# ЛАБОРАТОРНАЯ РАБОТА № 3
# СОЗДАНИЕ БАЗЫ ЗНАНИЙ В СРЕДЕ PROLOG

*Цель работы* – знакомство с основами логического программирования на языке Пролог, получение практических навыков работы с отношениями, создание баз знаний, объединяющих декларативные и процедурные знания.

## 1. ОСНОВЫ РАЗРАБОТКИ PROLOG ПРОГРАММ

### 1.1. Общие сведения о языке Prolog

Теоретической основой Prolog является раздел символьной логики, называемый исчислением предикатов. Предикат – это логическая функция, которая выражает некоторое отношение между своими аргументами и принимает значение "истина", если это отношение имеется, или "ложь", если оно отсутствует.

В отличие от процедурных языков, где программист должен описать процедуру решения шаг за шагом, при использовании Prolog достаточно описать задачу и основные правила ее решения. Он имеет ряд свойств, которыми не обладают процедурные языки, что делает его мощным средством логического программирования. К числу таких свойств относятся:

- механизм вывода с поиском и возвратом,
- встроенный механизм сопоставления с образцом
- простая, но выразительная структура данных с возможностью ее изменения.

Язык Prolog отличается единообразием программ и данных. Данные и программа лишь две различные точки зрения на объекты Prolog. В единой базе данных можно свободно создавать и уничтожать отдельные элементы. Поскольку не существует различия между программой и данными, то можно менять программу во время ее работы. Естественным методом программирования является рекурсия. Prolog является декларативным языком. Это означает, что имея необходимые факты и правила, он может использовать дедуктивные выводы для решения задач программирования. Prolog-программа содержит описание проблемы, которое представляет собой набор логических утверждений в форме фактов, таких как, например

```
Иван любит Марью.
Петр любит футбол.
```

или в форме правил, например,

```
Иван любит X, если Петр любит X
(Иван любит тоже самое, что любит и Петр).
```

На основе имеющихся правил и фактов Prolog делает выводы. Имея два вышеприведенных факта и одно правило, он придет к выводу, что «Иван любит футбол». А если задать запрос: «Кто любит футбол?», то он найдет все решения этой задачи. При этом он автоматически управляет решением задачи, стараясь во время выполнения программы найти все возможные наборы значений,

удовлетворяющие запросу. Поиск производится по всей базе данных, ранее введенной в систему.

Термин *база данных* используется при объединении набора фактов для совместного их использования при решении некоторой конкретной задачи. Если кроме фактов в описании предметной области содержатся еще и правила, то часто используют термин *база знаний*.

В языке Prolog используется *метод поиска с возвратом*, который позволяет ему в случае нахождения одного решения пересмотреть все сделанные предположения еще раз, чтобы определить, не приводит ли новое значение переменной к еще одному решению. Синтаксис Пролога очень короток и прост. Поэтому его проще освоить, чем синтаксис традиционных языков программирования.

### 1.2. Понятие факта, правила, запроса и процедуры

Программа на языке Prolog состоит из множества предложений. Каждое предложение может быть одного из трех типов: это либо факт, либо правило, либо запрос.

1) Факт – это предложение без условий, утверждение о том, что соблюдается некоторое отношение. Он записывается как имя предиката, за которым следует заключенный в скобки список аргументов. Каждый факт должен заканчиваться точкой. Например:

```
likes('Иван','Марья').
likes('Петр','футбол').
```

2) Правило – это факт, истинность которого зависит от истинности других фактов. Состоит из заголовка и тела, разделенных знаком : – , который читается как «если» и соответствует операции импликации. Заголовок правила – это предикат, а тело правила – последовательность предикатов, разделенных запятыми. Правило должно заканчиваться точкой, а запятая в теле правила означает конъюнкцию (логическое И). Например:

```
likes('Иван', X) :- likes('Петр', X).
```

Интуитивный смысл правила состоит в том, что цель, являющаяся заголовком, будет истинной, если Пролог сможет доказать, что все выражения (подцели) в теле правила являются истинными.

```
Заголовок :-  Подцель1, Подцель2, ... , ПодцельN.
```

Переменные, входящие в предикат заголовка правила, квантифицируются универсально. Это означает, что правило будет истинным для любых термов, которые удовлетворяют подцелям правила.

3) Запрос – это последовательность предикатов, разделенных запятыми или точкой с запятой и завершающаяся точкой. Как было указано ранее, запятая означает конъюнкцию (логическое И), точка с запятой обозначает дизъюнкцию (логическое ИЛИ). Предикат запроса называется целью. Простые запросы, не содержащие никаких переменных, допускают лишь два возможных ответа: "true" или "false". В случае ответа "true" говорят, что запрос завершился

успехом, цель достигнута. Использование переменных в запросах позволяет задавать более сложные вопросы. Например:

```
?- likes('Иван','футбол').
     true.
?- likes(Who,'Марья').
     Who = 'Иван'.
?- likes(Man,'футбол').
     Man = 'Петр' ; Man = 'Иван'.
?- likes(X,'Марья'),likes(X,'футбол').
     X = 'Иван'.
```

Последний из приведенных выше запросов на естественном языке можно интерпретировать как «Кто любит Марью и футбол?». С точки зрения логики предикатов – это составной (сложный) запрос.

4) Процедура – это несколько правил, заголовки которых содержат одинаковые предикаты. Так, например, два правила

```
max(X, Y, X) :- X >=Y.
max(X, Y, Y) :- X < Y.
```

реализуют процедуру нахождения наибольшего из двух чисел и используют одинаковый предикат `max` (число 1, число 2, результат).

### 1.3. Механизм сопоставления и поиска с возвратом

Особенность языка Prolog заключается в том, что он воспринимает в качестве программы некоторое описание задачи или базы знаний и сам производит логический вывод, а также поиск решения задач, используя при этом поиск с возвратом и унификацию.

*Унификация* представляет собой процесс сопоставления цели с фактами и правилами базы знаний. Цель может быть *согласована*, если она может быть сопоставлена с заголовком какого-либо предложения базы. Для этого предикат цели и предикат заголовка предложения должны иметь одинаковое имя и число аргументов. Рассмотрим этот процесс на примере запроса

```
?- likes(Man,'футбол').
```

При решении этой задачи Prolog ставит перед собой цель доказать истинность предиката `likes` и найти условия, при которых это возможно на имеющемся множестве фактов и правил. Приняв за цель предикат запроса, Prolog пытается найти в базе знаний предложения, в заголовке которых есть предикат `likes`.

Поиск подходящего для сопоставления предложения ведется с самого начала базы. Таким предложением будет первый факт `likes('Иван', 'Марья')`, имя предиката которого и его размерность совпадает с именем и арностью предиката цели.

```
?- likes(Man, 'футбол')       - исходная цель
likes('Иван', 'Марья')        - сопоставляемое предложение
Унификация: Man = 'Иван' и 'футбол' ≠ 'Марья'
Сопоставление:  неудача (fail)→ false
```

Проводя сопоставление этих двух предикатов, Prolog терпит неудачу при унификации аргументов этих предикатов. Если еще переменную Man он может унифицировать с константой 'Иван', то вторые аргументы предиката, а именно, две константы 'футбол' и 'Марья' ему унифицировать не удается.

Сопоставление цели с первым предложением базы знаний признается неудачным и Prolog откатывается к исходной цели. Он ищет в базе знаний новое предложение с заголовком likes для выполнения сопоставления. Таким предложением будет второй факт likes('Петр','футбол').

```
?- likes(Man, 'футбол')       - исходная цель
likes('Петр', 'футбол')       - сопоставляемое предложение
Унификация: Man = 'Петр' и 'футбол' = 'футбол'
Сопоставление:  истина → true
```

Это сопоставление Prolog признает успешным при условии, что переменная Man будет унифицирована символьной константой 'Петр'. Именно это значение и выводит Prolog как один из ответов на запрос к базе знаний.

Но Prolog еще не закончил просмотр всей базы знаний. Поэтому он выполняет откат к исходной цели и ищет еще какие-либо предложения для согласования. Таким предложением является правило

```
likes('Иван', X) :- likes('Петр', X).
```

которое можно согласовать с исходной целью при значениях переменных Man = 'Иван' и X='футбол' и при условии, что тело правило, то есть его правая часть будет истинной.

```
?- likes(Man, 'футбол')       - исходная цель
likes('Иван', X)              - сопоставляемое предложение
Унификация: Man = 'Иван' и 'футбол' = X
Сопоставление:  likes('Петр', X)
Новая цель:     likes('Петр', 'футбол')
```

Prolog в качестве новой цели принимает первую подцель тела правила, а именно предикат likes('Петр','футбол') и пытается доказать его истинность на множестве предложений базы знаний, начиная опять с первого факта. Весь рассмотренный выше процесс будет повторяться до тех пор, пока все возможные решения не будут найдены или процесс закончится неудачей. В системе Prolog для этого реализован метод решения, который называется поиском с возвратом.

### 1.4. Основные элементы языка Prolog

Аргументы предложений Prolog-программы называются термами, а саму Prolog-программу можно рассматривать как сеть отношений, существующих между термами. Каждый терм обозначает некоторый объект предметной области и записывается как последовательность литер, которые делятся на четыре категории: прописные буквы, строчные буквы, цифры и спецзнаки.

Стандартный Prolog распознает тип объекта по его синтаксической форме и не требует дополнительной информации (например, объявления типа данных)

для того, чтобы распознать тип объекта. В качестве основных объектов данных обычно выделяются константы, переменные (простые термы) и структуры (составные термы).

*Константа* – именованный конкретный объект или отношение. Существует два вида констант: атомы и числа. Атом – это либо последовательность латинских букв, цифр и знака подчеркивания, начинающаяся со строчной буквы, либо произвольная группа символов, заключенная в одинарные кавычки (иногда допускается использование и двойных кавычек). Примеры записи констант: ivan, ‘Ivan’, invoice_n, “иван”, ‘№_счета’, ‘Иван’.

Числа в Prolog бывают целыми (1, 1024, 0, -97) и вещественными (3.14, -0.0035, -0.5e4). Вещественные числа используются редко. Причина в том, что язык предназначен в первую очередь для обработки символьной, а не числовой информации. При символьной обработке используют целые числа, например, для подсчета количества элементов списка, а необходимость в вещественных числах очень невелика.

*Переменная* – обозначение объекта, значения которого меняются в ходе выполнения программы. В Prolog переменная имеет имя, начинающееся с прописной буквы или знака подчеркивания. Например: Name, X, Invoice_n. Переменная называется связанной, если имеется объект, который она обозначает. При отсутствии такого объекта переменная называется свободной. Для обозначения переменной, на которую отсутствует ссылка в программе, используется анонимная переменная, которая обозначается одиночным знаком подчеркивания «_». Область действия любой из переменных – это предложение. Одноименные переменные в разных предложениях могут иметь разные значения.

*Список* – объект, содержащий внутри себя произвольное число других объектов. Он задается перечислением элементов через запятую в квадратных скобках. Например:

```
[1,2,3,4]                  % список чисел
[a,b,c]                    % список констант
["Aa", "Bb", "Cc"]         % список строк
[1, a, "Aa", [22,33,44]]   % смешанный список
[]                         % пустой список
```

*Структура* (*составной терм*) – объект, состоящий из совокупности других объектов, которые называются компонентами. Составные термы аналогичны записям Паскаля или структурам Си. Они представляют собой определяемые программистами объекты произвольной сложности.

*Предикат* – логическая функция одного или нескольких аргументов, принимающая значение “истина” или “ложь”. Записывается в виде составного терма. Аргументы перечисляются через запятую и представляют собой какие-то объекты или свойства объектов, а имя предиката обозначает связь или отношение между аргументами. Предикат однозначно определяется парой: имя и количество аргументов. Количество аргументов предиката называют его арностью. Следует особо отметить, что имена предикатов записываются

строчными буквами и не допускают пробелов. Поэтому в именах предикатов в качестве разделителей можно использовать символ подчеркивания

## 2. СРЕДА РАЗРАБОТКИ TURBO PROLOG

Наиболее мощной современной реализацией языка Prolog является Visual Prolog, в который, по сравнению со стандартными реализациями добавлены средства объектно–ориентированного программирования, анонимные предикаты, императивные конструкции (foreach, if...then...else), коллекторы списков и многое другое. Visual Prolog является наследником системы Turbo Prolog, которая в конце 80-х и начале 90-х годов прошлого века являлась наиболее популярной, особенно в нашей стране, системой логического программирования. Turbo Prolog отличается от классического тем, что основан на строгой статической типизации. В отличие от других аналогов, он является компилируемым языком и отличается высокой скоростью трансляции и выполнения, что как раз и достигается за счет отхода в этой среде от классической реализации языка Prolog. К настоящему времени накоплен большой объем как технической, так и учебной информации, посвященной реализации Prolog-программ именно в этой среде. Кроме того, исходные коды программ, написанных на Turbo Prolog’е, могут импортироваться, загружаться и успешно работать также и в среде Visual Prolog. Тысячи примеров программ из различных областей человеческой деятельности, реализованных средствами Turbo Prolog, можно найти в Интернете. В этих условиях среда Turbo Prolog’а представляет интерес и в современных условиях. Особенно в учебных целях, как первая ступенька в изучении строго типизированного языка до перехода к его объектно-ориентированным расширениям.

### 2.1. Структура программы на Turbo Prolog

Программа на Turbo Prolog состоит из нескольких секций, каждая из которых идентифицируется ключевым словом и имеет следующую обобщенную структуру:

```
domains
      /* секция объявления доменов  */
database
      /* секция объявления динамических баз данных */
predicates
      /* секция объявления предикатов */
goal
      /* внутренняя цель */
clauses
      /* предложения (факты и правила) */
```

Обязательным в программе является наличие двух секций с именами `predicates` и `clauses`. В первой из них описываются структуры используемых в программе отношений, а во второй – эти отношения определяются, то есть декларируются. Turbo Prolog может иметь внутреннюю цель. В этом случае программа начинает выполняться с точкой входа в секции `goal`. Если внутренняя цель не задана, то в этом случае система работает в

интерактивном режиме и пользователь может формировать запросы к программе в окне Диалога из оболочки Turbo Prolog.

Факты и правила записываются в виде предложений, каждое из которых заканчивается точкой. Текст, заключенный в /* ... */ – это комментарий.

Рассмотрим более подробно назначение каждой из секций программы.

### 2.1.1. Секция domains

В этой секции объявляются любые нестандартные домены, которые используются в качестве аргументов предикатов. Домены в Turbo Prolog являются аналогами типов в других языках. Основными стандартными доменами Turbo Prolog'а являются:

- char – символ, заключенный в одиночные кавычки (например, 'a');
- integer – целое от -32768 до 32767 (переводится в вещественное автоматически, если необходимо);
- real – вещественное (например, -68.72, 6e-94, -791e+21);
- string – последовательность символов, заключенных в двойные кавычки (например, "Андрей Борисов");
- symbol – набор латинских букв, цифр и символов подчеркивания, где первый символ строчная буква (например, n_fax), либо заключенная в кавычки последовательность символов, которая начинается с большой буквы или имеет пробелы (например, "N fax", "футбол").

Кроме стандартных доменов можно использовать свои, определяемые пользователем, домены. Для этого в области объявления доменов могут быть использованы следующие форматы:

- name = stanDom, где stanDom – один из стандартных доменов: int, char, real, string или symbol, a name – одно или несколько имен доменов. Пример:

```
domains
   year, height = integer
   fio = symbol
```

- myList = elementDom*, где myList – область, состоящая из списков элементов из области elementDom, которая может быть определена пользователем или иметь стандартный тип.

```
domains
   numberlist = integer*
   letter = char*
```

- myCompDom = fun1($d_{11}$, ..., $d_{1n}$); fun2($d_{21}$, ..., $d_{2s}$); ... funM($d_{m1}$, ..., $d_{mq}$), где myCompDom – область, которая состоит из составных объектов, описываемых указанием функтора и областей для всех компонентов, а fun1, fun2, ... , funM – имена функторов. Правая часть этого описания может определять несколько альтернатив, разделенных символами «;».

```
domains
   packing = box(integer,integer,integer) ; bottle(integer)
```

Область `packing` соответствует одной из двух возможных структур `box` и `bottle`, которые различаются не только именами, но и количеством компонент.

### 2.1.2. Секция predicates

В секции predicates объявляются предикаты и типы аргументов этих предикатов. Имена предикатов должны начинаться со строчной латинской буквы, за которой следует последовательность букв, цифр и символов подчеркивания (до 250 знаков). В именах предикатов нельзя использовать символы пробела, минуса, звездочки, обратной (и прямой) черты. Объявление предикатов имеет форму:

```
predicates
   predicate_1 (domen_11, domen_12, ...,  domen_1m)
   predicate_2 (domen_21, domen_22, ...,  domen_2n)
```

Здесь domen_11 – либо стандартные домены, либо домены, объявленные в секции domains. Количество аргументов предиката определяют его размерность. Предикат может указываться только именем и не иметь аргументов. Обычно, имя предиката выбирается так, чтобы оно отражало определенный вид взаимосвязи между аргументами этого предиката.

```
predicates
   student(symbol,real)
   start
   good_student(symbol)
```

Например, student (symbol, integer) – предикат, связывающий фамилию студента и его средний балл.

Можно использовать несколько описаний одного и того же предиката. При этом все описания должны следовать одно за другим и должны иметь одно и то же число аргументов. Пусть требуется определить отношение между тремя аргументами, первые два из которых соответствуют слагаемым, а третий – сумме двух первых. Этот предикат может быть описан в следующем виде

```
predicates
   add(integer,integer,integer)
   add(real,real,real)
```

и позволит его аргументам принимать значения как из области целых, так и из области действительных чисел.

### 2.1.3. Секция clauses

В секции clauses размещаются факты и правила, с которыми будет работать Turbo Prolog, пытаясь разрешить цель программы. Факт – это утверждение о существовании некоторого отношения между аргументами, обозначаемого именем предиката. Представляют собой фразы без условий, содержат утверждения, которые всегда абсолютно верны. В отличие от фактов, правила содержат утверждения, истинность которых зависит от некоторых условий, образующих тело правила. Факты и правила могут быть записаны в несколько строк, но обязательно должны заканчиваться точкой. В общем случае формат секции clauses имеет вид:

```
clauses
   /* факты */
   predicate_1(term_11, term_12, ...,  term_1k).
   ...
```

```
    predicate_N(term_N1, term_N2, ...,  term_Ns).
    /* правила */
    goal_1 :- subGoal_11, subGoal_12, ... , subGoal_1k.
    ...
    goal_M :- subGoal_M1, subGoal_M2, ... , subGoal_Mn.
```

где:

- `predicate_K` – имена предикатов, которые до этого уже были описаны в секции predicates,
- `term_KJ` – аргументы предикатов (термы), количество которых должно соответствовать арности описания предиката,
- `goal_N` – заголовок правила, который имеет ту же форму, что и факт,
- `subGoal_N1`, ... , `subGoal_Nk` – тело правила, которое представляет собой список подцелей, разделенных запятыми (логическое И) или точкой с запятой (логическое ИЛИ).

Пример фактов и правил, определяющих отношения, заданные описанными в секции predicates предикатами `student` и `good_student`, может иметь вид:

```
clauses
    /* факты */
    student("Петров",4.5).
    student("Сидоров",3.75).
    /* правила */
    good_student(Name):- student(Name,B),B > 4.
```

Для того, чтобы заголовок правила был истинным, надо чтобы каждая подцель, входящая в тело правила, была истинной.

### 2.1.4. Секция goal

В секции goal задается внутренняя цель программы, что позволяет программе запускаться независимо от среды разработки. Если внутренняя цель включена в программу, то Turbo Prolog выполняет поиск только одного первого решения, и связываемые с переменными значения не выводятся на экран. Если внутренняя цель не используется, то в процессе работы есть возможность вводить в диалоговом окне внешнюю цель. При использовании внешней цели Turbo Prolog ищет все решения и выводит на экран все значения, связываемые с переменными. В систему Turbo Prolog включено более 200 встроенных стандартных предикатов и доменов. В случае использования этих предикатов и доменов нет необходимости объявлять их в программе.

Рассмотрим пример программы, в которой задана внутренняя цель и используется обращение к стандартным предикатам:

```
predicates
    hello
goal
    hello.
clauses
    hello:- write("Введите свое имя"),nl,
            readln(Name),nl,
            write("Привет ,",Name).
```

В этой программе предикат readln запрашивает ввод с клавиатуры значения некоторой переменной (Name), после ввода которой предикат write выводит на экран приветствие вместе со значением этой переменной. Однако чаще всего целью является сложный запрос к программе.

### 2.1.5. Секция database

Ключевое слово database указывает на начало последовательности описаний предикатов динамической базы данных (ДБД). ДБД – это база, в которую факты добавляются во время исполнения программы, ее иногда еще называют внутренней или оперативной базой данных.

При разработке и работе программ часто возникает необходимость в хранении информации. В процедурных языках это реализуется посредством использования, например, глобальных переменных. В языке Prolog подобное моделируется с помощью внутренней базы данных, которая и описывается в секции database.

Требования к описанию предикатов в этой секции, как и синтаксису их объявления, точно такие же, что и в секции predicates. Объявленные здесь предикаты не должны объявляться в секции predicates, но дизъюнкты этих предикатов могут присутствовать в секции clauses.

Факты, принадлежащие ДБД, обрабатываются отличным от обычных предикатов образом для того, чтобы ускорить работу с базой большого объема. Факты динамической базы могут модифицироваться в течение сеанса работы, добавляться и удаляться с помощью предикатов assert и restract, а также загружаться из файла с помощью стандартного предиката consult или записываться в файл предикатом save.

## 2.2. Оболочка системы Turbo Prolog

Для работы с системой достаточно запустить на выполнение файл prolog.exe. На экране дисплея появится заставка с указанием текущей конфигурации системы. Нажав на клавишу пробел, вы попадете в оболочку системы. На экране (рис.1) отображается главное меню системы и четыре системных окна: редактирования, диалога, сообщений и трассировки. Эти окна могут быть использованы в любой конфигурации, и любое из них может занимать весь экран или его часть.

Нижняя строка экрана содержит сообщения о состоянии системы, описывая доступные команды и назначение функциональных клавиш. Назначение клавиш меняется при изменении режима работы. Главное меню содержит набор команд и подчиненных иерархических меню. Далее рассмотрим опции главного меню системы и дадим краткое описание их назначения.

**Опция меню "Файлы” (Files)**

Выбор этой команды главного меню приводит к тому, что на экран дисплея выводится контекстное меню по работе с каталогами и файлами. Основные режимы этого меню по действию аналогичны этим режимам в других системах.

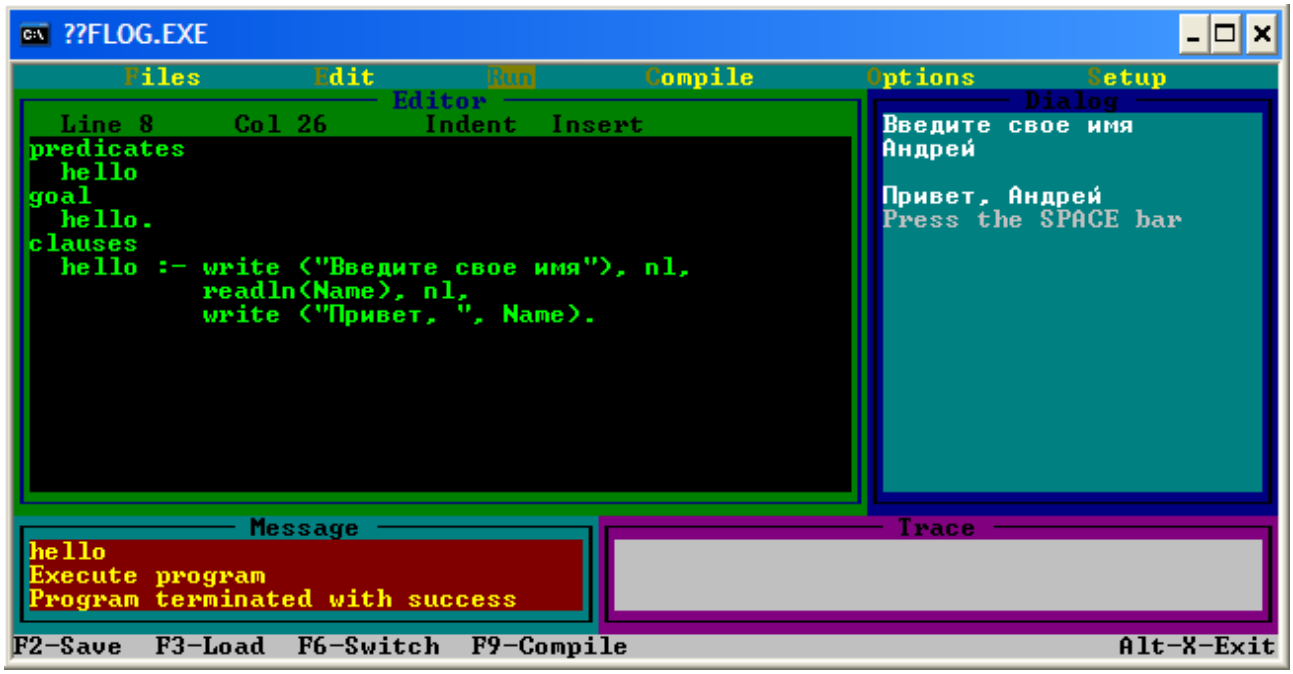



Рис.1 Интерфейс Turbo Prolog

Отметим только некоторые из них. Так, например, режим "Каталог" используется для выбора рабочего каталога. В частности каталог с именем PRO установлен по умолчанию, а для остальных следует указать путь к рабочему каталогу. При выборе команды "Загрузить" система запрашивает имя файла. При этом можно ввести любое доступное для DOS имя файла. Если расширение в имени файла опущено, то система автоматически добавляет расширение *.pro. Если на сообщение системы "Имя файла:" будет нажата клавиша Enter, то отобразится содержимое текущего каталога и выбор файла надо выполнить клавишами управления курсором. Режим "Переход к ДОС" вызывает временный выход в DOS, возврат из операционной системы в Turbo Prolog возможен после ввода команды exit.

**Опция меню "Редактировать" (Edit)**

По этой команде вызывается встроенный текстовый редактор, который позволяет вводить и редактировать текст программы. Методы работы с редактором такие же, как и с обычным текстовым редактором. Перечень основных команд и комбинаций клавиш для вызова этих команд можно получить, нажав клавишу F1. Особенностью встроенного редактора является наличие в его составе дополнительного окна, позволяющего одновременно работать с двумя файлами и обмениваться между ними блоками информации: копировать и переносить блоки программного кода из одного файла в другой.

**Опция меню "Выполнить" (Run)**

Эта команда используется для выполнения откомпилированной программы, находящейся в памяти. При этом возможны две ситуации:

1) Если цель содержится внутри программы, то есть в программе присутствует секция goal, то после выбора команды Run результат работы программы будет представлен в окне Dialog. Нажатие на клавишу пробел вызовет возврат в главное меню.

2) Если секция goal в программе отсутствует, то после выполнения команды Run активизируется окно Dialog, в котором пользователь может вводить запросы в интерактивном режиме.

В ходе выполнения программы некоторые из функциональных клавиш имеют специальное назначение:

F8 – повторный ввод предыдущего запроса в окне Dialog;

F9 – вызов редактора;

Shift+F9 – выбор системного окна для изменения его размеров;

Shift+F10 – изменяет размеры или двигает окно;

Ctrl+F10 – окно на весь экран / стандартный размер;

Ctrl+S – остановка вывода на экран / продолжение вывода;

Ctrl+C или Ctrl+Break – прерывание исполнение программы.

**Опция меню "Компилировать" (Compile)**

По этой команде выполняется компилирование программы, загруженной в окне редактора. Результат будет сохранен либо в памяти, либо на диске в виде *.OBJ или *.EXE файла, в зависимости от установки переключателя компиляции в меню "Options".

## 2.3. Отладка и трассировка программ

Любая программа после выбора в главном меню команды "Выполнить" (Run) запускается на компиляцию и выполнение. При этом система проверяет программу на соответствие синтаксису, отсутствие смешения значений из разных областей типов данных.

Если при компиляции обнаруживается ошибка, то соответствующее сообщение появляется в нижней строке окна редактора и курсор в окне редактирования указывает на место ошибки. Если программа успешно откомпилирована и система ожидает задание внешней цели для выполнения, то следующей стадией отладки является выбор таких запросов, чтобы программа тестировалась для достаточно широкого набора исходных данных.

При возникновении непредвиденных ситуаций следует перейти на пошаговую трассировку программы. Для этого в текст программы вводится директива trace. При выполнении программы, содержащей директиву trace, в окне трассировки по очереди выводятся все цели и утверждения программы, используемые в процессе ее выполнения. Это позволяет следить за ходом выполнения программы, переходами от одной цели к другой, за процессом поиска с возвратом. Для пошагового перехода от выполнения одной цели к другой следует нажимать клавишу F10.

## 3. СОЗДАНИЕ ПРОСТЕЙШИХ ПРОГРАММ

### 3.1. Задание родственных отношений

В данной части лабораторной работы требуется определить бинарные отношения родства, заданные на множестве людей (рис.2). Здесь Мария и Иван являются родителями Ани и Сережи. Стрелка направлена от родителя к ребенку, пунктиром изображен факт брака между супругами.

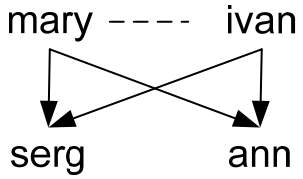



Рис.2 Генеалогическое дерево

В качестве базовых отношений для описания членов семьи будем использовать parent (рус.: родитель), married (рус.: женат), man (рус.: мужчина), woman (рус.: женщина):

- бинарное отношение parent (имя родителя, имя ребенка);
- бинарное отношение married (имя мужа, имя жены);
- унарное отношение man (имя мужчины);
- унарное отношение woman (имя женщины).

Все базовые отношения задаются в форме фактов. Имена членов семьи можно записывать различным образом: `mary`, `"Mary"`, `"Мария"`. С заглавной буквы начинаются только имена переменных языка Prolog, поэтому используйте двойные кавычки.

Используя предикаты parent (symbol, symbol), man (symbol), woman (symbol), married (symbol, symbol), запишем факты, описывающие представленную семью.

```
predicates
     man(symbol).
     woman(symbol).
     parent(symbol,symbol).
     married (symbol, symbol).
clauses
     man(ivan).              /* ivan – мужчина */
     man(serg).              /* serg – мужчина */
     woman(mary).            /* mary – женщина */
     woman(ann).             /* ann – женщина */
     married(ivan,mary).     /* ivan женат на mary */
     parent(ivan,ann).       /* ivan – родитель ann */
     parent(mary,ann).       /* mary – родитель ann */
     parent(ivan,serg).      /* ivan – родитель serg */
     parent(mary,serg).      /* mary – родитель serg */
```

### 3.2. Загрузка системы Turbo Prolog, ввод и запуск программы

До начала работы с Prolog-программами следует загрузить систему Turbo Prolog.

Первым этапом при работе с Prolog-программами является ввод их исходного текста, который выполняется в режиме редактирования (рис.3). Для перехода в этот режим следует использовать опцию Edit (Ред) главного меню системы.

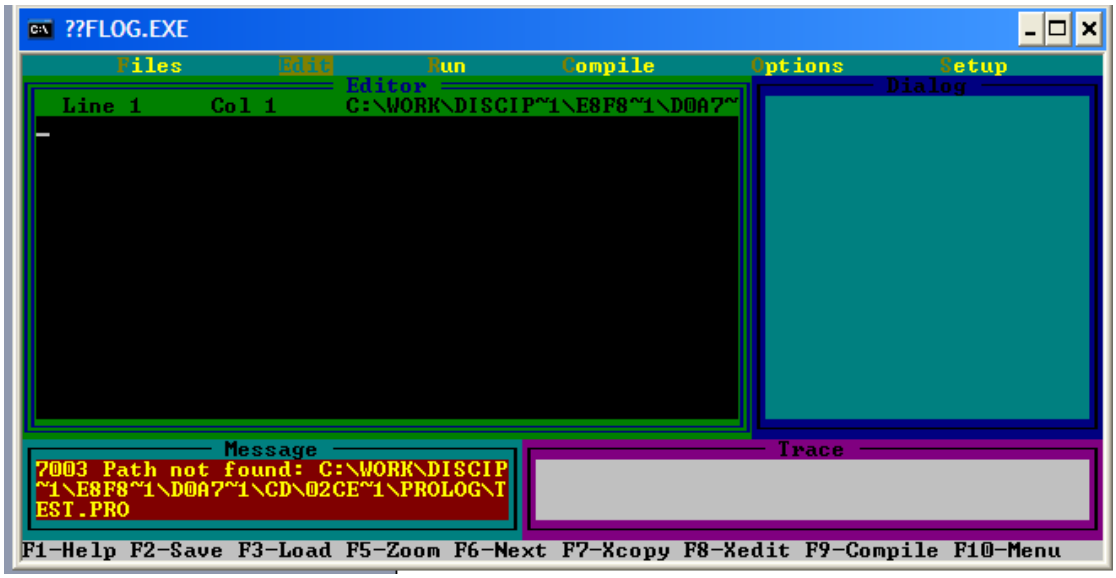



Рис.3 Переход в режим редактирования в среде Turbo Prolog

Если в окне редактирования остался текст какой-либо программы, следует просто удалить его или создать новый файл (меню Files – New).

1. Введите в редакторе текст программы из п. 3.1. Закончив ввод, выйдите из редактора в главное меню, нажав клавишу Esc.

2. Сохраните исходный код в файле с именем family_1.pro.

3. Запустите его на выполнение, выбрав в главном меню Run.

В данной программе нет секции goal, т.е. в программе отсутствует внутренняя цель, определяющая решение конкретной задачи. Поэтому, после запуска программы на выполнение и в случае отсутствия ошибок, системой активизируется окно Dialog (рис.4) и появляется приглашение на ввод внешней цели (Goal:)

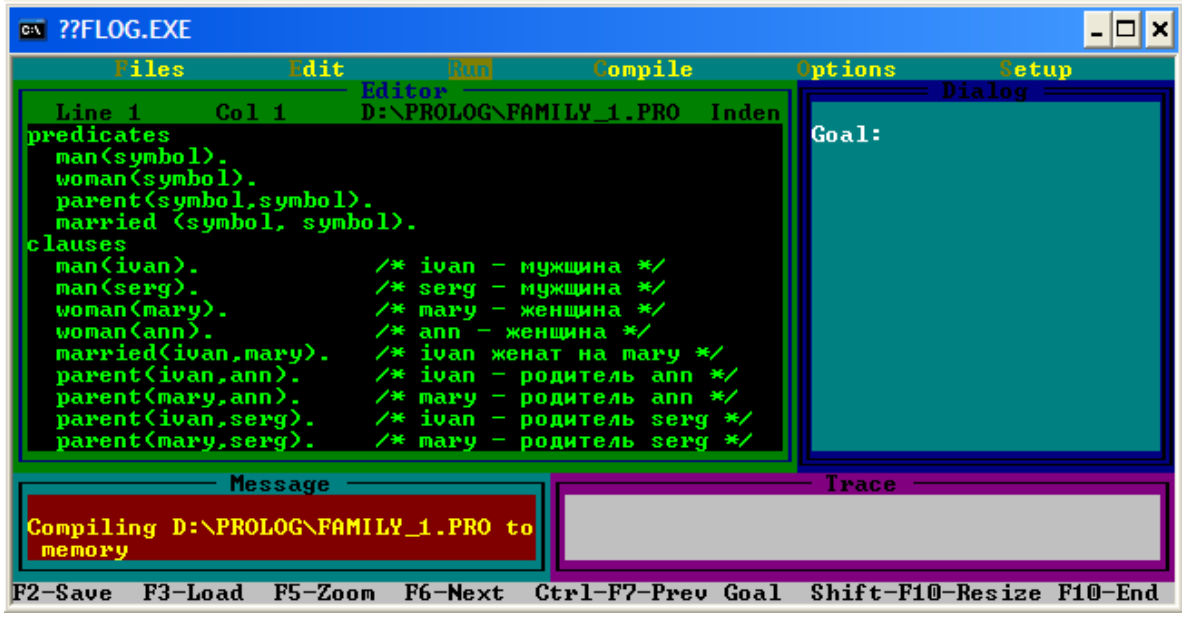



Рис.4 Запуск программы на выполнение в среде Turbo Prolog

### 3.3. Работа с программой в режиме диалога

Внешние цели – это запросы к программе, которые пользователь может формировать в диалоговом окне. Введите запрос

```
goal:
 parent(ivan,ann)
```

По окончанию набора текста запроса нажать Enter. До нажатия Enter запрос можно редактировать. Если в запросе нет ошибок, то в диалоговом окне Prolog выдаст сообщение: `yes`.

Система запоминает последний из введенных запросов. Для того, чтобы вызвать повторно предыдущий запрос, следует нажать клавишу F8. Вызовите повторно предыдущий запрос и отредактируйте его так, чтобы сформулировать следующий запрос к программе.

Задайте все вопросы к данной программе, объясните полученный результат и смысловое значение выводимых в окне Dialog сообщений.

Таблица 1

Виды и интерпретация запросов

| Вопросы | На Прологе | Ответ | Интерпретация ответа |
|---|---|---|---|
| Является ли Ivan родителем Ann? | `parent(ivan,ann)` | `yes` | Цель достигнута, успех |
| Является ли Ivan родителем Mary? | `parent(ivan,mary)` | `no` | Цель недостижима, неудача |
| Является ли Ivan родителем Petr? | `parent(ivan,petr)` | `no` | Цель недостижима, т.к. petr не упоминается в программе |
| Кто является родителем Ann? | `parent(X,ann)` | `X=ivan`<br>`X=mary` | Цель достижима при X=ivan, X=mary. Система выдает несколько ответов. |
| Кто дети Mary? Родителем кого является Mary? | `parent(mary,X)` | `X=ann`<br>`X=serg` | Система выдает несколько ответов |
| Кто чей родитель? | `parent(X,Y)` | `X=ivan, Y=ann`<br>`X=mary, Y=ann`<br>… | Цель достижима при X=Ivan и Y=Ann, при X=Mary и Y=Ann и т.д. |

### 3.4. Задание правил

1. Дополните программу *правилами*, позволяющими описывать следующие отношения: ребенок (кто, чей), отец (кто, чей), мать (кто, чья), сын (кто, чей) и дочь (кто, чья).

```
predicates
      ...
```

```
      child(symbol,symbol).
      father(symbol,symbol).
      mother(symbol,symbol).
      son(symbol,symbol).
      daughter(symbol,symbol).
clauses
      ...
      child(X,Y):- parent(Y,X).
      father(X,Y):- man(X),parent(X,Y).
      mother(X,Y):- woman(X),parent(X,Y).
      son(X,Y):- man(X),child(X,Y).
      daughter(X,Y):-woman(X),child(X,Y).
```

Первое правило `child(X,Y)` устанавливает, что «X является ребенком Y», если истинно выражение «Y является родителем X».

Второе правило `father(X,Y)` говорит, что «X является отцом Y», если выполняется, что «X является родителем Y» и «X является мужчиной».

Третье правило `mother(X,Y)` определяет, что «X – мама для Y», если «X является родителем Y» и «X – женщина».

Четвертое и пятое правила задают отношения `son(X,Y)` и `daughter(X,Y)` аналогично второму и третьему правилам, с учетом того, что «X является ребенком для Y».

2. Сохраните измененный код в файле с тем же именем family_1.pro.

3. Запустите его на выполнение, выбрав в главном меню Run.

4. Самостоятельно сформируйте запросы для определения мамы для Сергея, сына для Ивана, дочь для Ани и т.п. Введите эти запросы в окно Dialog, объясните полученный результат и смысловое значение выводимых сообщений.

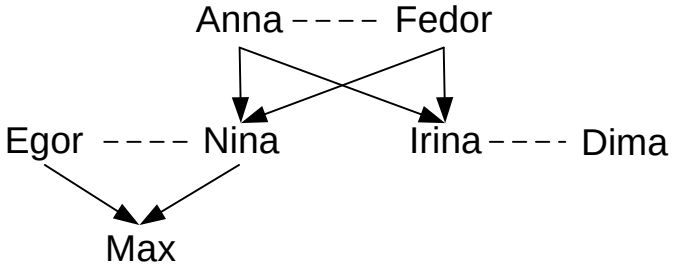
Введя в программу несколько таких правил, можно последующие родственные отношения определять через них. Например, правило для нахождения мужа_внучки (кто, чей) может быть записано следующим образом:

```
husband_granddaughter(X,Y):- granddaughter(G,Y),
                             husband(X,G).
```

Правило устанавливает, что «X для Y является мужем внучки», если выполняется, что существует такая G, что «G является внучкой для Y» и «X является мужем для G».

### 3.5. Индивидуальное задание

1. Предварительно на бумаге нарисуйте 3 поколения ***своего*** генеалогического дерева до бабушек и дедушек. Например,



2. Измените код исходной программы для своей семьи, добавляя при необходимости факты, описывающие женщин, мужчин, родителей и супругов. Обратите внимание на то, что предикаты с одинаковыми именами в разделе clauses должны быть сгруппированы вместе.

3. Введите в программу **4 новых правила**, определяющих тех членов семьи, которые требуются в соответствии с индивидуальным вариантом студента (см. таблицу 2 в п.3.6).

В качестве дополнения в программе могут быть заданы правила определения мужа или жены, брата или сестры. ***Но это должны быть правила, а не факты.*** Предикат вида sister (ann, serg) не допустим.

4. В программе должны быть присутствовать факты, определяющие тех людей-родственников, которых требуется найти согласно индивидуальному заданию (придумайте, если их реально не существует). Например, дедушка для Максима, теща для Егора, тетя для Максима и т.д.

### 3.6. Варианты индивидуальных заданий

Таблица 2

Виды родственных связей

| | | |
|---|---|---|
| 1. | брат (brother) | |
| 2. | сестра (sister) | |
| 3. | внук (grandson) | сын ребенка (сына или дочери) |
| 4. | внучка (granddaughter) | дочь ребенка (сына или дочери) |
| 5. | дедушка (grandfather) | отец родителя (отца или матери) |
| 6. | бабушка (grandmother) | мать родителя (отца или матери) |
| 7. | тетя (aunt) | сестра родителя (отца или матери) |
| 8. | дядя (uncle) | брат родителя (отца или матери) |
| 9. | племянник (nephew) | сын сестры |
| 10. | племянник (nephew) | сын брата |
| 11. | племянница (niece) | дочь сестры |
| 12. | племянница (niece) | дочь брата |
| 13. | двоюродный брат (cousin_brother) | сын тети |
| 14. | двоюродный брат (cousin_brother) | сын дяди |
| 15. | двоюродная сестра (cousin_sister) | дочь тети |
| 16. | двоюродная сестра (cousin_sister) | дочь дяди |
| 17. | муж (husband) | |
| 18. | жена (wife) | |
| 19. | тесть (wifes_father) | отец жены |
| 20. | теща (wifes_mother) | мать жены |
| 21. | свекор (husbands_father) | отец мужа |
| 22. | свекровь (husbands_mother) | мать мужа |
| 23. | деверь (husbands_brother) | брат мужа |
| 24. | золовка (husbands_sister) | сестра мужа |
| 25. | шурин (wifes_brother) | брат жены |
| 26. | свояченица (wifes_sister) | сестра жены |

| | | |
|---|---|---|
| 27. | зять (husband_of_daughter) | муж дочери |
| 28. | зять (husband_of_ sister) | муж сестры |
| 29. | невестка (wife_of_son) | жена сына |
| 30. | невестка (wife_of_ brother) | жена брата |

Таблица 3

Варианты заданий

| № варианта | Отношения (из табл.2) |
|---|---|
| 1. | 3, 10, 19, 23 |
| 2. | 3, 7, 21, 25 |
| 3. | 3, 15, 20, 23 |
| 4. | 4, 8, 22, 30 |
| 5. | 4, 9, 19, 24 |
| 6. | 4, 13, 21, 26 |
| 7. | 5, 11, 20, 24 |
| 8. | 5, 7, 22, 26 |
| 9. | 5, 16, 19, 27 |
| 10. | 6, 8, 21, 29 |
| 11. | 6, 10, 20, 27 |
| 12. | 6, 14, 21, 30 |
| 13. | 3, 11, 19, 28 |
| 14. | 5, 15, 22, 25 |
| 15. | 4, 14, 20, 28 |

## 4. ПОСТРОЕНИЕ БАЗЫ ЗНАНИЙ

Во второй части лабораторной работы требуется разработать базу знаний, содержащую информацию по заданной теме, в которой необходимо предусмотреть ответы на определенные вопросы. База знаний должна содержать не менее трех видов предикатов, 15–20 фактов. Вопросы к базе знаний необходимо представить в виде правил. Для вывода всех ответов использовать предикат `fail`, который всегда дает неудачу в доказательстве и, таким образом, поддерживает поиск с возвратом.

### 4.1. Простые встроенные предикаты ввода-вывода

Встроенные предикаты обеспечивают возможности ввода-вывода информации:

1. `write` – вывод значения аргумента на экран. Так как после команды `write` Пролог считает первое найденное решение задачи единственным и не проводит поиск остальных, то введение команды `fail` искусственно приводит к неудаче, тем самым заставляя искать следующее решение.

2. `nl` – перевод на следующую строку.

3. `readint, readchar` – чтение значения указанного типа, вводимого с клавиатуры, и его сопоставление с аргументом.

Пример программы, содержащей предикаты ввода-вывода, представлен на рис.1.

### 4.2. Пример базы знаний сдачи студентами экзаменационной сессии

В базе знаний будет содержаться информация студентах (номер зачетки, фамилия, год рождения, номер группы), группах (номер группы и кафедра) и экзаменационных оценках (номер зачетки, дисциплина, оценка).

```
predicates
     group(symbol,symbol).
     student(real,symbol,integer,symbol).
     exam(real,symbol,integer).
clauses
     group("I-1-2",is).
     group("I-1-3","applied math").
     student(980123, alekseev, 1986,"I-1-3").
     student(980167, petrov, 1986,"I-1-2").
     student(980170, ivanov, 1986,"I-1-2").
     exam(980123, math, 50).
     exam(980167, math, 40).
```

Вопрос 1. Кто учится в группе `I-1-2` ?

```
predicates
     question1(symbol).
clauses
     question1(Gr):-student(_,Res,_,Gr),write(Res) ,nl, fail.
goal
     question1("I-1-2").
```

*Замечание*: Для обозначения переменной, на которую отсутствует ссылка в программе, используется анонимная переменная, которая обозначается одиночным знаком подчеркивания '_'.

Вопрос 2. Как сдал `Алексеев` из группы `I-1-3` экзамен по `математике`?

```
predicates
     question2(symbol,symbol,symbol).
clauses
     question2(Fam,Gr,Dis):-student(N,Fam,_,Gr),
                             exam(N,Dis,Res),write(Res).
goal
     question2(alekseev,"I-1-3",math).
```

### 4.3. Варианты заданий

1. База знаний содержит информацию о железнодорожных поездах (номер рейса, пункты отправления и назначения, название вокзала), станциях (название и географическая область их расположения), остановках поездов с указанием времени стоянки каждого поезда. Должны поддерживаться ответы на вопросы:
   – Определите маршрут следования заданного поезда.
   – С какого вокзала отправляются поезда, останавливающиеся в данном городе?

– Какой поезд стоит на указанной станции более определенного времени? (вывести его номер, пункты отправления и назначения)
– Какие станции находятся в той же географической области, что и пункт назначения поезда с заданным номером?

2. Каталог студенческой библиотеки содержит информацию о читателях-студентах (номер его читательского билета, фамилию, номер группы), книгах (шифр книги, название, автор, год издания), а также сведения о том, когда следует вернуть книгу. Должны быть предусмотрены ответы на вопросы:
– Кто из студентов указанной группы взял книгу определенного автора?
– Каковы задолженности конкретного студента? (вывести название и автора книги, дату ее сдачи)
– Есть ли в библиотеке книги, изданные позднее указанного года?
– У кого из студентов есть книги автора, написавшего определенное произведение? (вывести его фамилию и номер группы)

3. В базе знаний присутствует информация о пациентах (номер карты, фамилия, пол, год рождения), болезнях (название и симптомы заболевания), данные медицинской карты (когда и чем болел пациент, способ лечения, фамилия врача). Предусмотреть ответы на вопросы:
– Какие симптомы заболевания данного пациента?
– Вывести список пациентов с указанием их заболеваний, возраст которых превышает заданное значение.
– У кого заболевание идентично заболеванию определенного пациента?
– Кого лечил указанный врач в заданный диапазон времени?

4. База знаний содержит информацию о работниках предприятия (табельный номер, фамилия, должность, отдел), проектах (шифр, название) и заданиях, выполняемых работниками в рамках проектов. Предусмотреть ответы на вопросы:
– Есть ли на предприятии сотрудники старше определенного возраста?
– В каких проектах участвуют сотрудники данного отдела? (вывести их фамилии и задания)
– Кто участвует в том же проекте, что и указанный работник?
– Какие задания в проектах выполняют специалисты заданной квалификации в данном отделе?

5. База знаний содержит информацию о сотрудниках фирмы (табельный номер, фамилия, пол, должность, отдел, зарплата), их начальниках (табельный номер руководителя и название возглавляемого им отдела), а так же о возрасте детей сотрудников. Предусмотреть ответы на вопросы:
– У кого из сотрудников-мужчин заданного отдела есть дети?
– Найти детей указанного сотрудника не старше определенного возраста (вывести их имена и возраст).
– Есть ли в данном отделе сотрудники заданной должности?
– Кто из сотрудников получает зарплату, равную зарплате начальника своего отдела?

6. База знаний содержит результаты теннисных партий (номер партии, имена игроков, счет), сыгранных членами некоторого клуба (фамилия, год рождения, рейтинг), проведенных в некоторых период времени (номер партии, место и дата проведения). Предусмотреть ответы на вопросы:

- С кем из членов клуба встречался заданный игрок и каковы результаты поединков?
- Выигрывал ли заданный игрок в определенный промежуток времени?
- Кто, когда и с каким счетом побеждал данного игрока?
- Если ли в клубе игроки младше определенного возраста? (вывести их имена и возраст)

7. База знаний содержит информацию о животных (название, класс, занесены ли они в Красную книгу), заповедниках (название, месторасположение, площадь), а также численность животных в каждом из заповедников. Предусмотреть ответы на вопросы:

- Какие животные из определенного класса имеются в данном заповеднике?
- Какие классы животных в данном регионе занесены в Красную книгу?
- Есть ли животные данного класса в данном регионе и какова их численность?
- В каких заповедниках заданного региона обитает указанное животное? (вывести название и площадь заповедника)

8. База знаний содержит информацию о программистах (фамилия, образование, год рождения), его профессиональной деятельности: разработанных программных продуктах (названия и сроки разработки) и предприятиях, на которых он работал (названия и период работы). Предусмотреть ответы на вопросы:

- Вывести список программистов с указанным образованием, занимавшихся созданием данного программного продукта.
- Какие программные продукты заданный программист разработал в определенный промежуток времени?
- Что конкретный программист разрабатывал на заданном предприятии?
- Кто был соавтором заданного программиста в указанное время?

9. База знаний содержит информацию о лекарствах (название, производитель, фармакологическая группа), аптеках города (номер и адрес), а также цены и количество лекарственных препаратов в каждой из аптек. Предусмотреть ответы на вопросы:

- В каких аптеках есть данное лекарство? (указать его количество и стоимость)
- Какие лекарства есть в продаже в указанной аптеке?
- Где продаются лекарства указанной фирмы-производителя из определенной фармакологической группы? (вывести название лекарства, номер и адрес аптеки)
- Где цена на данное лекарство ниже, чем в указанной аптеке?

10. База знаний содержит информацию о владельцах автомобилей (фамилия, год рождения, адрес, номер автомобиля), автомобилях (регистрационный номер, марка, цвет), производителях (фирма, страна, марка). Предусмотреть ответы на вопросы:

- Кто является владельцем определенного транспортного средства? (найти его имя, возраст, адрес)
- Есть ли владельцы автомобилей заданной марки определенного цвета?
- Какие машины предпочитают люди определенного возраста?
- Кто пользуется автомобилями того же производителя, что и данный автомобиль?

11. База знаний содержит информацию о спортсменах, занимающихся прыжками в высоту (фамилия, год рождения, пол, страна), спортивных мероприятиях (вид, дата и место проведения) и результатах выступления (взятая высота и занятое место). Предусмотреть ответы на вопросы:

- Кто из спортсменок выступал на указанном мероприятии (вывести фамилию и страну)?
- Занимали ли указанное призовое место спортсмены, старше определенного возраста? (вывести фамилию и год рождения)
- Вывести данные спортсменов из заданной страны, взявшие указанную высоту на определенном спортивном мероприятии.
- Какие еще спортивные мероприятия проводились в том же году, что и указанное?

12. База знаний содержит информацию о товарах фирмы (код, название, цена), клиентах (номер, фамилия, телефон) и заказах, сделанных клиентами (количество товара, дата и стоимость заказа). Предусмотреть ответы на вопросы:

- Что, когда и в каком количестве покупал конкретный клиент?
- Кто покупал указанный товар в определенный промежуток времени? (вывести фамилию и телефон)
- Стоимость каких заказов превышает заданное значение? (вывести фамилию клиента, товар и стоимость заказа)
- Когда и что заказывали те клиенты, которые делали заказ товара в указанном количестве?

13. База знаний содержит информацию об абонентах мобильной связи (номер телефона, тариф, баланс), тарифах (название, оператор, дата появления) и стоимостях услугах каждого тарифа. Предусмотреть ответы на вопросы:

- Кто пользуется услугами указанного оператора?
- Услугами какого оператора пользуются абоненты, баланс на счету которых превышает заданное значение?
- Вывести те тарифы, цена на определенную услугу которых меньше, чем в том тарифе, которым пользуется указанный абонент.
- Кто из абонентов не перешел на тарифы, появившиеся позднее указанного срока? (вывести фамилию и телефон)

14. База знаний содержит информацию о музыкальных группах (название, страна, дата создания), их составах (имя исполнителя, год рождения, пол) и альбомах (тираж, дата выхода). Предусмотреть ответы на вопросы:

- Какие альбомы и когда выпустила группа, в которой играет указанный исполнитель?
- В составе каких групп есть женщины моложе указанного возраста?
- Каким тиражом разошелся альбом музыкальной группы из определенной страны в указанный год?
- Кто еще является членом коллектива, в котором выступает данный исполнитель?

15. База знаний содержит информацию о кинофильмах (название, режиссер, жанр, год выпуска), актерах (фамилия, год рождения, пол) и сыгранных ролях. Предусмотреть ответы на вопросы:

- В фильмах каких режиссеров играл указанный актер?
- Играла ли в данном фильме актриса старше указанного возраста?
- Какие роли исполнял данный киноактер в фильмах определенного жанра? (вывести название фильма и роль)
- Кто еще играл в тех же кинофильмах, что и заданный актер?

## 5. ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

**Задание 1.** Ознакомиться с общими сведениями о языке Prolog, представленными в п.1 описания лабораторной работы №3.

**Задание 2.** Изучить среду разработки Turbo Prolog и ознакомиться со структурой программ на Turbo Prolog, описанной в п.2 лабораторной работы №3.

**Задание 3.** Проверить работоспособность программы, листинг которой представлен в п.3.1. Задайте вопросы к данной программе (см. табл.1) и объясните полученный результат.

**Задание 4.** Выполните индивидуальное задание, согласно последовательности действий, представленной в п.3.5. Создайте запросы к программе, позволяющие находить требуемых родственников для конкретных людей (например, дядю для Максима).

**Задание 5.** Разработать базу знаний, содержащую информацию по теме, указанной преподавателем. База знаний должна содержать не менее трех видов предикатов и 15–20 фактов. Вопросы к базе знаний необходимо представить в виде правил, вызываемых в секции goal.

## 6. СПИСОК КОНТРОЛЬНЫХ ВОПРОСОВ

1. Почему поиск решения задачи в пространстве состояний является одним из основных методов решения интеллектуальных задач?
2. Приведите примеры задач, представление которых возможно с помощью графа пространстве состояний.
3. Что такое граф пространства состояний?

4. В чем заключается смысл построения графа пространства состояний?

5. Какие понятия используются при формализации задачи в пространстве состояний?

6. Дайте определение понятию “состояние задачи”.

7. Как могут быть заданы целевые состояния?

8. Что характерно для операторов задачи?

9. Как определяется решение задачи и пространство состояний в терминах состояний и операторов?

10. Как графически изображается пространство состояний?

11. Какими тремя компонентами описывается задача при ее формализации с использованием пространства состояний?

12. Чем явно заданный граф пространства состояний отличается от неявно заданного?

13. Каким образом выбор операторов влияет на размер пространства состояний?

14. Как можно уменьшить пространство состояний?

15. Опишите использование переменных в описаниях состояний.

16. Приведите пример формализации задачи в пространстве состояний.

17. Почему поиск решения в глубину может привести к бесконечному

## 7. СОДЕРЖАНИЕ ОТЧЕТА

1. Титульный лист.
2. Генеалогическое дерево студента до третьего поколения родственников.
3. Условие первого индивидуального задания.
4. Листинг программы на языке Prolog с комментариями.
5. Экранные формы (скриншоты) работы программы с вызовом каждого предиката-отношения, требуемого по условию задания.
6. Условие второго индивидуального задания для разработки базы знаний.
7. Листинг программы на языке Prolog с комментариями.
8. Экранные формы (скриншоты) работы базы знаний.

## 8. СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. Братко, Иван. Алгоритмы искусственного интеллекта на языке PROLOG, 3-е издание. : Пер. с англ. – М. : Издательский дом "Вильяме", 2004. – 640 с.

2. Шрайнер П.А. Основы программирования на языке Пролог. Курс лекций. – ИНТУИТ, 2005. – 176 с.

3. Хабаров С.П. Интеллектуальные информационные системы. PROLOG - язык разработки интеллектуальных и экспертных систем.: Учеб. пос. – СПб.: СПбГЛТУ, 2013. – 138 с.