Лабораторная работа.
Унификация декларативной программы.

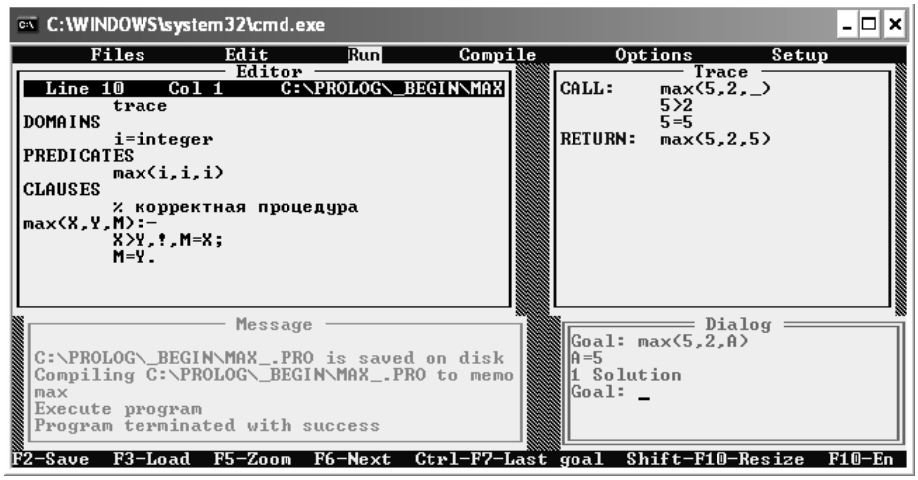
Время: 180 мин.
Что нужно освоить:
1) каким образом факты и правила описывают отношения между объектами предметного поля;
2) что такое унификация (сопоставление с образом) и какие варианты её использования возможны;
3) основные правила работы с системой Пролог.

Часть 1. Самое начало или «где у него кнопка?».

Переходим в папку с системой Пролог, например, C:\PROLOG. Находим там файл start.bat. Его содержимое:

```
key /ext /scan=29 /graph=255
prolog
```

Запускаем на выполнение файл start.bat. Последовательно запускается русификатор и система Пролог. Первая командная строка запускает русификатор, настроенный таким образом, что однократное нажатие на **правый Ctrl** приводит к смене языка ввода. Вторая командная строка запускает систему Пролог.



Верхнюю строчку экрана занимает Меню (File, Edit, Run, Compile, Options, Setup), названия пунктов которого говорят сами за себя. Вход в соответствующие пункты меню возможен через сочетания клавиш:

Активировать меню ФАЙЛ.....Alt-F
Активировать редактор программ .....Alt-E
Активировать меню ОПЦИИ .....Alt-O
Активировать меню УСТАНОВКИ.....Alt-S

Выполните следующие задачи.

1. Установите режим компиляции в память компьютера.
2. Сконфигурируйте положение, размеры и цветовые палитры окон.
   Переключение между окнами – клавиша F6. Размеры окон можно менять стрелочками, положение окон стрелочками на дополнительной цифровой клавиатуре (при нажатой клавише NumLock).
3. Определите пути доступа к файлам, установив нужные для вас каталоги.
4. Выполненные установки запишите в файл конфигурации.

Вторую строчку экрана занимает служебная строка, где отображаются:

– текущая позиция курсора в тексте программы,
– путь к файлу программы,

– режим отступа,

– режим ввода с клавиатуры.

Ниже идут окна Пролога:

1) редактор программы (Edit) – вход в окно Alt-E;

2) окно сообщений (Message);

3) окно диалога / выполнения программы (Dialog) – вход в окно Alt-R;

4) окно трассировки программы (Trace).

Перейдите в режим редактирования (активируйте окно редактора программ) – Alt-E, затем нажмите F1. Вам будут предложены разделы Справки Пролога. Изучите содержимое разделов. Выход из справки – Esc. Справку удобнее смотреть в обычном текстовом редакторе, например, в Блокноте (файл справки PROLOG.HLP в корневом каталоге Пролога).

В приложении №1 приведены различные варианты использования горячих клавиш. Исследуйте эти возможности. Наиболее важные, на ваш взгляд, сочетания клавиш и порядок их использования запишите у себя в тетради.

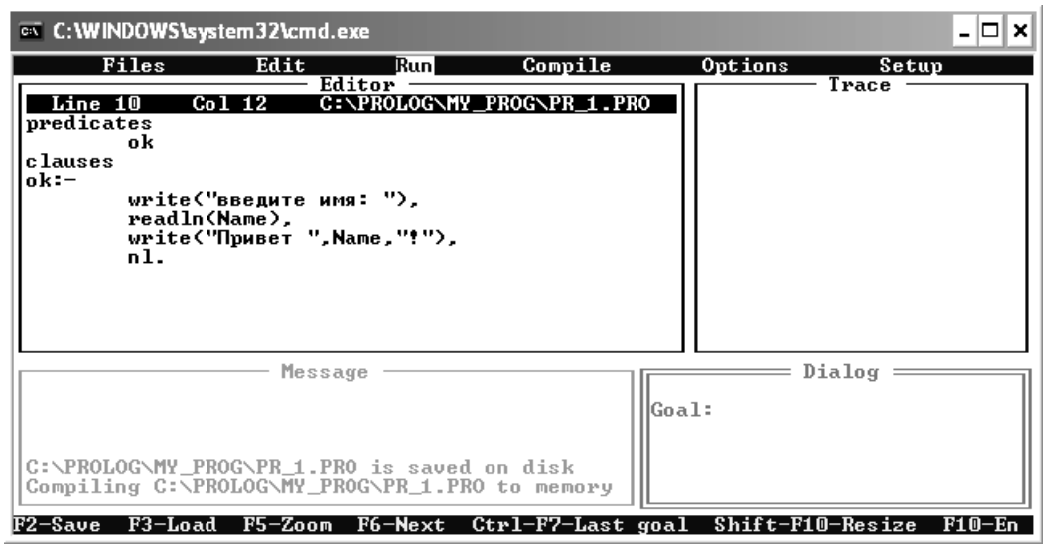
Часть 2. Приступаем к работе (первые шаги).

Шаг 1.

Каждая программа на Прологе представляет собой текстовый файл, который для запуска из системы Пролог должен иметь расширение «.PRO».

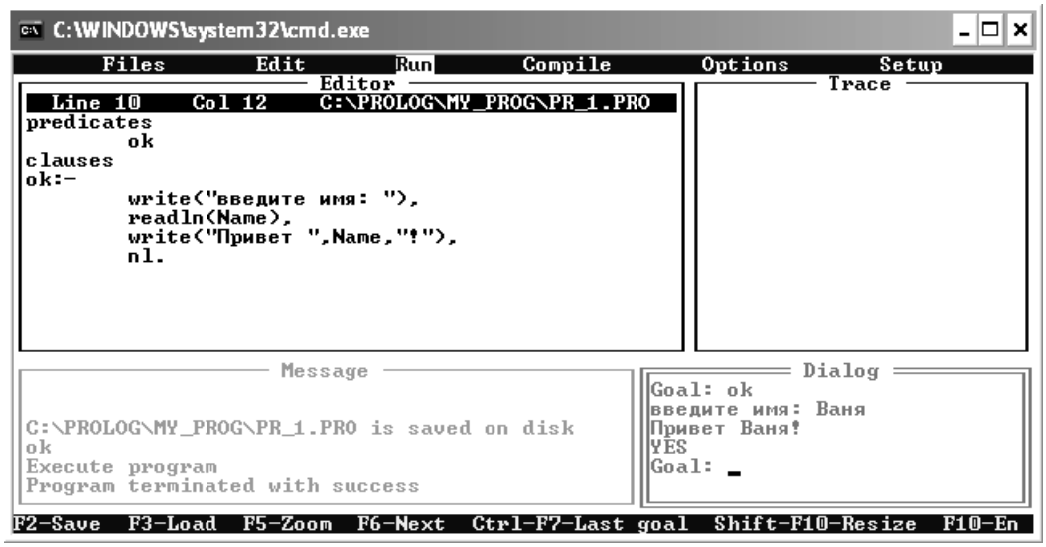
Создайте Проводником в папке с системой Пролог C:\PROLOG (путь к системе Пролог может быть иной) дополнительную папку для своей работы с именем, например, MY_PROG и установите текущую директорию работы Пролога C:\PROLOG\MY_PROG (меню: Files/Change dir).

Создайте уже в Прологе через меню новый файл. На начальном этапе будем опираться в основном на встроенные предикаты (write, readln, nl). Наберите следующий текст программы, содержащий безаргументный предикат **ok**:



Сохраните программу (клавиша F2) под именем pr_1.pro и проверьте, в какую папку была сохранена программа (должно быть так как установили вы: C:\PROLOG\MY_PROG).

Нажмите сочетание клавиш Alt-R для запуска программы на выполнение – курсор переместится в окно Dialog и система Пролог будет ожидать от вас вопроса **Goal: _** (цели, которую необходимо достичь). Введите **ok** и нажмите Enter. После диалога с Прологом вы можете получить такой результат:



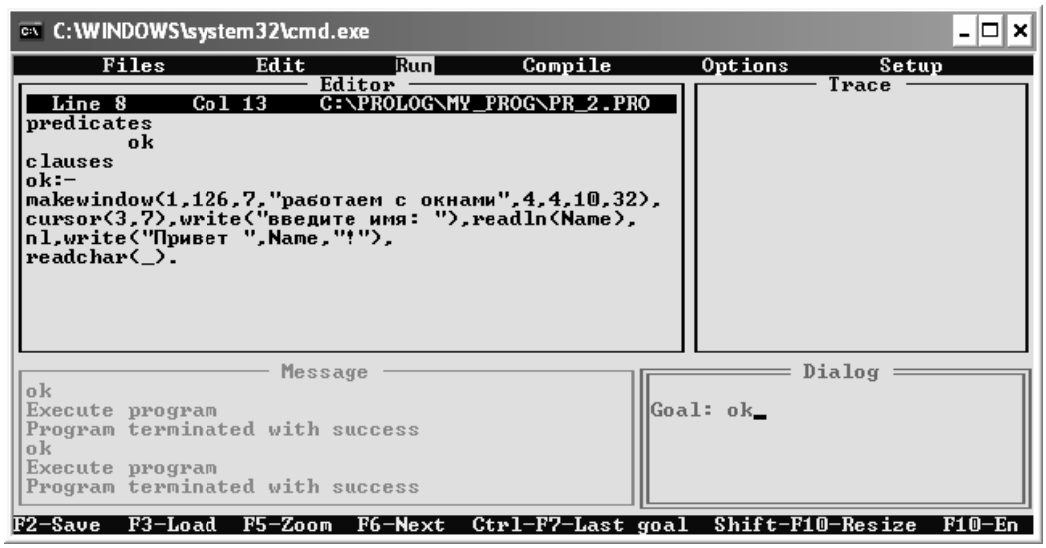
Что произошло? После получения задачи на достижение цели **ok** Пролог приступает к унификации. В данном случае находит голову искомой цели ok в теле программы и пытается проверить её реализуемость. Оказывается, что ok это предложение вида правило, то есть это предложение состоит из головы и тела. Тело, в свою очередь, представлено несколькими подцелями (предикатами). Чтобы предикат **ok** выполнился необходимо достичь подцель write(“введите имя: ”), подцель readln(Name), подцель write(“Привет ”,Name,”!”,) и подцель nl. Все эти предикаты выполняются безусловно. Пролог пытается последовательно их достичь и когда все они выполнятся, то он с удовлетворением отмечает, что цель достижима – **Yes** (см. окно Dialog). После чего объявляет пользователю, что он готов к поиску следующей цели – **Goal: _** .

В этой программе используется один нестандартный предикат, то есть тот который отсутствует в лексике Пролога, а именно предикат **ok**. Все нестандартные предикаты должны быть объявлены (см. тест программы, раздел predicates). Вы можете поменять имя предиката на такое, которое вам кажется более приемлемым для обозначения решаемой им задачи и попробовать запустить программу вновь.

Шаг 2.

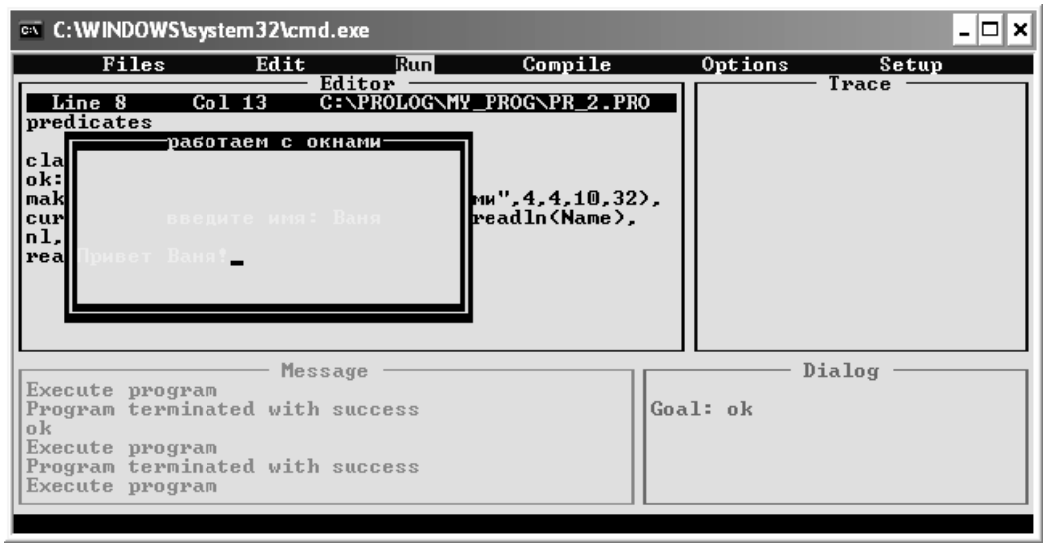
Работа с окнами (интерфейс разрабатываемой вами программы).

Внесите некоторые изменения в текст программы:



и сохраните её под именем pr_2.pro (только в данном случае ***не используйте F2***, иначе вы сотрете предыдущую программу pr_1.pro; надо использовать пункт меню Files/Write to).

Поработайте с программой. В частности запустите её с предикатом readchar и без него. И вам станет понятно, зачем я добавил его...



Естественно, что этот стандартный предикат удерживает окно до того момента пока не получит какой-нибудь символ (т.е. нажатие любой клавиши).

Знак подчеркивания в качестве аргумента предполагает, что это анонимная переменная и после нажатия той самой «какой-нибудь» клавиши её код нигде не будет сохранен. Сравните с readln(Name), здесь как раз нам важно сохранить набранную пользователем строку, поэтому у переменной есть имя.

Дальше я предлагаю вам поработать самим. Разберитесь с предикатом cursor, с оформлением окна (дизайн: цвет фона и шрифта, размеры окна и оформление рамки окна). Напомню, что в Прологе есть справка.

Часть 3. Описание мира и процесс унификации.

Возвращаемся собственно к трудностям, возникающим на начальном этапе освоения Пролога. Чтобы решить задачу хочется машине рассказать, как её решать, то есть составить алгоритм. Однако, программа в логическом программировании не есть алгоритм, а есть база знаний – совокупность фактов и правил, описывающих отношения между объектами предметного поля. Мы формулируем вопрос к базе знаний, а Пролог сам в ходе последовательной унификации пытается найти ответ.

Унификация есть процесс сопоставления вопроса с фактами и правилами базы знаний с целью доказательства его реализуемости. Здесь важно отметить, что процесс сопоставления первичной основной цели происходит последовательно по всем предложениям в базе знаний – сверху вниз. Причем, если цель оказывается головой какого-то предложения, тело которого состоит из нескольких предикатов, то основная цель разбивается на подцели, доказательством которых Пролог занимается по отдельности и последовательно. То есть процесс сопоставления подцелей в теле правила всегда происходит слева направо.

Попробуем разобраться как именно происходит процесс унификации, для чего обсудим два вопроса:

1) определение отношений на основе фактов;

2) определение отношений на основе правил.

Определение отношений на основе фактов.

Пролог приспособлен для решения не вычислительных задач, которые связаны с описанием объектов и отношений между ними.

Пусть мир представлен фактами:

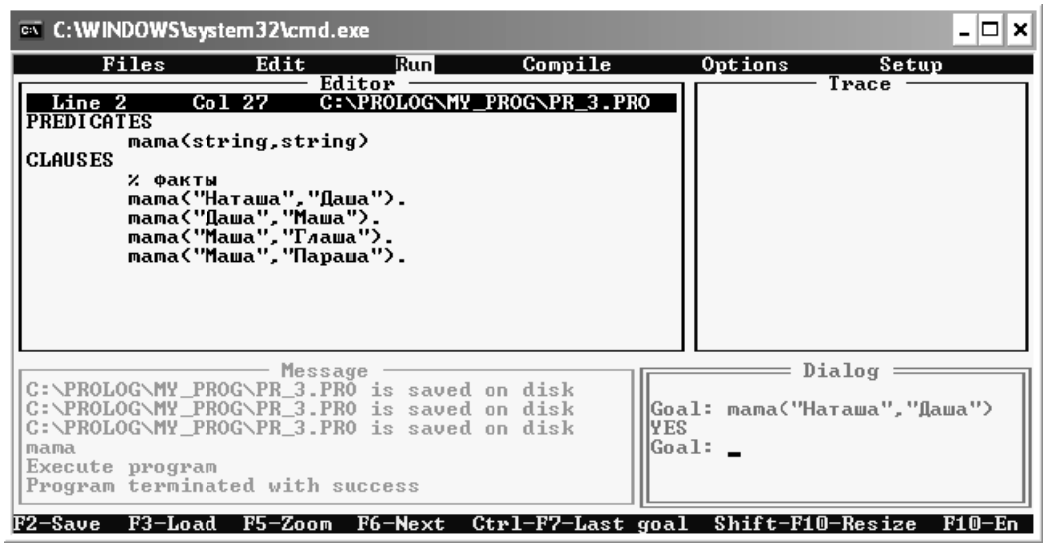
1) Наташа есть мама Даши;

2) Даша – мама Маши;

3) Маша – мама Глаши и Параши.

Пусть двухаргументный предикат `mama` задает отношение между первым и вторым аргументами, предписывая, что первый аргумент является мамой по отношению ко второму.

Создайте новый файл pr_3.pro и наберите следующую программу:



Спросим, является ли Наташа мамой Даши:

`mama ("Наташа", "Даша")`

Приступив к последовательному (сверху - вниз) сравнению поставленной цели с фактами, содержащимися в базе знаний, и, обнаружив уже на втором шаге, что mama (“Наташа”,”Даша”) – это факт, о существовании которого явно утверждается в тексте программы, Пролог отвечает Yes. Иными словами, если вопрос по сути есть предикат с означенными аргументами, то поиск решения есть простое сравнение со всеми фактами из базы знаний, а возможные исходы поиска Yes или No.

Спросим иначе:

```
mama (X, "Даша")
```

Этот вопрос содержит только один означенный аргумент (Даша), а в качестве первого аргумента – неозначенная переменная X. Пролог последовательно просмотрит все факты и, наткнувшись на такой, который содержит в качестве второго аргумента значение “Даша” конкретизирует переменную X значением “Наташа”.

! Сравните, чем будет отличаться поиск ответа на вопрос: `mama ("Маша", X).`

Проверьте также вариант: `mama (Y, X).`

Если есть необходимость определить только часть информации, например, перечислить только дочерей, тогда можно использовать анонимную переменную в вопросе:

```
mama (_, X).
```

Получим ответ:

```
X=Даша
X=Маша
X=Глаша
X=Параша
4 Solutions
```

И, наконец, если надо получить ответ на вопрос: есть ли информация о людях, находящихся в отношении "мама - дочка", то его можно сформулировать в виде:

```
mama (_,_),
```

В данном случае нам не важны конкретные имена, а интересует, есть ли в нашей базе знаний хотя бы один соответствующий факт. Ответом в данном случае будет просто "Yes". Система сообщит о том, что у нее есть информация об объектах, связанных отношением "mama".

Однако, нашей программе можно задать вопрос и посложнее, например, спросить кто является мамой мамы Глаши: `mama(X,Y),mama(Y,”Глаша”).`

! Проверьте, какой будет ответ. Поменяйте в вопросе предикаты местами и проверьте, будет ли получено верное решение и имеет ли значение для Пролога порядок следования предикатов?

Можно также узнать, найдется ли такая женщина, которая является одновременно мамой как для Глаши так и для Параши.

Можно задать вопрос и более общий – есть ли такая мама, у которой две дочери:

```
mama(Q,W),mama(Q,E),W<>E.
```

! Проверьте как сработает Пролог при поиска ответа на такой вопрос и прокомментируйте результаты.

Определение отношений на основе правил.

Правила значительно расширяют возможности базы знаний, так как они универсальны и могут вместо конкретных значений содержать переменные.

К примеру, введем в программу правило, которое определяет отношение бабушка, как мама мамы:

```
baba(X,Y):-
          mama(X,Z),
          mama(Z,Y).
```

Обычно для легкости чтения правила записывают именно в несколько строчек, но это не предопределено специально синтаксисом Пролога и при необходимости вы можете то же самое правило записать и в одну строчку:

```
baba(X,Y):-mama(X,Z),mama(Z,Y).
```

Сделайте программу такой как показано ниже:

```
DOMAINS
      s = string
PREDICATES
      mama(s,s)
      baba(s,s)
CLAUSES
      mama("Наташа","Даша").
      mama("Даша","Маша").
      mama("Маша","Глаша").
      mama("Маша","Параша").
baba(X,Y):-
      mama(X,Z),
      mama(Z,Y).
```

Сохраните эту программу под именем pr_4.pro.

Несколько комментариев по поводу оформления программы.

Комментарий №1.

Программа на Прологе состоит из разделов. В данном случае вы видите следующие разделы: DOMAINS – раздел описания типов (доменов), PREDICATES – раздел описания предикатов и CLAUSES – раздел описания предложений. Возможно использование и других разделов: GOAL – раздел описания внутренней цели, CONSTANTS – раздел описания констант и DATABASE – раздел описания предикатов внутренней базы данных

Комментарий №2.

Заметьте, что раньше мы писали mama(string,string), а теперь mama(s,s).

Дело в том, что можно использовать описание доменов для сокращения имен стандартных доменов. В частности, чтобы не писать каждый раз string, можно задать описание s = string и далее использовать вместо ключевого слова string односимвольное обозначение s.

К стандартным доменам относятся:

integer - целое число (из промежутка -32768...32767);

real - действительное число (лежащее между ±1e-307...±1e308);

char - символ, заключенный в одиночные апострофы;

string - последовательность символов, заключенная в двойные кавычки.

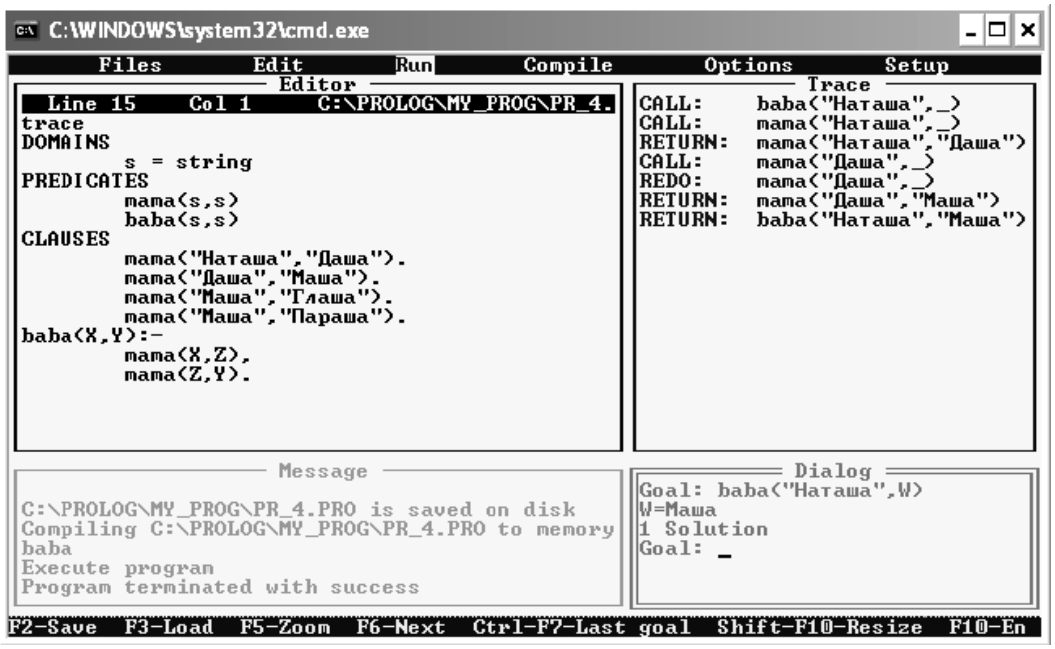
Запустите программу pr_4.pro и задайте вопрос baba(“Наташа”,W).

Процесс унификации будет проходить следующим образом:

Последовательно сверху – вниз будет происходить поиск предиката baba. Первые четыре факта не подходят. Наткнувшись на правило baba, Пролог сопоставляет вопрос baba(“Наташа”,W) с головой правила baba(X,Y), после чего переменная X конкретизируется значением “Наташа”, а W связывается с переменной Y из правила baba(X,Y). Далее Пролог пытается последовательно достигнуть подцели mama(“Наташа”,Z) и mama(Z,Y).

Просматривая факты Пролог натыкается на факт mama("Наташа","Даша"), что приводит к конкретизации переменной Z значением “Даша”. Подцель считается достигнутой и Пролог переходит к доказательству подцели mama(Z,Y), а если точнее, то – mama(“Даша”,Y). Поиск снова начинается с первого факта, который в данном случае не может быть сопоставлен с подцелью ввиду несопоставимости первых аргументов (оба они конкретизированы, но разными значениями, соответственно, “Наташа” и “Даша”). Но уже на втором шаге сопоставление становится возможным и переменная Y конкретизируется значением “Маша”.

Этот процесс можно лицезреть воочию. Для этого добавьте первой строчкой программы слово trace (см. скриншот).



Запустите программу, она будет выполняться пошагово с отображением текущего выполняемого действия (курсором в тексте программы и результатом этого действия в окне Trace). Чтобы выполнить следующий шаг необходимо нажимать клавишу **F10**. Исследуйте процесс унификации всех предыдущих программ. Стандартный предикат trace обычно используют в процессе отладки программы.

Если описываемый мир несколько сложнее, чем это может описать одно правило, то необходимо внести коррективы. Например, бабушка это не только мама мамы, но и мама папы:

```
baba(X,Y):-mama(X,Z),mama(Z,Y).
baba(X,Y):-mama(X,Z),papa(Z,Y).
```

Совокупность предложений с одинаковым предикатом в заголовке принято называть процедурой. Считается, что между правилами процедуры неявно существует логическая связка «ИЛИ».

То есть, к примеру, в семье, где у Глаши есть папа Степа и мама Маша, причем, у Степы мама Оля, а у Маши мама Даша – выходит, что у Глаши две бабушки.

! Попробуйте самостоятельно написать базу знаний про эту семью и сформулировать так вопрос к ней, чтобы получить исчерпывающий ответ об именах бабушек Глаши.

Унификация как процесс последовательного сопоставления.

Разберем простейший пример. Пусть задан предикат t с двумя аргументами. К примеру, первый аргумент есть порядковый номер какого-либо объекта, а второй – его величина. В базе есть один факт t(1,2).

```
DOMAINS
   i = integer
PREDICATES
   t(i,i)
CLAUSES
   t(1,2).
```

Зададим в окне диалога вопрос (запрос) к этой программе: t(X,Y).

Получим ответ: X=1, Y=2.

Процесс унификации проходил следующим образом.

Для Пролога вопрос есть цель, которую необходимо достичь. Пролог берет вопрос t(X,Y) и начинает последовательно сверху-вниз сравнивать его с фактами и правилами базы знаний. Там где обнаруживается предикат с таким же идентификатором как и у вопро-

са и с таким же количеством аргументов происходит сопоставление. В данном случае в базе знаний есть только один факт и нет никаких правил. Если в вопросе аргументы ничем не означены (то есть X и Y свободные переменные), то значения из найденного факта присваиваются соответствующим переменным.

Зададим другой вопрос: t(1,Y).

Получим ответ: Y=2.

Процесс унификации проходил следующим образом.

Пролог взял вопрос t(1,Y) и последовательно сверху-вниз сравнивая его с фактами из базы знаний обнаружил, что существует предикат с таким же идентификатором как и у вопроса и с таким же количеством аргументов. Так как в вопросе первый аргумент имеет конкретное значение, то Пролог проводит сопоставление сравнивая значение из вопроса и из факта. В данном случае они совпали, поэтому процесс сопоставления переходит на вторые аргументы. В вопросе второй аргумент ничем не означен, поэтому ему передается значение из факта. О чем, собственно, Пролог и сообщает.

Немного усложним программу (базу знаний), заменив факт правилом. Пусть задан предикат t с двумя аргументами. Он выполняется, если выполнимы две цели, указанные в его теле. Первая цель X=1, а вторая цель 2=Y. Запятая между ними означает, что предикат выполнится только при условии выполнения обеих целей в его теле, то есть запятая имеет смысл "И".

```
DOMAINS
   i = integer
PREDICATES
   t(i,i)
CLAUSES
   t(X,Y):-
      X=1,
      2=Y.
```

Зададим в окне диалога вопрос этой программе: t(X,Y).

Получим ответ: X=1, Y=2.

Процесс унификации проходил следующим образом.

По аналогии с описанным выше Пролог берёт вопрос t(X,Y) и начинает последовательно сверху-вниз сравнивать его с фактами и правилами базы знаний. В данной программе есть только одно правило. Наткнувшись на него Пролог уясняет, что идентификатор правила и количество его аргументов соответствуют вопросу. После чего Пролог пытается выяснить – выполнимо ли это правило при условиях, указанных в вопросе. В вопросе все переменные свободны, а правило выполнимо при выполнимости двух целей из его тела: X=1 и 2=Y.

В Прологе знак "=" не есть только знак присвоения. Его действие зависит от контекста. Только если с одной стороны от знака равенства стоит свободная переменная (то есть на момент унификации не имеющая значения), а с другой стороны расположено какое-либо значение или означенная переменная, тогда действие знака "=" можно считать эквивалентным присвоению. По итогам его выполнения свободная переменная получит значение с противоположной стороны от знака равенства (сторона не имеет значение) и цель признается выполненной.

В данном случае обе переменные свободны, поэтому каждая из целей в теле правила оказывается достижимой с выполнением присвоения соответствующих значений. Обратите внимание, что здесь не имеет значения ни последовательность целей в предложении ни место расположения относительно знака равенства переменной и ее будущего значения. Эта особенность характерна для декларативных программ.

Зададим программе другой вопрос: t(X,2).

Получим ответ: X=1.

Процесс унификации проходил следующим образом.

Первая переменная как и ранее означивается "1" - цель №1 достижима. Переходим ко второй цели: Y=2. Здесь перед выполнением цели, переменная уже не была свободной, а имела значение "2". Когда с обеих сторон от знака равенства находятся конкретные значения, тогда знак "=" имеет смысл сравнения. При равенстве обеих сторон - цель признается достижимой. Именно так и произошло в данном случае.

Сформулируем вопрос иначе: t(X,3).

Получим ответ: No Solution.

Процесс унификации проходил следующим образом.

Первая переменная как и ранее означивается "1" - цель №1 достижима. Переходим ко второй цели: Y=2. Здесь перед выполнением цели, переменная уже не была свободной, а имела значение "3". Когда с обеих сторон от знака равенства находятся конкретные значения, тогда знак "=" имеет смысл сравнения. Так как значения сторон отличаются - цель признается не достижимой.

Но в базе знаний может оказаться несколько фактов или правил. Разберем процесс унификации для таких случаев.

```
DOMAINS
   i = integer
PREDICATES
   t(i,i)
CLAUSES
   t(1,3).
   t(X,Y):-
      X=1,
      2=Y.
```

Зададим в окне диалога вопрос этой программе: t(X,3).

Получим ответ: X=1.

1 Solution.

Процесс унификации проходил следующим образом.

Пролог последовательно сопоставляет вопрос со всеми фактами и правилами из базы знаний. Очевидно, что один факт и одно правило совместно предоставляют две возможности к унификации исходной цели. Обратите внимание, что в данном случае последовательность предложений в программе не имеет никакого значения. Можно поменять факт и правило местами без изменения смысла программы. Однако в данном случае процесс унификации проходит только на факте. О чем и заявляет Пролог. Но возможна такая постановка вопроса, которая приведет к унификации цели как на факте, так и на правиле.

Зададим такой вопрос: t(X,Y).

Получим ответ:

X=1, Y=3

X=1, Y=2

2 Solutions.

Часть 4. Задания для самостоятельного исполнения.

Все задания предполагают отсутствие внутренней цели в программе, результаты можно получить только через окно Диалога.

1. Создайте предикат gip(K1,K2,G), находящий длину гипотенузы прямоугольного треугольника по длинам катетов.

Первые два аргумента входные, третий – выходной. То есть при запросе gip(3,4,G) мы должны получить ответ G=5.

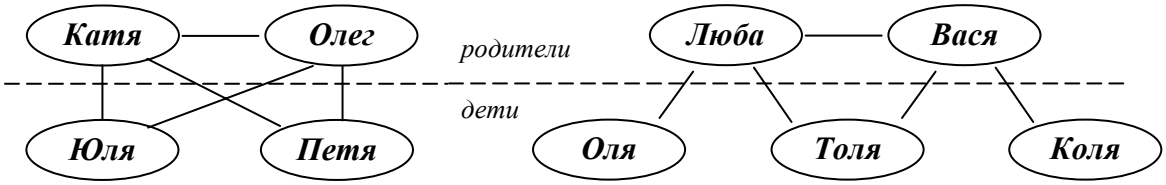
2. Создайте предикат, проверяющий попадает ли точка в круг. Предикат должен иметь пять входных аргументов – координаты точки, координаты центра окружности и длина радиуса. Если точка попадает в круг, то результатом работы программы будет вывод Yes, иначе – No.

Эта программа должна ссылаться на предикат из предыдущего задания, поэтому можно 1 и 2 программу сделать в одном файле. Предикат gip пригодится для того, чтобы найти расстояние от центра окружности до точки. Если найденное расстояние больше радиуса значит точка лежит за пределами окружности.

3. Создайте предикат, определяющий как называть мужчину в зависимости от его возраста:

– от 0 до 8 лет – дитя,
– от 9 до 14 лет – отрок,
– от 15 до 20 лет – юноша,
– от 21 до 60 лет – мужчина,
– от 61 года и далее – старик.

4. Для семей, изображенных на схеме:



задайте факты типа:

– мужчина (одноаргументный, всего 5 фактов),
– женщина (одноаргументный, всего 4 факта),
– родитель (двухаргументный, всего 8 фактов),
– в браке (двухаргументный, всего 2 факта).

Создайте двухаргументные предикаты (первый аргумент входной, второй – выходной), определяющие для конкретного человека (указанного в первом аргументе):

1) мать/отца;
2) дочь/сына;
3) сестру/брата;
4) кровную сестру/брата;
5) сводную сестру/брата;
6) жену/мужа.

Горячие клавиши

Показать горячие клавиши ..... Alt-H
Помощь по системе команд Prolog Shift-F1

Выход из Prolog ..... Alt-X
Загрузить файл F3
Загрузить файл из списка последних Alt-F3
Сохранить файл F2
Запустить программу на исполнение ..... Alt-R

Сменить окно просмотра F6
Активировать меню ФАЙЛ ..... Alt-F
Активировать редактор программ ..... Alt-E
Активировать меню ОПЦИИ ..... Alt-O
Активировать меню УСТАНОВКИ ..... Alt-S
Активировать меню ФАЙЛ ..... Alt-F
Показать информацию о системе Prolog ..... Alt-V

Текущее окно в полный экран/обратно F5
Перейти в следующее окно F6
Открыть вспомогательный файл F8
Вернуться в основной ..... Esc

Редактировать содержимое буфера Ctrl-F8
Режим редактора – “запись поверх” ..... Ins
Авто отступ при нажатии Enter ..... Alt-I

Текущее слово в верхний регистр ..... Ctrl-B + Ctrl-U
Текущее слово в нижний регистр ..... Ctrl-B + Ctrl-L
Сменить регистр текущего слова ..... Ctrl-B + Ctrl-R
Сменить текущую директорию ..... Ctrl-K + Ctrl-L

Копировать блок из файла ..... F7 …
выбираем файл … Ctrl-K + Ctrl-R …
выделяем блок … Ctrl-K + Ctrl-R

Поиск фразы ..... Ctrl-F3 … фраза … Ctrl-F3
или Ctrl-Q + Ctrl-F … слово … Enter
Поиск следующей фразы ..... Shift-F3 или Ctrl-O
Заменить ..... Ctrl-F4 или Ctrl-Q + Ctrl-A
Заменить снова ..... Shift-F4 или Ctrl-L

Прервать вып-ние прог-мы .. Ctrl-Break

Cursor movement		
Line up	↑	Ctrl-E
Line down	↓	Ctrl-X
Left	←	Ctrl-S
Right	→	Ctrl-D
Word left	Ctrl-←	Ctrl-A
Word right	Ctrl-→	Ctrl-F
Start of line	Home	Ctrl-Q Ctrl-S
End of line	End	Ctrl-Q Ctrl-D
Start of page	Ctrl-Home	Ctrl-Q Ctrl-E
End of page	Ctrl-End	Ctrl-Q Ctrl-X
Scroll up		Ctrl-W
Scroll down		Ctrl-Z
Page up	PgUp	Ctrl-R
Page down	PgDn	Ctrl-C
Start of text	Ctrl-PgUp	Ctrl-Q Ctrl-R
End of text	Ctrl-PgDn	Ctrl-Q Ctrl-C
Previous position		Ctrl-Q Ctrl-P
Goto line	Ctrl-F2	
Goto position	Shift-F2	
Goto blockstart		Ctrl-Q Ctrl-B
Goto blockend		Ctrl-Q Ctrl-K

Insert & Delete		
Insert new line		Ctrl-N
Backspace	Ctrl-H	Ctrl-H
Delete character	Del	Ctrl-G
Delete word		Ctrl-T
Delete to start of line		Ctrl-Q Ctrl-T
Delete to end of line		Ctrl-Q Ctrl-Y
Delete line	Ctrl-BackSpace	Ctrl-Y

Block functions		
Block select		Ctrl-K Ctrl-M
Copy block to pastebuffer		Ctrl-K Ctrl-I
Move block to pastebuffer		Ctrl-K Ctrl-Y
Paste	Ctrl-F7	Ctrl-U
Mark blockstart		Ctrl-K Ctrl-B
Mark blockend		Ctrl-K Ctrl-K
WordStar show/hide block		Ctrl-K Ctrl-H
WordStar copy block		Ctrl-K Ctrl-C
WordStar move block		Ctrl-K Ctrl-V
MultiMate copy block	Ctrl-F5	
MultiMate move block	Alt-F6	
MultiMate delete block	Alt-F7	