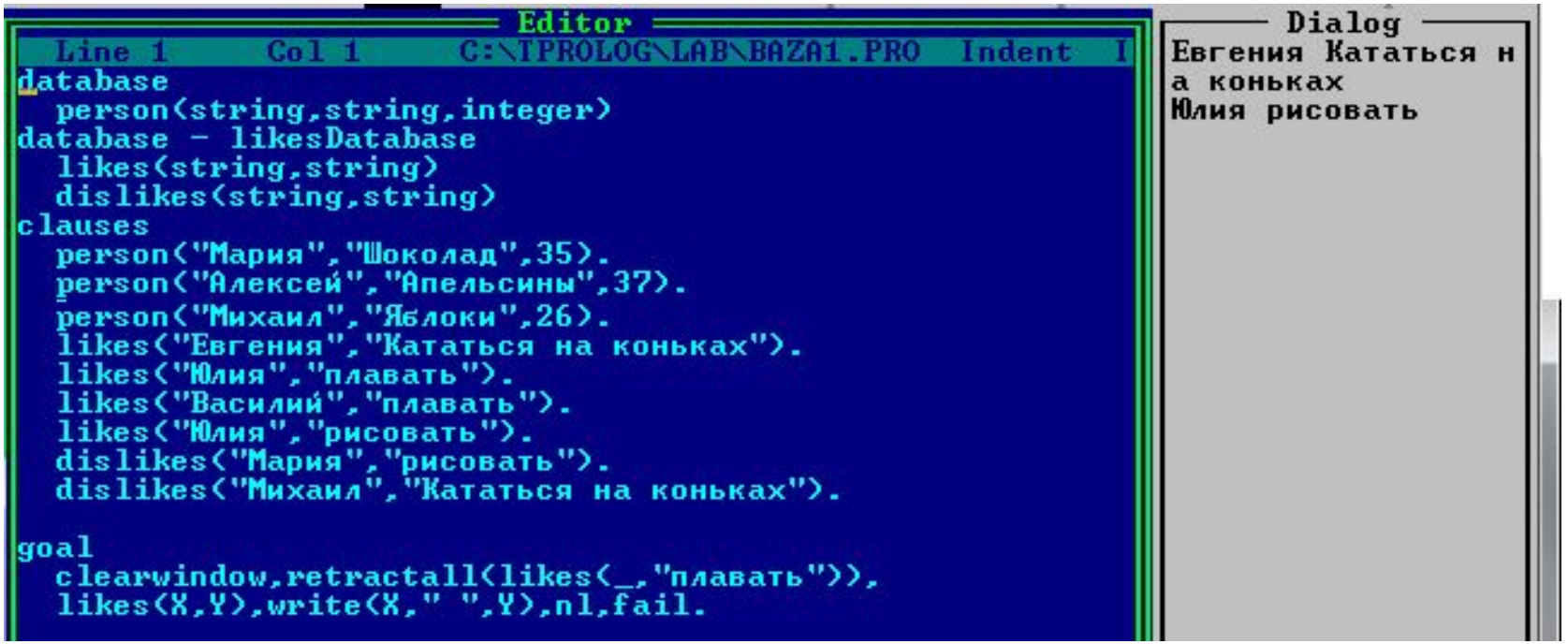
Обратите внимание, какая запись удалилась из базы!



Задайте указанную цель. Ответьте на вопрос: почему Пролог выдал ошибку?



Удаление всех записей из базы тех людей, которые любят плавать



Занесение фактов во время выполнения программы

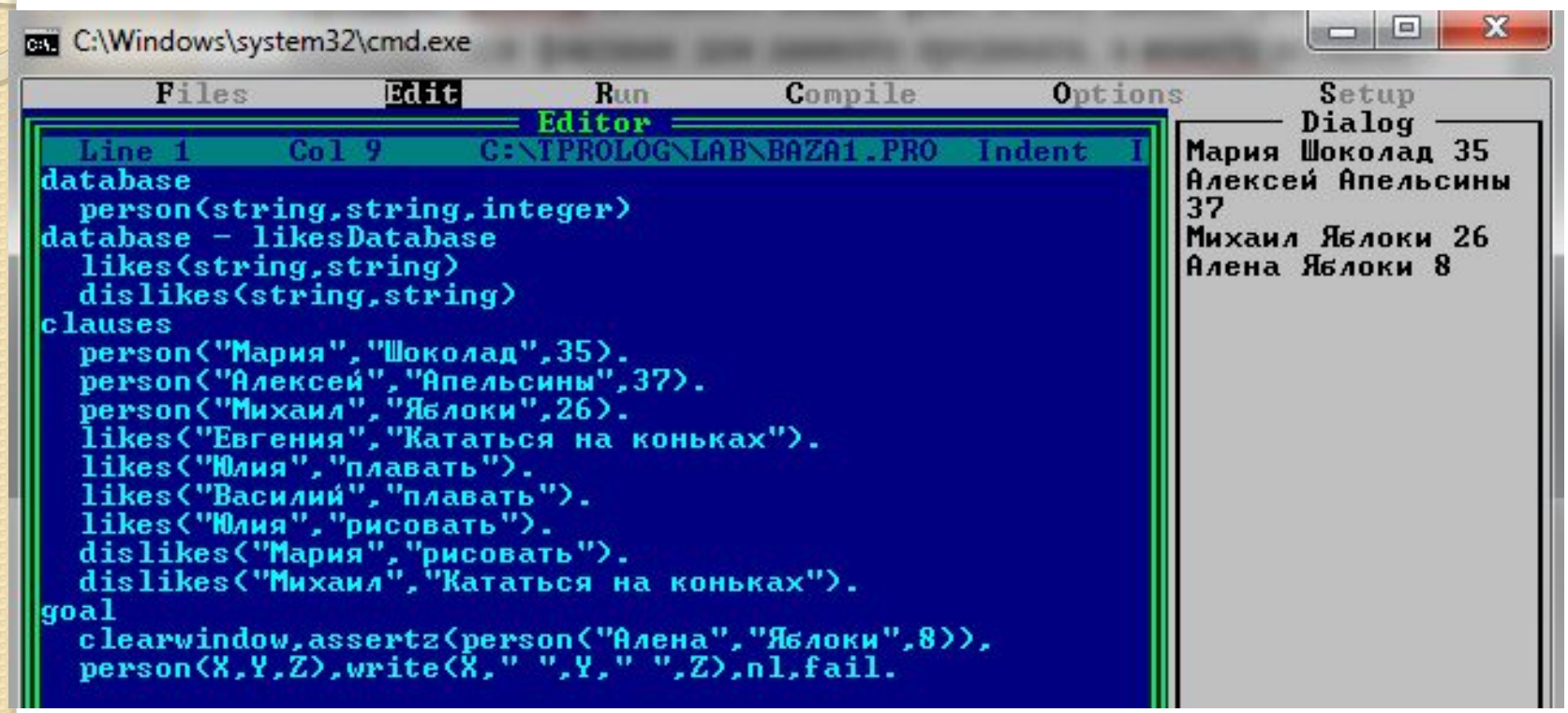
- Во время выполнения факты могут быть добавлены во внутреннюю базу данных фактов посредством предикатов:
  assert, **asserta** и **assertz**, или путем загрузки фактов из файла с помощью **consult**.
- Предикат **asserta** вставляет новый факт в базу данных фактов перед имеющимися фактами для данного предиката, а **assertz** вставляет факты после имеющихся фактов данного предиката. Использование предиката **assert** дает результат, аналогичный использованию **assertz**.
-

```
asserta(the fact) % (i)
asserta(the fact,facts_sectionName) % (i,i)
assertz(the fact) % (i)
assertz(the fact,facts_sectionName) % (i,i)
assert(the fact) % (i)
assert(the fact,facts_sectionName) % (i,i)
```

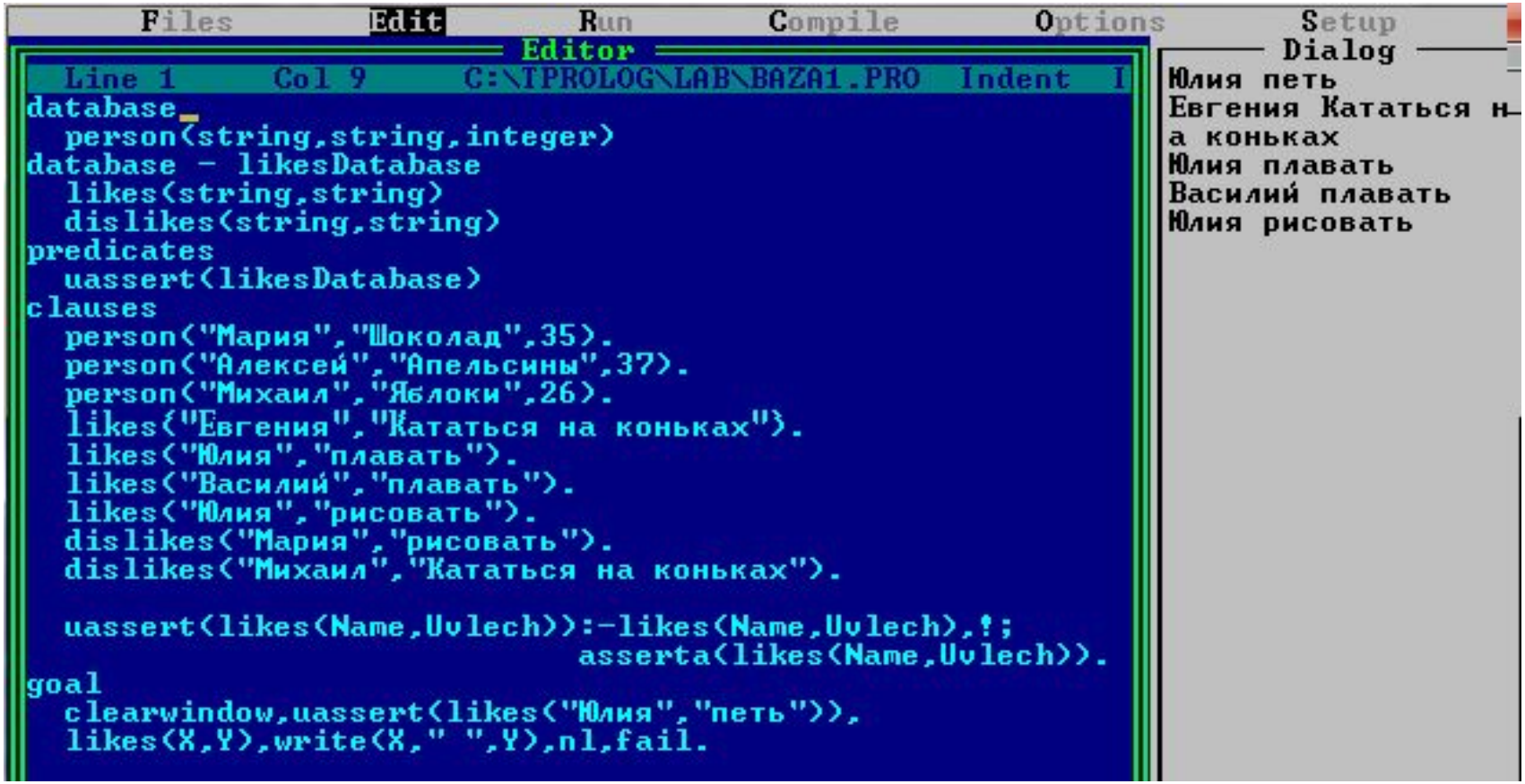
Проверка уникальности записи перед ее занесением

```
database - people
    person(string,string)
predicates
    uassert(people)
clauses
    uassert(person(Name,Address)):-
        person(Name,Address) ,
        !,
        ; % OR
        assert(person(Name,Address)).
```

Пример занесения фактов в конец базы данных без проверки уникальности



Пример занесения фактов в начало базы данных с проверкой уникальности



Создание своей базы

- Мы будем хранить информацию о футболистах.
- Программа должна позволять записывать данные в базу, удалять данные из базы, выводить данные на экран и выходить из системы.

Данные о футболистах

Имя игрока	Команда	Номер	Позиция	Рост	Вес	Кол-во
Денис Глушаков	Спартак	8	пз	182	82	46
Арсений Логашов	Локомотив	15	з	183	73	1
Сергей Нарубин	Динамо	42	в	196	93	0
Роман Шишкин	Локомотив	49	з	176	73	12
Роман Павлюченко	Урал	9	н	188	84	50

Описание предиката

- Для работы необходим предикат, кодирующий эту информацию. Подходящим является:
- **player**(p_name, /* *полное имя игрока (string)* */
- t_name, /**название команды (string)* */
- p_number, /**номер игрока (integer)* */
- pos, /**позиция игрока (string)* */
- height, /**рост (string)* */ weight, /*
- *вес (integer)* */
- nfl_exp) /**стаж выступлений за сборную РФ (integer)* */

```
domains
   p_name, t_name, pos, height=string
   p_number, weight, nfl_exp=integer
database
   dplayer(p_name,t_name,p_number,pos,height,weight,nfl_exp)
predicates
   repeat
   do_mbase
   assert_database
   menu
   process(integer)
   clear_database
   player(p_name,t_name,p_number,pos,height,weight,nfl_exp)
   error
goal
   do_mbase.
clauses
   repeat.
   repeat:-
      repeat.
   player("Денис Глушаков","Спартак",8,"пз","182",82,46).
   player("Арсений Логашов","Локомотив",15,"з","183",73,1).
   player("Сергей Нарубин","Динамо",42,"в","196",93,0).
   player("Роман Пишкин","Локомотив",49,"з","176",73,12).
   player("Роман Павлюченко","Урал",9,"н","188",84,50).
   assert_database:-
      player(P_name,T_name,P_number,Pos,Ht,Wt,Exp),
      assertz(dplayer(P_name,T_name,P_number,Pos,Ht,Wt,Exp)),
      fail.
   assert_database:-!.
   clear_database:-
      retract(dplayer(_,_,_,_,_,_,_)),fail.
   clear_database:-!.
   do_mbase:-
      assert_database,
      makewindow(1,7,7,"Футбольная база данных",0,0,25,80),
```

```
    makewindow(1,7,7,"Футбольная база данных",0,0,25,80),
    menu,
    clear_database.
menu:-
    repeat,
    clearwindow,
    write(" * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *"),nl,
    write(" *                                                               * "),nl,
    write(" * 1. Занесение новой информации об игроках                      * "),nl,
    write(" * 2. Удаление данных об игроках                                 * "),nl,
    write(" * 3. Просмотр информации                                        * "),nl,
    write(" * 4. Выход из программы                                         * "),nl,
    write(" *                                                               * "),nl,
    write(" * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *"),nl,
    nl,
    write(" Выберите номер: 1, 2, 3 или 4 : "),
    readint(Choice),nl,
    process(Choice),
    Choice = 4,
    !.
process(1):-
    makewindow(2,7,7,"Добавить данные в базу",2,20,18,58),
    shiftwindow(2),
    write("Имя игрока: "),
    readln(P_name),
    write("Команда: "),
    readln(T_name),
    write("Номер: "),
    readint(P_number),
    write("Амплуа: "),
    readln(Pos),
    write("Рост: "),
    readln(Ht),
    write("Вес: "),
    readint(Wt),
    write("Количество игр: "),
```

2

```
    write("Количество игр: "),
    readint(Exp),
    assertz(dplayer(P_name,T_name,P_number,Pos,Ht,Wt,Exp)),
    write(P_name," помещен в базу данных."),
    nl,!,
    write("Нажмите пробел. "),
    readchar(_),
    removewindow.
process(2):-
    makewindow(3,7,7,"Удаление из базы",10,30,7,40),
    shiftwindow(3),
    write("Задайте имя для удаления: "),
    readln(P_name),
    retract(dplayer(P_name,_,_,_,_,_,_)),
    write(P_name," был успешно удален из базы."), nl, !,
    write("Нажмите пробел."),
    readchar(_),
    removewindow.
process(3):-
    makewindow(4,7,7," Просмотр информации",7,30,16,47),
    shiftwindow(4),
    write("Задайте имя игрока: "),
    readln(P_name),
    dplayer(P_name,T_name,P_number,Pos,Ht,Wt,Exp),nl,
    nl,write(" Имя:         ",P_name),
    nl,write(" Команда:",T_name),
    nl,write(" Амплуа:",Pos),
    nl,write(" Номер:",P_number),
    nl,write(" Рост:",Ht," см"),
    nl,write(" Вес:",Wt," кг "),
    nl,write(" Количество игр:",Exp),
    nl,nl,!,
    nl,write("Нажмите пробел"),
    readchar(_),
    removewindow.
process(3):-
```

3

```
process(3):-
    makewindow(5,7,7," Ошибка ",14,7,5,60),
    shiftwindow(5),
    write("Такого игрока нет в базе данных."),nl,
    nl,!,
    write("Нажмите пробел."),
    readchar(_),
    removewindow,
    shiftwindow(1).
process(4):-
    write("Закончить работу с программой? (д/н)"),
    readln(Answer),
    frontchar(Answer,'д',_), !.
process(Choice):-
    Choice<1,error.
process(Choice):-
    Choice>4,error.
error:-
    write("Укажите число от 1 до 4."),
    write("(Нажмите пробел для продолжения)"),
    readchar(_).
```

4

Считывание фактов из файла

- Предикат **consult** считывает из файла **fileName** факты, описанные в разделе **database**, и вставляет их в вашу программу в конец соответствующей базы фактов. Предикат **consult** имеет один или два аргумента:

```
consult(fileName) % (i)
consult(fileName,databaseName) % (i,i)
```

Создание экспертной системы

Определение, является ли лицо отцом по группе крови

Группа крови			
мать	*отец*	*ребенок*	*вероятность конфликта*
1	1	1	нет
1	2	1 или 2	50%
1	3	1 или 3	50%
1	4	2 или 3	100%
2	1	1 или 2	нет
2	2	1 или 2	нет
2	3	любая	50%
2	4	1 или 2 или 4	66%
3	1	1 или 3	нет
3	2	любая	25%
3	3	1 или 3	нет
3	4	1 или 3 или 4	66%
4	1	2 или 3	нет
4	2	1 или 2 или 4	нет
4	3	1 или 3 или 4	нет
4	4	2 или 3 или 4	нет

```
domains
        list=integer*
predicates
        rep
        f
        muttergruppe_bekannt(integer)
        fatergruppe_bekannt(integer)
        kindgruppe_bekannt(integer)
        alles_knappt(integer,integer)
        gruppen_bekannt(integer,integer,integer)
        base(integer,integer,integer)
        gutchar
        gut(integer)
        rechnung(integer,integer,integer)
        mitglied(integer,list)
        len(integer,list)
        link(list,list,list)
clauses

  f:-
    clearwindow,
      gruppen_bekannt(M,K,F),
        rechnung(M,F,K),
          readchar(_).

  gruppen_bekannt(M,K,F):-
            makewindow(1,30,30," Ввод исходных данных ",4,26,10,42),
              rep,
               clearwindow,
                muttergruppe_bekannt(M),
                  kindgruppe_bekannt(K),
                    alles_knappt(M,K),
                      fatergruppe_bekannt(F).

   muttergruppe_bekannt(Muttergruppe) :-
               nl,write(" Введите группу крови матери: "),
```

1

```
muttergruppe_bekannt(Muttergruppe) :-
           nl,write(" Введите группу крови матери: "),
            readint(Muttergruppe).

fatergruppe_bekannt(Fatergruppe) :-
           nl,write(" Введите группу крови отца: "),
            readint(Fatergruppe).

kindgruppe_bekannt(Kindgruppe) :-
          nl,write(" Введите группу крови ребенка: "),
           readint(Kindgruppe).

alles_knappt(Muttergruppe,Kindgruppe):-
          base(_,Muttergruppe,Kindgruppe);
           base(Muttergruppe,_,Kindgruppe).

alles_knappt(_,_):-
      makewindow(2,78,78," Гоп-стоп! ",15,16,5,62),
      gotowindow(2),
      nl,
    write(" Вы ввели противоречивые данные (или это не мать ребенка)."),
     nl,
       nl,
    write(" <Enter> -  повторить ввод, <Esc> - прервать программу: "),
    gutchar.

 gutchar:-
          readchar(Ch),
           char_int(Ch,X),
            gut(X).
 gut(X):-X=13,!,
     removewindow(),
       gotowindow(1),
         fail.
 gut(X):-X=27,exit.
```

2

```
gut(X):-X=27,exit.
gut(_):-gutchar.


rechnung(M,F,K):-
       findall(All,base(M,All,K),List),
       findall(All,base(All,M,K),List1),
         link(List,List1,Reslist),
          mitglied(F,Reslist),!,
           len(X,Reslist),
          makewindow(2,78,78," Поздравляем ! ",15,26,5,42),
          gotowindow(2),
          nl,write("     Это отец с вероятностью 1/"), write(X),
          write(".").

rechnung(_,_,_):-
        makewindow(2,78,78," Поздравляем ! ",15,26,5,42),
        gotowindow(2),nl,write("     Это точно не отец.").

mitglied(R,[R|_]):-!.
mitglied(R,[_|T]):-mitglied(R,T).

len(0,[]).
len(X,[_|T]):-len(Y,T),X=Y+1.

link([],L,L).
link([H|E],Z,D):-mitglied(H,Z),!,link(E,Z,D).
link([H|E],Z,[H|D]):-link(E,Z,D).

rep.
```

```
    rep.
    rep:-rep.

        base(1,1,1).
        base(1,2,1).
        base(1,2,2).
        base(1,3,1).
        base(1,3,3).
        base(1,4,2).
        base(1,4,3).

        base(2,2,1).
        base(2,2,2).
        base(2,3,1).
        base(2,3,2).
        base(2,3,3).
        base(2,3,4).
        base(2,4,1).
        base(2,4,2).
        base(2,4,4).

        base(3,3,1).
        base(3,3,3).
        base(3,4,1).
        base(3,4,3).
        base(3,4,4).

        base(4,4,2).
        base(4,4,3).
        base(4,4,4).

goal f
```

4

Тематика создания экспертной системы

Фамилии студентов	Тематика экспертной системы
Аслезов Юрий Вавилина Александра	Алгоритмы поиска.
Емельянова Екатерина	Алгоритмы сортировки.
Коренева Дарья Корнилов Вадим	Компьютерные игры.
Смирнов Дмитрий	Языки программирования.
Еремин Дмитрий	Компьютерные вирусы.
Сухачев Антон Харчук Юрий	Смартфоны.
Серышев Юрий	Операционные системы.
Чавкин Андрей	Микропроцессоры.
Ширшов Никита	Компьютерные сети.