**ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
”САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ имени академика С.П.КОРОЛЕВА”**

Программирование на языке ПРОЛОГ

С А М А Р А 2008

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
”САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ имени академика С.П.КОРОЛЕВА”

Программирование на языке ПРОЛОГ

Утверждено Редакционно-издательским советом университета
в качестве методических указаний к лабораторным работам

Самара
Издательство СГАУ
2008

УДК 004.89(075.8)
ББК 32.813

Рецензент д-р техн. наук, проф. А. Н. К о в а р ц е в

Программирование на языке ПРОЛОГ: метод. указания к лабораторным работам */ О. П. Солдатова, И.В. Лёзина.* – Самара: Изд-во Самар. гос. аэрокосм. ун-та, 2008. – 52 с.

Предназначены для студентов факультета заочного обучения, обучающихся по специальности «Автоматизированные системы обработки информации и управления», и содержат основные положения языка логического программирования ПРОЛОГ. Приведены примеры программ на языке Visual Prolog, задания для выполнения контрольных и лабораторных работ по курсу «Системы искусственного интеллекта» и даны рекомендации по запуску программ в среде Visual Prolog 5.2.

Данные методические указания будут также полезны студентам дневного и вечернего отделения факультета информатики при изучении курса «Системы искусственного интеллекта».

СОДЕРЖАНИЕ

1. ОСНОВНЫЕ ПОЛОЖЕНИЯ ЯЗЫКА ПРОГРАММИРОВАНИЯ ПРОЛОГ

1.1 Основы логического программирования

Язык программирования Пролог (PROgramming LOGic) предполагает получение решения задачи при помощи логического вывода из ранее известных фактов. Программа на языке Пролог не является последовательностью действий – она представляет собой набор фактов и правил, обеспечивающих получение логических заключений из данных фактов. Поэтому Пролог считается *декларативным* языком программирования.

Пролог базируется на *фразах (предложениях) Хорна*, являющихся подмножеством формальной системы, называемой *логикой предикатов.*

Пролог использует упрощенную версию синтаксиса логики предикатов, он прост для понимания и очень близок к естественному языку.

Пролог имеет механизм вывода, который основан на сопоставлении образцов. С помощью подбора ответов на запросы Пролог извлекает хранящуюся информацию. Пролог пытается ответить на запрос, запрашивая информацию, о которой уже известно, что она истинна.

Одной из важнейших особенностей Пролога является то, что он ищет не только ответ на поставленный вопрос, но и все возможные альтернативные решения. Вместо обычной работы программы на процедурном языке от начала и до конца Пролог может возвращаться назад и просматривать все остальные пути при решении всех частей задачи.

Программист на Прологе описывает *объекты* и *отношения*, а также *правила,* при которых эти отношения являются истинными.

Объекты рассуждения в Прологе называются *термами* – синтаксическими объектами одной из следующих категорий:

- константы,
- переменные,
- функции (составные термы или структуры), состоящие из имени функции и списка аргументов-термов, имена функций начинаются со строчной буквы.

Константа в Прологе служит для обозначения имен собственных и начинается *со строчной буквы*.

Переменная в Прологе служит для обозначения объекта, на который нельзя сослаться *по имени*.

Пролог не имеет оператора присваивания.

Переменные в Прологе инициализируются при сопоставлении с константами в фактах и правилах.

До инициализации переменная свободна, после присвоения ей значения она становится связанной. Переменная остается связанной только то время, которое необходимо для получения решения по запросу, затем Пролог освобождает ее и ищет другое решение.

Переменные в Прологе предназначены для установления соответствия между термами предикатов, действующих в пределах одной фразы (предложения), а не являются местом памяти для хранения данных. Переменная начинается с прописной буквы или знаков подчеркивания.

В Прологе программист свободен в выборе имен констант, переменных, функций и предикатов. Исключение составляют резервированные имена и числовые константы. Переменные от констант отличаются первой буквой имени: у констант она строчная, у переменных – заглавная буква или символ подчеркивания.

Область действия имени представляет собой часть программы, где это имя имеет один и тот же смысл:

- для переменной областью действия является предложение (факт, правило или цель), содержащее данную переменную;
- для остальных имен (констант, функций или предикатов) – вся программа.

Специальным знаком «_» обозначается анонимная переменная, которая используется тогда, когда конкретное значение переменной не существенно для данного предложения. Анонимные переменные не отличаются от обычных при поиске соответствий, но не принимают значений и не появляются в ответах. *Различные вхождения знака подчеркивания означают различные анонимные переменные.*

Отношения между объектами в Прологе называются фактами. Факт соответствует фразе Хорна, состоящей из одного положительного литерала.

Факт – это простейшая разновидность предложения Пролога.

Любой факт имеет соответствующее значение истинности и определяет отношение между термами.

Факт является простым предикатом, который записывается в виде функционального терма, состоящего из имени отношения и объектов, заключенных в круглые скобки, например:

мать(мария, анна).

отец(иван, анна).

Точка, стоящая после предиката, указывает на то, что рассматриваемое выражение является фактом.

Вторым типом предложений Пролога является вопрос или *цель. Цель* – это средство формулировки задачи, которую должна решать программа. Простой вопрос (цель) синтаксически является разновидностью факта, например:

Цель: мать (мария, юлия).

В данном случае программе задан вопрос, является ли мария матерью юлии. Если необходимо задать вопрос, кто является матерью юлии, то цель будет иметь следующий вид:

Цель: мать(X, юлия).

Сложные цели представляют собой конъюнкцию простых целей и имеют следующий вид:

Цель: Q_1, Q_2,…,Q_n, где запятая обозначает операцию конъюнкции, а Q_1, Q_2,…,Q_n – подцели главной цели.

Конъюнкция в Прологе истинна только при истинности всех компонент, однако, в отличие от логики, в Прологе учитывается *порядок оценки истинности компонент* (слева направо).

Пример 1.

Пусть задана семейная база данных (БД) при помощи перечисления родительских отношений в виде списка фактов:

мать(мария, анна).

мать(мария, юлия).

мать(анна, петр).

отец(иван, анна).

отец(иван, юлия).

Тогда вопрос, является ли иван дедом петра, можно задать в виде следующей цели:

Цель: отец(иван, X), мать(X, петр).

На самом деле БД Пролога включает не только факты, но и правила. Факты и правила представляют собой не множество, а список. Для получения ответа БД просматривается по порядку, то есть в порядке следования фактов и предикатов в тексте программы.

Цель достигнута, если в БД удалось найти факт или правило, который (которое) удовлетворяет предикату цели, то есть превращает его в истинное высказывание. В нашем примере первую подцель удовлетворяют фак-

ты *отец(иван, анна) и отец(иван, юлия).* Вторую подцель удовлетворяет факт *мать(анна, петр).* Следовательно, главная цель удовлетворена, переменная *X* связывается с константой *анна*.

Третьим типом предложения является *правило*. Правило позволяет вывести один факт из других фактов. Иными словами, правило – это заключение, для которого известно, что оно истинно, если одно или несколько других найденных заключений или фактов являются истинными.

Правила – это предложения вида

H: - P_1, P_2,..., P_n.

Символ «: -» читается как «если», предикат *H* называется заключением, а последовательность предикатов P_1, P_2,..., P_n называется посылками. Приведенное правило является аналогом хорновского дизъюнкта $\neg P_1 \vee \neg P_2, ..., \vee \neg P_n \vee H$. Заключение истинно, если истинны все посылки. В посылках переменные связаны квантором существования, а в заключении - квантором всеобщности.

Пример 2.

Добавим в БД примера 18 правила, задающие отношение «дед»:

мать(мария, анна).

мать(мария, юлия).

мать(анна, петр).

отец(иван, анна).

отец(иван, юлия).

дед (X, Y): - отец(X, Z), мать(Z, Y).

дед (X, Y): - отец(X, Z), отец(Z, Y).

Тогда вопрос, является ли иван дедом петра, можно задать в виде следующей цели:

Цель: дед(иван, петр).

Правила - самые общие предложения Пролога, факт является частным случаем правила без правой части, а цель – правило без левой части.

Все предложения для одного предиката связаны между собой отношением «или».

Очень часто правила в Прологе являются рекурсивными. Например, для нашей семейной БД предикат «предок» определяется рекурсивно:

предок(x, y): - мать(x, y).

предок(x, y): - отец(x, y).

предок(x, y): - мать(x, z), предок(z, y).

предок(x, y): - отец (x, z), предок(z, y).

Рекурсивное определение предиката обязательно должно содержать нерекурсивную часть, иначе оно будет логически некорректным и программа зациклится. Чтобы избежать зацикливания, следует также позаботиться о порядке выполнения предложений, поэтому практически полезно, а порой и необходимо придерживаться принципа: *«сначала нерекурсивные выражения».*

Программа на Прологе - это конечное множество предложений.

К комментарию на Прологе относится всё, что находится между знаками /* и */.

/* Это комментарий. */

Либо можно поставить в начале строки знак % и тогда вся строка будет считаться комментарием.

% И это тоже комментарий.

1.2 Использование дизъюнкции и отрицания

Чистый Пролог разрешает применять в правилах и целях только конъюнкцию, однако язык, используемый на практике, допускает применение дизъюнкции и отрицания в телах правил и целях. Для достижения цели, содержащей дизъюнкцию, Пролог–система сначала пытается удовлетворить левую часть дизъюнкции, а если это не удается, то переходит к поиску решения для правой части дизъюнкции. Аналогичные действия производятся при выполнении тела правил, содержащих дизъюнкцию. Для обозначения дизъюнкции используется символ « ; ».

В Прологе отрицание имеет имя *«not»* и для представления отрицания какого-либо предиката *P* используется запись *not(P). Цель not(P) достижима тогда и только тогда, когда не удовлетворяется предикат (цель) P.* При этом переменным значения не присваиваются. В самом деле, если достигается *P,* то не достигается *not(P),* значит надо стереть все присваивания, приводящие к данному результату. Наоборот, если *P* не достигается, то переменные не принимают никаких значений.

1.3 Управление поиском решения

Встроенный в Пролог механизм поиска с возвратом может привести к поиску ненужных решений, в результате чего снижается эффективность программы в случае, если надо найти только одно решение. В других случаях бывает необходимо продолжить поиск, даже если решение найдено.

Пролог обеспечивает два встроенных предиката, которые дают возможность управлять механизмом поиска с возвратом: предикат *fail* – используется для инициализации поиска с возвратом и предикат *отсечения !* – используется для запрета возврата.

Предикат fail всегда имеет ложное значение!

Пример 3: Использование предиката fail. Для примера 1 можно добавить правило для печати всех матерей, которые есть в БД:

печать_матерей:-мать(X,Y), write(X," есть мать",Y),nl,fail.

goal

печать_матерей.

В результате будет выдано 3 решения:

X=мария, Y= анна.

X=мария, Y= юлия.

X=анна, Y= петр.

Отсечение так же, как и *fail,* помещается в тело правила. *Однако в отличие от fail предикат отсечения имеет всегда истинное значение.*

При этом выполняется обращение к другим предикатам в теле правила, следующим за отсечением. Следует иметь в виду, что невозможно произвести возврат к предикатам, расположенным в теле правила перед отсечением, а также невозможен возврат к другим правилам данного предиката.

Существует только два случая применения предиката отсечения:

1. Если заранее известно, что определенные посылки никогда не приведут к осмысленным решениям – это так называемое «зеленое отсечение».
2. Если отсечения требует сама логика программы для исключения альтернативных подцелей – это так называемое «красное отсечение».

1.4 Процедурность Пролога

Пролог – декларативный язык. Описывая задачу в терминах фактов и правил, программист предоставляет Прологу самому искать способ решения. В процедурных языках программист должен сам писать процедуры и функции, которые подробно «объясняют» компьютеру, какие шаги надо сделать для решения задачи.

Тем не менее, рассмотрим Пролог с точки зрения процедурного программирования:

1. Факты и правила можно рассматривать как определения процедур.
2. Использование правил для условного ветвления программы. Правило, в отличие от процедуры, позволяет задавать множество альтернативных определений одной и той же процедуры. Поэтому правило можно считать аналогом оператора *case* в Паскале.
3. В правиле может быть выполнено сравнение, как в условных операторах.
4. Отсечение можно считать аналогом *go to*.
5. Возврат вычисленного значения производится аналогично процедурам. В Прологе это делается путем связывания свободных переменных при сопоставлении цели с фактами и правилами.

2. СТРУКТУРА ПРОГРАММЫ НА ЯЗЫКЕ ПРОЛОГ

Программа, написанная на Прологе, состоит из пяти основных разделов: описания доменов, базы данных, описания предикатов, описания предложений, описания цели. Ключевые слова *domains, constants, database (facts), predicates, clauses, goal* отмечают начала соответствующих разделов. Назначение этих разделов таково:

- *domains* содержит определения доменов, которые описывают различные типы данных, используемых в программе;
- *constants* используется для объявления символических констант, используемых в программе;
- *database (facts)* содержит описания предикатов внутренней базы данных Пролога, если программа такой базы данных не требует, то этот раздел может быть опущен;
- *predicates* служит для описания предикатов, не принадлежащих внутренней базе данных;
- в *clauses* заносятся факты и правила самой программы;
- в *goal* на языке Пролог формулируется назначение создаваемой программы. Составными частями при этом могут являться некие подцели, из которых формируется единая цель программы.

В Visual Prolog разрешает объявление разделов domains, facts, predicates, clauses как глобальных разделов, то есть с ключевым словом global.

Пролог имеет следующие встроенные типы доменов:

Тип данных	**Ключевое слово**	**Диапазон значений**	**Примеры использования**
Символы	char	Все возможные символы	‘a’, ’b’, ‘#’, ‘B’, ‘%’
Целые числа	integer byte word dword	От –32768 до 32767 От 0 до 255 От 0 до 65535 От 0 до 2^{32}	-63, 84, 2349
Действительные числа	real short ushort long ulong unsigned	От +1E-307 до +1E308 16 битов со знаком 16 битов без знака 32 бита со знаком 32 бита без знака 16 или 32 бита без знака	360, - 8324, 1.25E23, 5.15E-9
Строки	string	Последовательность символов (не более 250)	«today», «123», «school_day»
Символические имена	symbol	1.Последовательность букв, цифр, символов подчеркивания; первый символ – строчная буква. 2. Последовательность любых символов, заключенная в кавычки.	flower, school_day «string and symbol»
Ссылочный тип	ref		
Файлы	file	Допустимое в DOS имя файла	mail.txt, LAB.PRO

Если в программе необходимо использовать новые домены данных, то они должны быть описаны в разделе *domains*.

Пример 4:

```
domains
number=integer
name, person=symbol.
```

Различие между *symbol* и *string* - в машинном представлении и выполнении, синтаксически они не различимы.

Visual Prolog выполняет автоматическое преобразование типов между доменами *string* и *symbol*. Однако по принятому соглашению символическую строку в двойных кавычках нужно рассматривать как *string,* а без кавычек – как *symbol:*

Symbol - имена, начинающиеся с символа нижнего регистра и содержащие только символы, цифры и символы подчеркивания.

String – в двойных кавычках могут содержать любую комбинацию символов, кроме #0, который отмечает конец строки.

Visual Prolog поддерживает и другие типы стандартных доменов данных, например, для работы с внешними БД или объектами.

Предикаты описываются в разделе *predicates*. Предикат представляет собой строку символов, первым из которых является строчная буква. Предикаты могут не иметь аргументов, например «go» или «repeat». Если предикаты имеют аргументы, то они определяются при описании предикатов в разделе predicates.

Пример 5:
predicates
mother (symbol, symbol)
father (symbol, symbol).

Факты и правила определяются в разделе *clauses,* а вопрос к программе задается в разделе *goal* – в этом случае цель называется *внутренней целью*. Программа на Турбо-Прологе может не содержать раздел *goal,* в этом случае цель задается в процессе работы программы и является *внешней целью*. Для задания внешней цели в окне *Dialog* будет выведено приглашение *Goal.* После удовлетворения внешней цели программа на Турбо-Прологе не заканчивает свою работу, а просит ввести следующую цель, таким образом можно задать программе несколько различных целей. Если цель в программе является внутренней целью, то процесс вычисления остановится после первого ее успешного вычисления. Если цель в программе является внешней целью, то процесс вычисления цели повторяется до тех пор, пока не будут найдены все успешные способы вычисления цели.

В Visual Prolog раздел goal в тексте программы является обязательным. Разница в режимах исполнения программы состоит в разном использовании утилиты Test Goal. Если утилита создается для запуска любой программы, то при этом ищутся все решения, если утилита создается для автономного запуска программы, то ищется одно решение.

3. ИСПОЛЬЗОВАНИЕ СПИСКОВ В ПРОЛОГЕ

Список – это упорядоченный набор объектов одного и того же типа. Элементами списка могут быть целые числа, действительные числа, символы, строки, символические имена и структуры. Порядок расположения элементов в списке играет важную роль: те же самые элементы списка, упорядоченные иным способом, представляют уже совсем другой список.

Совокупность элементов списка заключается в квадратные скобки ([]), элементы друг от друга отделяются запятыми. Список может содержать произвольное число элементов, единственным ограничением является объем оперативной памяти. Количество элементов в списке называется его длиной. Список может содержать один элемент и даже не содержать ни одного элемента. Список, не содержащий элементов, называется пустым, или нулевым списком.

Непустой список можно рассматривать как список, состоящий из двух частей: головы – первого элемента списка; хвоста – остальной части списка. Голова является элементом списка, хвост – списком. Голова списка – это неделимое значение, хвост представляет собой список, составленный из того, что осталось от исходного списка в результате «отделения головы». Этот новый список обычно можно делить и дальше. Если список состоит из одного элемента, то его можно разделить на голову, которой будет этот самый элемент, и хвост, являющийся пустым списком. Пустой список нельзя разделить на голову и хвост.

Операция деления списка на голову и хвост обозначается при помощи вертикальной черты (|):

[Head | Tail].

Head здесь является переменной для обозначения головы списка, переменная Tail обозначает хвост списка (для имен головы и хвоста списка пригодны любые допустимые Прологом имена). Данная операция также присоединяет элемент в начало списка Например, для того, чтобы присоединить X к списку S, следует написать [X | S].

Отличительной особенностью описания списков является наличие звездочки (*) после имени домена элементов.

Пример 6: объявление списков, состоящих из элементов стандартных типов доменов или типа структуры.

```
domains
list1=integer*
list2=char*
list3=string*
list4=real*
list5=symbol*
personal_library = book (title, author, publisher, year)
list6= personal_library*
list7=list1*
list8=list5*
```

В первых пяти объявлениях списков в качестве элементов используются стандартные домены данных, в шестом типе списка в качестве элемента используется домен структуры personal_library, в седьмом и восьмом типе списка в качестве элемента используется ранее объявленный список.

Пример 7: демонстрация разделения списков на голову и хвост.

Список	*Голова*	*Хвост*
[1, 2, 3, 4, 5]	*1*	*[2, 3, 4, 5]*
[6.9, 4.3]	*6.9*	*[4.3]*
[cat, dog, horse]	*Cat*	*[dog, horse]*
['S', 'K', 'Y']	*'S'*	*['K', 'Y']*
[«PIG»]	*«PIG»*	*[]*
[]	*Не определена*	*Не определен*

4. ПРИМЕНЕНИЕ СПИСКОВ В ПРОГРАММАХ

4.1 Поиск элемента в списке

Для применения списков в программах на Прологе необходимо описать домен списка в разделе *domains*, предикаты, работающие со списками, необходимо описать в разделе *predicates*, задать сам список можно либо в разделе *clauses,* либо в разделе *goal*.

Над списками можно реализовать различные операции: поиск элемента в списке, разделение списка на два списка, присоединение одного списка к другому, удаление элементов из списка, сортировку списка, создание списка из содержимого БД и так далее.

Поиск элемента в списке является очень распространенной операцией. Поиск представляет собой просмотр списка на предмет выявления соответствия между объектом поиска и элементом списка. Если такое соответствие найдено, то поиск заканчивается успехом, в противном случае поиск заканчивается неуспехом. Стратегия поиска при этом будет состоять в рекурсивном выделении головы списка и сравнении ее с объектом поиска.

Пример 8: поиск элемента в списке.

```
domains
list=integer*
predicates
member (integer, list)
clauses
```

member (Head, [Head | _]).
member (Head, [_ | Tail]):- member (Head, Tail).
goal
member (3, [1, 4, 3, 2]).

Правило поиска может сравнить объект поиска и голову текущего списка, эта операция записана в виде факта предиката member. Этот вариант предполагает наличие соответствия между объектом поиска и головой списка. Отметим, что хвост списка в данном случае не важен, поэтому хвост списка присваивается анонимной переменной. Если объект поиска и голова списка различны, то в результате исполнения первого предложения будет неуспех, происходит возврат и поиск другого правила или факта, с которыми можно попытаться найти соответствие. Для этой цели служит второе предложение, которое выделяет из списка следующий по порядку элемент, то есть выделяет голову текущего хвоста, поэтому текущий хвост представляется как новый список, голову которого можно сравнить с объектом поиска. В случае исполнения второго предложения голова текущего списка ставится в соответствие анонимной переменной, так как значение головы списка в данном случае не играет никакой роли.

Процесс повторяется до тех пор, пока первое предложение даст успех в случае установления соответствия, либо неуспех – в случае исчерпания списка. В представленном примере предикат *find* находит все совпадения объекта поиска с элементами списка. Для того, чтобы найти только первое совпадение, следует модифицировать первое предложение следующим образом:

member (Head, [Head |_]):- !.

Отсечение отменяет действие механизма возврата, поэтому поиск альтернативных успешных решений реализован не будет.

4.2 Объединение двух списков

Слияние двух списков и получение таким образом третьего списка принадлежит к числу наиболее полезных при работе со списками операций. Обозначим первый список L1, а второй список - L2. Пусть L1 = [1, 2, 3], а L2 = [4, 5]. Предикат append присоединяет L2 к L1 и создает выходной список L3, в который он должен переслать все элементы L1 и L2. Весь процесс можно представить следующим образом:

1. Список L3 вначале пуст.
2. Элементы списка L1 пересылаются в L3, теперь значением L3 будет [1, 2, 3].

3. Элементы списка L2 пересылаются в L3, в результате чего тот принимает значение [1, 2, 3, 4, 5].

Тогда программа на языке Пролог имеет следующий вид:

Пример 9: объединение двух списков.

```
domains
list=integer*
predicates
append (list, list, list)
clauses
append ( [], L2, L2).
append ([H|T1], L2, [H|T3 ]):- append (T1, L2, T3).
goal
append ( [1, 2, 3], [4, 5], L3).
```

Основное использование предиката *append* состоит в объединении двух списков, что делается при помощи задания цели вида *append ([1, 2, 3], [4, 5], L3)*. Поиск ответа на вопрос типа: *append (L1, [3, 4, 5], [1, 2, 3, 4, 5])* – сводится к поиску такого списка *L1=[1, 2]*, который при слиянии со списком *L2 = [3, 4, 5]* даст список *L3 = [1, 2, 3, 4, 5]*. При обработке *цели append (L1, L2, [1, 2, 3, 4, 5])* ищутся такие списки *L1* и *L2*, что их объединение даст список *L3 = [1, 2, 3, 4, 5]*.

4.3 Сортировка списков

Сортировка представляет собой переупорядочение элементов списка определенным образом. Назначением сортировки является упрощение доступа к нужным элементам. Для сортировки списков обычно применяются три метода:

- метод перестановки,
- метод вставки,
- метод выборки.

Также можно использовать комбинации указанных методов.

Первый метод сортировки заключается в перестановке элементов списка до тех пор, пока он не будет упорядочен. Второй метод осуществляется при помощи неоднократной вставки элементов в список до тех пор, пока он не будет упорядочен. Третий метод включает в себя многократную выборку и перемещение элементов списка.

Второй из методов, метод вставки, особенно удобен для реализации на Прологе.

Пример 10: сортировка списков методом вставки.

```
domains
list=integer*
predicates
insert_sort (list, list)
insert (integer, list, list)
asc_order (integer, integer)
clauses
insert_sort ( [], []).
insert_sort ([H1|T1], L2):- insert_sort (T1, T2),
                            insert(H1, T2, L2).
insert (X, [H1| T1], [H1| T2]) :- asc_order (X, H1), !,
                                  insert (X, T1, T2).
insert (X, L1, [X| L1]).
asc_order (X, Y):- X>Y.
goal
insert_sort ([4, 7, 3, 9], L).
```

Для удовлетворения первого правила оба списка должны быть пустыми. Для того, чтобы достичь этого состояния, по второму правилу происходит рекурсивный вызов предиката *insert_sort,* при этом значениями *H1* последовательно становятся все элементы исходного списка, которые затем помещаются в стек. В результате исходный список становится пустым и по первому правилу выходной список также становится пустым.

После того, как произошло успешное завершение первого правила, Пролог пытается выполнить второй предикат *insert,* вызов которого содержится в теле второго правила. Переменной *H1* сначала присваивается первое взятое из стека значение *9,* а предикат принимает вид *insert (9, [], [9]).*

Так как теперь удовлетворено все второе правило, то происходит возврат на один шаг рекурсии в предикате *insert_sort.* Из стека извлекается 3 и по третьему правилу вызывается предикат *asc_order,* то есть происходит попытка удовлетворения пятого правила *asc_order (3, 9):- 3>9.* Так как данное правило заканчивается неуспешно, то неуспешно заканчивается и третье правило, следовательно удовлетворяется четвертое правило и *3* вставляется в выходной список слева от *9*: *insert (3, [9], [3, 9]).*

Далее происходит возврат к предикату *insert_sort*, который принимает следующий вид: *insert_sort ([3, 9], [3, 9]).*

На следующем шаге рекурсии из стека извлекается 7 и по третьему правилу вызывается предикат *asc_order* в виде *asc_order (7, 3):- 7>3.* Так как данное правило заканчивается успешно, то элемент 3 убирается в стек и *insert* вызвается рекурсивно еще раз, но уже с хвостом списка – *[9]: insert (7, [9], _).* Так как правило *asc_order (7, 9):- 7>9* заканчивается неуспешно, то выполняется четвертое правило, происходит возврат на предыдущие шаги рекурсии сначала *insert,* затем *insert_sort.*

В результате *7* помещается в выходной список между элементами *3* и *9*: *insert (7, [3, 9], [3, 7, 9]).*

При возврате еще на один шаг рекурсии из стека извлекается 4 и по третьему правилу вызывается предикат *asc_order* в виде *asc_order (4, 3):- 4>3.* Так как данное правило заканчивается успешно, то элемент 3 убирается в стек и *insert* вызывается рекурсивно еще раз, но уже с хвостом списка – *[7, 9]: insert (4, [7, 9], _).* Так как правило *asc_order (4, 7):- 4>7* заканчивается неуспешно, то выполняется четвертое правило, происходит возврат на предыдущие шаги рекурсии сначала *insert,* затем *insert_sort.*

В результате *4* помещается в выходной список между элементами *3* и *7*:

insert (4, [3, 7, 9], [3, 4, 7, 9]).
insert_sort [4, 7, 3, 9], [3, 4, 7, 9]).

4.4 Компоновка данных в список

Иногда при программировании определенных задач возникает необходимость собрать данные из фактов БД в список для последующей их обработки. Пролог содержит встроенный предикат *findall*, который позволяет выполнить данную операцию. Описание предиката *findall* выглядит следующим образом:

Findall (Var_, Predicate_, List_), где Var_ обозначает имя для терма предиката Predicate_, в соответствии с типом которого формируются элементы списка List_.

Пример 11: использование предиката findall.

```
domains
d=integer
predicates
decimal (d)
write_decimal
clauses
decimal (0).
```

decimal (1).
decimal (2).
decimal (3).
decimal (4).
decimal (5).
decimal (6).
decimal (7).
decimal (8).
decimal (9).
write_decimal:- findall(C, decimal (C), L), write (L).
goal
write_decimal.

5. ИСПОЛЬЗОВАНИЕ СОСТАВНЫХ ТЕРМОВ

В Прологе функциональный терм или предикат можно рассматривать как структуру данных, подобную записи в языке Паскаль. Терм, представляющий совокупность термов, называется составным термом. Предикаты, записанные в виде составного терма, называются составной структурой данных. Составные структуры данных в Турбо-Прологе объявляются в разделе *domains.* Если термы структуры относятся к одному и тому же типу доменов, то этот объект называется *однодоменной структурой данных*. Если термы структуры относятся к разным типам доменов, то такая структура данных называется *многодоменной структурой данных*. Использование доменной структуры упрощает структуру предиката.

Пример 12: Необходимо создать БД, содержащую сведения о книгах из личной библиотеки. Зададим составной терм с именем personal_library, имеющим следующую структуру: personal_library= book (title, author, publisher, year), и предикат collection (collector, personal_library). Терм book называется функтором структуры данных. Пример программы, использующей составные термы для описания личной библиотеки и поиска информации о книгах, напечатанных в 1990 году, выглядит следующим образом:

domains
collector, title, author, publisher = symbol
year = integer
personal_library = book (title, author, publisher, year)
predicates

collection (collector, personal_library)

clauses

collection (irina, book («Using Turbo Prolog», «Yin with Solomon», «Moscow, World», 1993)).

collection (petr, book («The art of Prolog», «Sterling with Shapiro», «Moscow, World», 1990)).

collection (anna, book («Prolog: a relation language and its applications», «John Malpas», «Moscow, Science», 1990)).

goal

collection (X, book(Y,_, _, 1990)

Представление данных часто требует наличия большого числа структур. В Прологе эти структуры должны быть описаны. Для более компактного описания структур данных в Прологе предлагается использование альтернативных описаний доменов.

Пример 13: Необходимо создать БД, содержащую сведения о книгах и аудиозаписях из личной библиотеки.

domains

person, title, author, artist, album, type = symbol

thing = book (title, author); record (artist, album, type)

predicates

owns (person, thing)

clauses

owns (irina, book («Using Turbo Prolog», «Yin with Solomon»)).

owns (petr, book («The art of Prolog», «Sterling with Shapiro»)).

owns (anna, book («Prolog: a relation language and its applications», «John Malpas»)).

owns (irina, record («Elton John», «Ice Fair», «popular»)).

owns (petr, record («Benny Goodman», «The King of Swing», «jazz»)).

owns (anna record («Madonna», «Madonna», «popular»»)).

goal

owns (X, record(_, _, «jazz»)

Пример 14:

Создать базу данных о заданной предметной области в виде множества фактов языка Пролог (не менее 5 фактов). Информацию о каждом компоненте БД представить в виде структуры. Разработать набор предикатов, осуществляющих взаимодействие с БД, при помощи кото-

рых можно реализовать все типы запросов, приведенные в варианте задания.

Предметная область – база данных товаров мебельного магазина. Каждый магазин может быть описан структурой: название магазина, адрес, фамилия директора, список имеющихся товаров. Каждый товар может быть описан структурой: название товара, цена, страна-производитель, список имеющихся расцветок.

Реализовать следующие запросы:

1. *Найти адрес магазина, директором которого является человек с заданной фамилией.*
2. *Найти названия всех магазинов, продающих заданный товар.*
3. *Найти название товара, имеющего максимальную цену.*

В приведенном ниже примере используются следующие домены:

list_colours - список расцветок, *list_products* - список имеющихся товаров, *name_shop* - название магазина, *address* –адрес магазина, *chief* - фамилия директора, *name_product* – название товара, *country* - страна-производитель, *price* – цена.

shop – предикат, описывающий заданную предметную область; *q1,q2, q3* – предикаты, реализующие первый, второй и третий запрос соответственно; *q21, q31*–вспомогательные предикаты для второго и третьего запросов; *max* – динамический предикат, необходимый для третьего запроса.

```
domains
        list_colours = symbol*
        list_products = product*
        name_shop, address, chief, name_product, country = symbol
        price =real
        product = prod (name_product, price, country, list_colours)
facts
        max(name_product, price)
predicates
        shop (name_shop, address, chief, list_products)
        q1(chief)
        q2(name_product)
        q21(name_product, list_products)
        q3
        q31(list_products)
clauses
```

```
shop("Company1","Debenko_5","Ivanov",[prod(table,3000,"Italy",[white
, red, black]),prod(chair,4000,"Japan",[brown, grey])]).
shop("Company2","S.Lazo_4","Petrov",[prod(sofa,9000,"China",[brown,
black]),prod(chair,5000,"England",[red, grey])]).
shop("Company3","Sadovaya_3","Sidorov",[prod(sofa,19000,"Mexico",
[black, white])]).
shop("Company4","Dachnaya_2","Orlov",[prod(bookcase,7000,"Russia"
, [brown, black]),prod(table,6000,"Spain",[red,grey])]).
shop("Company5","Pobeda_1","Galkin",[prod(bed,5000,"China",[brown
, black]),prod(arm_chair,21000,"Japan",[red, grey])]).

/* первый запрос: обращаемся к предикату shop, проверяем, совпадает ли фамилия директора данного магазина с заданной фамилией, для этого используется одна и та же переменная F, в случае совпадения печатаем адрес магазина A.*/
q1(F):-shop (_,A,F,_),write(A),nl.

/* второй запрос: обращаемся к предикату shop, извлекаем название магазина N и список имеющихся товаров L; вызываем вспомогательный предикат q21, которому передаем название заданного товара P и список имеющихся товаров L; q21 будет истинен, если заданный товар есть в списке имеющихся товаров, в этом случае выводим на экран название магазина. */
q2(P):-shop(N,_,_,L),q21(P,L),write(N),nl,fail.
q21(H,[H1|_]):-H1=prod(H,_,_,_).
q21(H,[_|T]):-q21(H,T).

/* третий запрос: записываем в динамический предикат max любое название товара, например, a и его цену 0; так как мы ищем название товара, имеющего максимальную цену, соответственно первоначальная цена должна быть маленькая; чтобы обязательно был товар с более высокой ценой, обращаемся к предикату shop, извлекаем список имеющихся товаров L, вызываем вспомогательный предикат q31, которому передаем список имеющихся товаров L; q31 проверяет, имеет ли очередной товар из списка имеющихся товаров цену, более высокую, чем цена, записанная в предикате max; в этом случае предикатом retract удаляем старые название товара и цену, предикатом assert записываем новое название товара и цену, иначе – просто переходим к следующему товару. */
```

```
max(a,0).
q3:-shop(_,_,_,L),q31(L),fail.
q31([]).
q31([H1|T]):-H1=prod(H,P,_,_), max(A,B), P>B, retract(max(A,B)), assert(max(H,P)), q31(T).
q31([H1|T]):-H1=prod(_,P,_,_),max(_,B),P<=B,q31(T).
goal
        % вызываем каждый запрос по очереди.
        q1("Petrov").
        %q2(table).
        %q3;max(A,B).
```

Пример 15: Решить предыдущую задачу с использованием внутренней базы данных. Найденные решения записать в виде фактов внутренней базы данных Пролога. Предусмотреть проверку факта, являющегося ответом на запрос в БД. Если такой факт существует, то выдать его в качестве ответа на запрос. Если такого факта не существует во внутренней базе данных, то запустить запрос на выполнение и записать результат в БД.

```
domains
        list_colours = symbol*
        list_products = product*
        name_shop,address,chief,name_product,country = symbol
        price =real
        product = prod (name_product,price,country,list_colours)
facts
        max(name_product,price)
        % динамические предикаты для второго и третьего запроса
        dq1(address)
        dq2(name_shop)
predicates
        shop (name_shop,address,chief,list_products)
        q1(chief)
        q2(name_product)
        q21(name_product,list_products)
        q3
        q31(list_products)
clauses
```

```
shop("Company1","Debenko_5","Ivanov",[prod(table,3000,"Italy",[white,
red,black]),prod(chair,4000,"Japan",[brown,grey])]).
shop("Company2","S.Lazo_4","Petrov",[prod(sofa,9000,"China",[brown,
black]),prod(chair,5000,"England",[red,grey])]).
shop("Company3","Sadovaya_3","Sidorov",[prod(sofa,19000,"Mexico",
[black,white])]).
shop("Company4","Dachnaya_2","Orlov",[prod(bookcase,7000,"Russia",
[brown,black]),prod(table,6000,"Spain",[red,grey])]).
shop("Company5","Pobeda_1","Galkin",[prod(bed,5000,"China",[brown,
black]),prod(arm_chair,21000,"Japan",[red,grey])]).

/* Проверяем наличие факта, являющегося ответом на запрос в БД,
если такой факт существует, то выдаём его в качестве ответа на за-
прос, печатаем перед ним "*". */
q1(_):-dq1(A), write("*",A),nl.
/* Если такого факта не существует во внутренней базе данных,
то запускаем запрос на выполнение и записываем  результат в БД. */
q1(F):-not(dq1(_)), shop (_,A,F,_),write(A),nl,assert(dq1(A)).

q2(_):-dq2(A), write("*",A),nl,fail.
q2(P):-not(dq2(_)), shop(N,_,_,L),q21(P,L),write(N),nl,assert(dq2(N)),fail.
q21(H,[H1|_]):-H1=prod(H,_,_,_).
q21(H,[_|T]):-q21(H,T).

max(a,0).
q3:-shop(_,_,_,L),q31(L),fail.
q31([]).
q31([H1|T]):-
H1=prod(H,P,_,_),max(A,B),P>B,retract(max(A,B)),assert(max(H,P)),q31(T).
q31([H1|T]):-H1=prod(_,P,_,_),max(_,B),P<=B,q31(T).
goal
    /* Запускаем каждый запрос на исполнение два раза - первый
    раз программа находит ответ на запрос, а второй раз выдает от-
    вет из БД. */
    q1("Petrov"),q1("Petrov").
    %q2(table);q2(table).
    %q3;max(A,B).
```

Пример 16: Решить логическую головоломку. В пяти домах, окрашенных в разные цвета, обитают мужчины разных национальностей. Они держат разных животных, предпочитают разные напитки и курят сигареты разных марок. Известно, что англичанин живет в красном доме; у испанца есть собака; кофе пьют в зеленом доме; украинец пьет чай; зеленый дом – первый по правую руку от дома цвета слоновой кости; курильщик «Уинстона» держит улиток; сигареты «Кул» курят в жёлтом доме; молоко пьют в среднем доме; норвежец живет в крайнем слева доме; мужчина, курящий «Честерфилд», живет в доме, соседнем с домом мужчины, у которого есть лиса; сигареты «Кул» курят в доме, соседнем с домом, где имеется лошадь; мужчина, предпочитающий «Лаки страйк», пьёт апельсиновый сок; японец курит сигареты «Парламент»; норвежец живет в доме рядом с голубым домом, у одного из мужчин есть зебра, один из мужчин пьет воду. Кто в каком доме живет, какое животное держит, что пьёт и курит?

```
domains
        /* описание дома - цвет дома, национальность живущего там
мужчины, его животное, напиток и марка сигарет.*/
        house=h(colour,nationality,pet,drink,cigarette)
        houses=house*
        colour,nationality,pet,drink,cigarette=symbol
predicates
        colour(house, colour)
        nationality(house, nationality)
        pet(house, pet)
        drink(house, drink)
        cigarette(house, cigarette)
        first(house,houses)
        middle(house,houses)
        solve
        neighbourhood(house, house,houses)
        neighbourhoodright(house, house,houses)
clauses
        % в описании дома цвет дома находится на первом месте
        colour(h(C,_,_,_,_),C).
        /* в описании дома национальность живущего там мужчины
находится на втором месте*/
```

```
nationality(h(_,N,_,_,_),N).
% в описании дома животное находится на третьем месте
pet(h(_,_,P,_,_),P).
% в описании дома напиток находится на четвертом месте
drink (h(_,_,_,D,_),D).
% в описании дома сигареты находятся на пятом месте
cigarette(h(_,_,_,_,S),S).
%один из домов - крайний слева
first(X,[X,_,_,_,_]).
% один из домов находится в середине
middle(Y,[_,_,Y,_,_]).
% перебор всех возможных соседних домов
neighbourhood(A,B,[A,B,_,_,_]).
neighbourhood(A,B,[B,A,_,_,_]).
neighbourhood(A,B,[_,A,B,_,_]).
neighbourhood(A,B,[_,B,A,_,_]).
neighbourhood(A,B,[_,_,A,B,_]).
neighbourhood(A,B,[_,_,B,A,_]).
neighbourhood(A,B,[_,_,_, A,B]).
neighbourhood(A,B,[_,_,_, B,A]).
/* перебор всех возможных соседних домов, из которых дом B
по правую руку от дома A */
neighbourhoodright (A,B,[A,B,_,_,_]).
neighbourhoodright (A,B,[_,A,B,_,_]).
neighbourhoodright (A,B,[_,_,A,B,_]).
neighbourhoodright (A,B,[_,_,_, A,B]).
% англичанин живет в красном доме
solve:- neighbourhood(H1,_,Houses), nationality(H1,englishman),
colour(H1,red),
%; у испанца есть собака
neighbourhood(H2,_,Houses), nationality(H2, spaniard),
pet(H2,dog),
% кофе пьют в зеленом доме
neighbourhood(H3,_,Houses), colour(H3,green), drink(H3, coffee),
% украинец пьет чай
neighbourhood(H4,_,Houses), nationality(H4, ukrainian),
drink(H4, tea),
```

```
    /* зеленый дом – первый по правую руку от дома цвета слоновой кости */
    neighbourhoodright(H5,H6,Houses), colour(H6,green),
    colour(H5, ivory),
    % курильщик «Уинстона» держит улиток
    neighbourhood(H7,_,Houses), cigarette(H7,winston), pet(H7, snail),
    % сигареты «Кул» курят в жёлтом доме;
    neighbourhood(H8,_,Houses), cigarette(H8,cool), colour(H8, yellow),
    % молоко пьют в среднем доме
    middle(H9,Houses), drink(H9, milk),
    % норвежец живет в крайнем слева доме;
    first(H10, Houses), nationality(H10, norwegian),
    /* мужчина, курящий «Честерфилд», живет в доме, соседнем с домом мужчины, у которого есть лиса */
    neighbourhood(H11,H12,Houses),
    cigarette(H11,chesterfild),pet(H12,fox),
    /* сигареты «Кул» курят в доме, соседнем с домом, где имеется лошадь */
    neighbourhood(H13,H14,Houses),
    cigarette(H13,cool),pet(H14,horse),
    /* мужчина, предпочитающий «Лаки страйк», пьёт апельсиновый сок */
    neighbourhood(H15,_,Houses),cigarette(H15,lakestrike),
    drink(H15,orange),
    % японец курит сигареты «Парламент»
    neighbourhood(H16,_,Houses),nationality(H16,japanese),
    cigarette(H16,parlament),
    % норвежец живет в доме рядом с голубым домом
    neighbourhood(H17,H18,Houses),nationality(H17,norwegian),
    colour(H18,blue),
    % у одного из мужчин есть зебра
    neighbourhood(H19,_,Houses), pet(H19,zebra),
    % один из мужчин пьет воду
    neighbourhood(H20,_,Houses), drink(H20,water),
    write(Houses).

goal
    solve.
```

6. ОСНОВЫ РАБОТЫ В VISUAL PROLOG

6.1. Создание TestGoal –проекта для выполнения программ

Для использования утилиты TestGoal для выполнения программ требуется определить некоторые опции компилятора Visual Prolog. Для этого необходимо выполнить следующие действия:

1. Запустите среду визуальной разработки Visual Prolog.
2. Создайте новый проект: выберите команду Project | New Project, активизируется диалоговое окно Application Expert.
3. Определите базовый каталог и имя проекта, рекомендуется имя в поле Base Directory задать по следующему образцу: C:\student\641\ivanov\TestGoal, а имя в поле Project Name следует определить как TestGoal. Далее следует установить флажок Multi-programmer Mode и щелкнуть мышью внутри полей Name of.VPR File и Name of.PRJ File. Определите цель проекта: на вкладке Target рекомендуется выбрать следующие параметры: Windows32, Easywin, exe, Prolog.
4. Установите требуемые опции компилятора для созданного Test-Goal-проекта, для этого следует активизировать диалоговое окно Compiler Options при помощи команды Options | Project | Compiler Options. Далее откройте вкладку Warnings и выполните следующие действия:
    - Установите переключатель Nondeterm. Это нужно для того, чтобы компилятор принимал по умолчанию, что все определенные пользователем предикаты – недетерминированные (могут породить более одного решения).
    - Снимите флажки Non Quoted Symbols, Strong Type Conversion Check, Check Type of Predicates.
    - Нажмите кнопку OK, чтобы сохранить установки опций компилятора.

6.2. Запуск и тестирование программы

Для проверки правильности настройки системы, создадим простую тестовую программу. Для создания нового окна редактирования можно

использовать команду меню File | New. Редактор среды визуальной разработки – стандартный текстовый редактор.

Введите в окне редактирования следующий текст:

Goal write (“Hello world!”),nl.

Это раздел цели программы, и для того, чтобы она могла быть выполнена, следует активизировать команду Project | Test Goal или нажать комбинацию клавиш <Ctrl> + <G>.

Результат выполнения программы будет расположен вверху в отдельном окне, которое необходимо закрыть перед тем, как тестировать другую Goal.

Утилита Test Goal интерпретирует Goal как специальную программу, которая компилируется, генерируется в исполняемый файл и Test Goal запускает его на выполнение. Эта утилита внутренне расширяет заданный код Goal, чтобы сгенерированная программа находила все возможные решения и показывала значения всех используемых переменных.

Замечание: утилита Test Goal компилирует только тот код, который определен в активном окне редактора.

7. ЗАДАНИЯ ДЛЯ ЛАБОРАТОРНЫХ РАБОТ

7.1 Задания для лабораторной работы на тему: работа со списками

Вариант №1

Написать программу вычисления номера элемента в списке.

Вариант №2

Написать программу замены всех вхождений элемента в список на заданную константу.

Вариант №3

Написать программу замены N-го элемента в списке на заданную константу.

Вариант №4

Написать программу удаления N-го элемента в списке.

Вариант №5

Написать программу удаления M элементов из списка, начиная с N-й позиции.

Вариант №6

Написать программу объединения двух списков в третий так, чтобы нечетные (по номеру) элементы были из первого списка, а четные - из второго.

Вариант №7

Написать программу разделения списка на два так, чтобы нечетные (по номеру) элементы были в первом списке, а четные - во втором.

Вариант №8

Написать программу разделения списка на два так, чтобы в первом списке были элементы с первого до N-го, а во втором - с N+1 до последнего.

Вариант №9

Написать программу разделения списка на два так, чтобы в первом списке были элементы с первого до заданного значения элемента, а во втором - остальные.

Вариант №10

Написать программу вставки заданного элемента в список на N-ю позицию.

Вариант №11

Написать программу замены элементов списка, номера которых заданы другим списком, на произвольную константу.

Вариант №12

Написать программу удаления из списка всех вхождений заданного элемента.

Вариант №13

Написать программу удаления элементов из списка, номера которых заданы другим списком.

Вариант №14

Написать программу объединения двух списков в третий так, чтобы одинаковые элементы из разных списков не повторялись.

Вариант №15

Написать программу инвертирования списка.

Вариант №16

Написать программу вставки в список другого списка, начиная с N-й позиции.

Вариант №17

Написать программу определения номера позиции в списке, с которого начинается заданный подсписок.

Вариант №18

Написать программу замены m элементов списка, начиная с N-й позиции, на произвольную константу.

Вариант №19

Написать программу подсчета количества вхождений элемента в список.

Вариант №20

Написать программу вычисления числа элементов списка после заданного.

Вариант №21

Написать программу вычисления числа элементов списка после N-й позиции.

Вариант №22

Написать программу замены всех четных (по номеру) элементов списка на произвольную константу.

Вариант №23

Написать программу замены всех нечетных (по номеру) элементов списка, начиная с N-й позиции, на произвольную константу.

Вариант №24

Написать программу вставки m раз в список произвольной константы, начиная с N-й позиции.

Вариант №25

Написать программу создания нового списка путем дублирования всех элементов исходного списка.

Вариант №26

Написать программу разделения исходного списка по N-му элементу и соединения полученных частей в новый список таким образом, чтобы первая часть стала второй, а вторая - первой.

Вариант №27

Написать программу разделения исходного списка на три части: с первого по N-й элемент, с N+1 по M-й и с M+1 элемента до конца списка.

Вариант №28

Написать программу удаления из исходного списка N элементов с конца списка.

Вариант №29

Написать программу замены на заданную константу N элементов исходного списка, считая с конца списка.

Вариант №30

Написать программу дописывания заданной константы N раз в конец списка.

7.2 Задания для лабораторной работы на тему: получение структурированной информации из базы данных

Текст задания

1. *Создать базу данных о заданной предметной области в виде множества фактов языка Пролог (не менее 5 фактов). Информацию о каждом компоненте БД представить в виде структуры.*
2. *Разработать набор предикатов, осуществляющих взаимодействие с БД, при помощи которых можно реализовать все типы запросов, приведенные в варианте задания. Найденные решения записать в виде фактов внутренней базы данных Пролога.*
3. *Предусмотреть проверку факта, являющегося ответом на запрос в БД. Если такой факт существует, то выдать его в качестве ответа на запрос. Если такого факта не существует в базе данных, то запустить запрос на выполнение и записать результат в БД.*

Вариант №1

Предметная область – семья. Каждая семья может быть описана структурой из трех компонентов: мужа, жены и детей. Каждый член семьи может быть описан структурой: имя, отчество, фамилия, год рождения, пол, ежемесячный доход. Для детей добавить поле «близнец».

Реализовать следующие типы запросов:

1. Проверить, существует ли в БД заданный человек (по ФИО).
2. Найти всех работающих детей.
3. Найти всех работающих мужей, чей доход больше, чем у жены.
4. Найти всех людей, которые не работают и родились до указанного года.
5. Найти число семей, у которых есть близнецы.

Вариант №2

Предметная область – семья. Каждая семья может быть описана структурой из трех компонентов: мужа, жены и детей. Каждый член семьи может быть описан структурой: имя, отчество, фамилия, год рождения, пол, ежемесячный доход. Для детей добавить поле «близнец».

Реализовать следующие типы запросов:

1. Найти всех близнецов.
2. Найти всех детей, родившихся в заданном году.
3. Найти всех работающих жен, чей доход больше заданной суммы.
4. Найти фамилии людей, у которых есть заданное число детей.
5. Найти самого старшего ребенка в БД.

Вариант №3

Предметная область – семья. Каждая семья может быть описана структурой из трех компонентов: мужа, жены и детей. Каждый член семьи может быть описан структурой: имя, отчество, фамилия, год рождения, пол, ежемесячный доход. Для детей добавить поле «близнец».

Реализовать следующие типы запросов:

1. Найти всех людей, чей доход меньше заданного.
2. Найти всех детей, младше заданного возраста.
3. Найти всех неработающих жен, которые родились позже заданного года.
4. Найти всех детей, у которых разница в возрасте родителей превышает заданную величину.
5. Подсчитать количество семей, у которых нет близнецов.

Вариант №4

Предметная область – библиотека. Каждая книга может быть описана структурой: название, автор, список изданий, число экземпляров. Автор может быть описан структурой: имя, фамилия, год рождения. Издание может быть описано структурой: издательство, номер издания, год издания, количество страниц, цена.

Реализовать следующие типы запросов:

1. Найти автора, у которого книга имеет самый ранний год издания.
2. Найти все книги, изданные более одного раза (проверка по номеру издания).
3. Найти все книги, изданные в заданном издательстве за последние десять лет.
4. Найти все книги заданного автора.
5. Найти все книги, цена которых превышает заданную сумму.

Вариант №5

Предметная область – библиотека. Каждая книга может быть описана структурой: название, автор, список изданий, число экземпляров. Автор может быть описан структурой: имя, фамилия, год рождения. Издание может быть описано структурой: издательство, номер издания, год издания, количество страниц, цена.

Реализовать следующие типы запросов:

1. Найти книгу, у которой минимальная цена.
2. Найти все книги, изданные только один раз (проверка по номеру издания).
3. Найти всех авторов, родившихся позже указанного года.
4. Найти все издательства, в которых была издана указанная книга.
5. Найти все книги, количество страниц в которых не превышает заданного значения.

Вариант №6

Предметная область – библиотека. Каждая книга может быть описана структурой: название, автор, список изданий, число экземпляров. Автор может быть описан структурой: имя, фамилия, год рождения. Издание может быть описано структурой: издательство, номер издания, год издания, количество страниц, цена.

Реализовать следующие типы запросов:

1. Найти книгу, у которой максимальное число страниц.
2. Найти все книги, изданные в заданном издательстве.

3. Найти всех авторов, книги которых имеют цену, находящуюся в заданном диапазоне.
4. Найти все книги, которые имеются в нескольких экземплярах.
5. Найти все издательства, выпускавшие книги после указанного года.

Вариант №7

Предметная область – страны мира. Каждая страна может быть описана структурой: название, площадь, географическое положение, население. Географическое положение может быть описана структурой: часть света, материк, океаны, моря, горные хребты. Население может быть описано структурой: численность, государственный язык, национальный состав. Национальный состав может быть описан структурой: национальность, численность, процент от всего населения.

Реализовать следующие типы запросов:

1. Найти страну, у которой максимальная численность населения.
2. Найти все страны, находящиеся на указанном материке с населением больше заданной величины.
3. Найти все страны, у которых однородный национальный состав (численность основной национальности более 90%).
4. Найти все страны, имеющие выход к указанному морю.
5. Найти все страны с указанным государственным языком.

Вариант №8

Предметная область – страны мира. Каждая страна может быть описана структурой: название, площадь, географическое положение, население. Географическое положение может быть описана структурой: часть света, материк, океаны, моря, горные хребты. Население может быть описано структурой: численность, государственный язык, национальный состав. Национальный состав может быть описан структурой: национальность, численность, процент от всего населения.

Реализовать следующие типы запросов:

1. Найти страну, которую омывает больше всего морей.
2. Найти все страны, на территории которых находится указанный горный хребет.
3. Найти все страны, у которых число национальностей превышает заданную величину.
4. Найти все горные хребты, находящиеся на территории указанной страны.
5. Найти все страны, у которых численность населения меньше заданной величины.

Вариант №9

Предметная область – страны мира. Каждая страна может быть описана структурой: название, площадь, географическое положение, население. Географическое положение может быть описана структурой: часть света, материк, океаны, моря, горные хребты. Население может быть описано структурой: численность, государственный язык, национальный состав. Национальный состав может быть описан структурой: национальность, численность, процент от всего населения.

Реализовать следующие типы запросов:

1. Найти страну, у которой максимальная плотность населения.
2. Найти все моря, которые омывают территорию указанной страны.
3. Найти страну, у которой численность ни одной из национальностей не превышает 50%.
4. Найти все страны, имеющие выход к указанному океану.
5. Найти все страны, у которых название части света совпадает с названием материка.

Вариант №10

Предметная область – биржа труда. Каждая вакансия может быть описана структурой: название предприятия, должность, ежемесячный доход, требования к соискателю, список соискателей. Каждый соискатель может быть описан структурой: фамилия, имя, отчество, соответствие требованиям. Требования к соискателю и соответствие требованиям могут быть описаны одной структурой: образование, возраст, пол, владение иностранными языками, умение работать на ПК, стаж работы по специальности.

Реализовать следующие типы запросов:

1. Найти должность, для которой существует максимальное число соискателей.
2. Найти все должности для мужчин, с высшим образованием и свободно владеющих иностранным языком.
3. Найти все предприятия, предлагающие доход выше указанного уровня.
4. Найти всех соискателей, умеющих работать на ПК, и имеющих стаж работы более 5 лет.
5. Найти предприятия, у которых есть заданная вакансия.

Вариант №11

Предметная область – биржа труда. Каждая вакансия может быть описана структурой: название предприятия, должность, ежемесячный до-

ход, требования к соискателю, список соискателей. Каждый соискатель может быть описан структурой: фамилия, имя, отчество, соответствие требованиям. Требования к соискателю и соответствие требованиям могут быть описаны одной структурой: образование, возраст, пол, владение иностранными языками, умение работать на ПК, стаж работы по специальности.

Реализовать следующие типы запросов:

1. Найти всех соискателей, которые соответствуют требованиям по заданной должности.
2. Подсчитать количество соискателей, имеющих высшее образование.
3. Найти все должности для соискателей с указанным стажем работы по специальности.
4. Найти все должности для мужчин с ежемесячным доходом выше указанного значения.
5. Найти все должности, для которых не требуется высшего образования.

Вариант №12

Предметная область – биржа труда. Каждая вакансия может быть описана структурой: название предприятия, должность, ежемесячный доход, требования к соискателю, список соискателей. Каждый соискатель может быть описан структурой: фамилия, имя, отчество, соответствие требованиям. Требования к соискателю и соответствие требованиям могут быть описаны одной структурой: образование, возраст, пол, владение иностранными языками, умение работать на ПК, стаж работы по специальности.

Реализовать следующие типы запросов:

1. Найти должность, у которой минимальный ежемесячный доход.
2. Найти все должности для мужчин с указанным уровнем образования, владеющих хотя бы одним иностранным языком.
3. Найти предприятия, для вакансий которых нет соискателей.
4. Найти все должности для женщин, не старше указанного возраста.
5. Найти все должности, для которых требуется знание иностранного языка.

Вариант №13

Предметная область – служба знакомств. Каждый клиент может быть описан структурой: фамилия, имя, отчество, характеристика клиента, требования к партнеру, список возможных партнеров. Характеристика

клиента и требования к партнеру могут быть описаны одной структурой: возраст, образование, национальность, ежемесячный доход, владение жилой площадью, наличие детей, отсутствие вредных привычек. Возможный партнер может быть описан следующей структурой: фамилия, имя, отчество, характеристика партнера. Характеристика партнера может быть описана структурой, одинаковой со структурой характеристики клиента.

Реализовать следующие типы запросов:

1. Подсчитать количество клиентов в БД.
2. Найти всех клиентов с указанным уровнем образования, имеющих жилую площадь, без вредных привычек.
3. Найти всех возможных партнеров с указанной национальностью.
4. Найти всех клиентов, которым необходим партнер не старше указанного возраста и не имеющий детей.
5. Найти клиента, которому требуется самый молодой партнер.

Вариант №14

Предметная область – служба знакомств. Каждый клиент может быть описан структурой: фамилия, имя, отчество, характеристика клиента, требования к партнеру, список возможных партнеров. Характеристика клиента и требования к партнеру могут быть описаны одной структурой: возраст, образование, национальность, ежемесячный доход, владение жилой площадью, наличие детей, отсутствие вредных привычек. Возможный партнер может быть описан следующей структурой: фамилия, имя, отчество, характеристика партнера. Характеристика партнера может быть описана структурой, одинаковой со структурой характеристики клиента.

Реализовать следующие типы запросов:

1. Найти всех клиентов, для которых требования к партнеру совпадают с характеристикой партнера.
2. Найти всех партнеров указанного клиента, у которых есть дети.
3. Найти всех клиентов с заданным уровнем дохода и младше указанного возраста.
4. Найти самого старого клиента службы знакомств.
5. Найти всех клиентов, у которых нет жилой площади и есть вредные привычки.

Вариант №15

Предметная область – служба знакомств. Каждый клиент может быть описан структурой: фамилия, имя, отчество, характеристика клиента, требования к партнеру, список возможных партнеров. Характеристика клиента и требования к партнеру могут быть описаны одной структурой:

возраст, образование, национальность, ежемесячный доход, владение жилой площадью, наличие детей, отсутствие вредных привычек. Возможный партнер может быть описан следующей структурой: фамилия, имя, отчество, характеристика партнера. Характеристика партнера может быть описана структурой, одинаковой со структурой характеристики клиента.

Реализовать следующие типы запросов:

1. Найти самого молодого возможного партнера в БД.
2. Найти клиента, у которого нет возможных партнеров.
3. Найти всех клиентов указанной национальности, не старше указанного возраста.
4. Найти всех партнеров указанного клиента без вредных привычек.
5. Найти всех клиентов, у которых нет детей, и которым подходит партнер, имеющий детей.

Вариант №16

Предметная область – спортивные соревнования. Каждое соревнование может быть описано структурой: ранг соревнований, вид спорта, год проведения, страна проведения, список команд - участников. Команды - участники могут быть описаны следующей структурой: название команды, страна, результаты соревнований. Результаты соревнований могут быть описаны списком структур: название команды-соперника, страна, тип результата (выигрыш, проигрыш, ничья).

Реализовать следующие типы запросов:

1. Найти ранг соревнования, в котором участвовало минимальное число команд, в заданном году и в заданном виде спорта.
2. Найти все команды указанного ранга соревнований и года проведения, у которых не было ни одного проигрыша.
3. Найти всех соперников указанной команды в соревнованиях заданного ранга в заданном году.
4. Найти вид спорта, в котором проводились соревнования в заданном году.
5. Найти все команды указанной страны, участвовавшие в соревнованиях заданного ранга.

Вариант №17

Предметная область – спортивные соревнования. Каждое соревнование может быть описано структурой: ранг соревнований, вид спорта, год проведения, страна проведения, список команд-участников. Команды-участники могут быть описаны следующей структурой: название команды, страна, результаты соревнований. Результаты соревнований могут

быть описаны списком структур: название команды-соперника, страна, тип результата (выигрыш, проигрыш, ничья).

Реализовать следующие типы запросов:

1. Найти вид спорта, в котором участвовало максимальное число команд, в заданном году и в заданном ранге соревнований.
2. Найти все страны, где проводились чемпионаты мира по указанному виду спорта.
3. Найти всех соперников указанной команды в соревнованиях определенного ранга в заданном году.
4. Найти все соревнования, проводившиеся в заданном году.
5. Найти все команды, у которых были ничьи.

Вариант №18

Предметная область – спортивные соревнования. Каждое соревнование может быть описано структурой: ранг соревнований, вид спорта, год проведения, страна проведения, список команд-участников. Команды-участники могут быть описаны следующей структурой: название команды, страна, результаты соревнований. Результаты соревнований могут быть описаны списком структур: название команды-соперника, страна, тип результата (выигрыш, проигрыш, ничья).

Реализовать следующие типы запросов:

1. Найти год, в который участвовало максимальное число команд, в заданном виде спорта и в заданном ранге соревнований.
2. Найти вид спорта, в котором выступает заданная команда.
3. Найти все команды, которые участвовали в Олимпийских играх по определенному виду спорта.
4. Найти все команды, участвовавшие в соревнованиях в заданном году.
5. Найти все команды определенной страны, у которых были выигрыши.

Вариант №19

Предметная область – видеотека. Каждая видеокассета может быть описана структурой: название фильма, год создания, киностудия, атрибуты фильма. Атрибуты фильма могут быть описаны структурой: автор сценария, режиссер, список фамилий исполнителей главных ролей, премии. Премии могут быть описаны списком из следующих структур: название фестиваля, год проведения.

Реализовать следующие типы запросов:

1. Найти фильм, получивший больше всего премий.

2. Подсчитать число фильмов указанного режиссера.
3. Найти всех режиссеров, фильмы которых создавались на заданной киностудии.
4. Найти все фильмы, определенного актера за указанный период времени.
5. Найти всех сценаристов, в фильмах которых снимался определенный актер.

Вариант №20

Предметная область – видеотека. Каждая видеокассета может быть описана структурой: название фильма, год создания, киностудия, атрибуты фильма. Атрибуты фильма могут быть описаны структурой: автор сценария, режиссер, список фамилий исполнителей главных ролей, премии. Премии могут быть описаны списком из следующих структур: название фестиваля, год проведения.

Реализовать следующие типы запросов:

1. Найти сценариста, в фильме которого снялось минимальное число актеров.
2. Найти режиссеров и сценаристов, у которых есть фильмы, получившие премии.
3. Найти все фильмы указанного режиссера, снятого на определенной киностудии.
4. Найти всех исполнителей главных ролей указанного фильма.
5. Найти все киностудии, которые работали с указанным режиссером.

Вариант №21

Предметная область – видеотека. Каждая видеокассета может быть описана структурой: название фильма, год создания, киностудия, атрибуты фильма. Атрибуты фильма могут быть описаны структурой: автор сценария, режиссер, список фамилий исполнителей главных ролей, премии. Премии могут быть описаны списком из следующих структур: название фестиваля, год проведения.

Реализовать следующие типы запросов:

1. Найти режиссера, у которого фильм получил максимальное число премий.
2. Найти все фильмы, получившие премии, в которых снимался указанный актер.
3. Найти все фильмы определенного сценариста, снятые в указанном году.

4. Найти количество актеров, снимавшихся на определенной киностудии.
5. Найти всех актеров, снимавшихся в фильмах определенного сценариста.

Вариант №22

Предметная область – учебная группа факультета. Каждая учебная группа может быть описана структурой: название факультета, код специальности, номер группы, состав группы. Состав группы может быть описан списком структур, описывающих отдельного студента: фамилия, имя, отчество, обучение на военной кафедре, сводная ведомость. Сводная ведомость может быть описана списком из следующих структур: предмет, оценка.

Реализовать следующие типы запросов:

1. Подсчитать число групп определенной специальности.
2. Найти оценку определенного студента по заданному предмету.
3. Найти группу, которая сдала больше всего предметов в сессию.
4. Найти всех студентов, имеющих задолженности.
5. Подсчитать число студентов, обучающихся на военной кафедре.

Вариант №23

Предметная область – учебная группа факультета. Каждая учебная группа может быть описана структурой: название факультета, код специальности, номер группы, состав группы. Состав группы может быть описан списком структур, описывающих отдельного студента: фамилия, имя, отчество, обучение на военной кафедре, сводная ведомость. Сводная ведомость может быть описана списком из следующих структур: предмет, оценка.

Реализовать следующие типы запросов:

1. Подсчитать общее число студентов на указанном факультете.
2. Найти группу, у которой больше всего отличников.
3. Найти все предметы, которые изучала определенная группа.
4. Найти всех студентов, не обучающихся на военной кафедре.
5. Найти все группы, изучавшие определенный предмет.

Вариант №24

Предметная область – учебная группа факультета. Каждая учебная группа может быть описана структурой: название факультета, код специальности, номер группы, состав группы. Состав группы может быть описан списком структур, описывающих отдельного студента: фамилия, имя, отчество, обучение на военной кафедре, сводная ведомость. Сводная ве-

домость может быть описана списком из следующих структур: предмет, оценка.

Реализовать следующие типы запросов:

1. Подсчитать средний балл сессии по указанной группе.
2. Найти группу, в которой минимальное число студентов.
3. Найти все предметы в указанной группе, по которым сдавался экзамен.
4. Найти код специальности указанной группы.
5. Найти всех студентов, обучающихся на военной кафедре.

Вариант №25

Предметная область – база данных продажи автомобилей. Каждый автомобиль может быть описан структурой: марка автомобиля, страна фирмы-изготовителя, список фирм-продавцов. Фирма-продавец может быть описана структурой: название фирмы, страна, список имеющихся моделей. Модель может быть описана структурой: наименование модели, цена, список имеющихся расцветок.

Реализовать следующие типы запросов:

1. Найти марку автомобиля, которую продает больше всего фирм.
2. Подсчитать число стран, в которых продаются автомобили заданной марки.
3. Найти все фирмы, продающие автомобили заданной марки.
4. Найти все модели автомобилей, цена которых ниже заданной.
5. Найти все фирмы, которые продают автомобили заданной модели.

Вариант №26

Предметная область – база данных продажи автомобилей. Каждый автомобиль может быть описан структурой: марка автомобиля, страна фирмы-изготовителя, список фирм-продавцов. Фирма-продавец может быть описана структурой: название фирмы, страна, список имеющихся моделей. Модель может быть описана структурой: наименование модели, цена, список имеющихся расцветок.

Реализовать следующие типы запросов:

1. Найти марку и модель автомобиля, у которой минимальная цена
2. Подсчитать число расцветок автомобиля заданной модели у определенного продавца
3. Найти все страны-изготовители, выпускающие автомобили заданной марки
4. Найти все марки автомобилей, продающиеся в заданной стране

5. Найти все фирмы, которые продают автомобили заданной расцветки.

Вариант №27

Предметная область – база данных продажи автомобилей. Каждый автомобиль может быть описана структурой: марка автомобиля, страна фирмы-изготовителя, список фирм-продавцов. Фирма-продавец может быть описана структурой: название фирмы, страна, список имеющихся моделей. Модель может быть описана структурой: наименование модели, цена, список имеющихся расцветок.

Реализовать следующие типы запросов:

1. Найти марку и модель автомобиля, у которой максимальное число расцветок.
2. Подсчитать число фирм, в которых продаются автомобили заданной марки.
3. Найти все фирмы, продающие автомобили в заданной стране.
4. Найти все марки автомобилей, выпускающиеся в заданной стране.
5. Найти все фирмы, которые продают автомобили заданной цены.

Вариант №28

Предметная область – расписание движения самолетов. Каждый рейс может быть описан структурой: название авиакомпании, номер рейса, пункт отлета, пункт прилета, время отлета, время прилета, список пунктов промежуточных посадок, список тарифов. Тариф может быть описан структурой: тип класса салона, цена.

Реализовать следующие типы запросов:

1. Найти авиакомпанию, у которой максимальная стоимость билета по заданному маршруту.
2. Подсчитать число авиакомпаний, самолеты которых летают в заданный город (включая промежуточные посадки).
3. Найти все номера рейсов, улетающих до указанного времени.
4. Найти все авиакомпании, самолеты которых летают из указанного города.
5. Найти все рейсы, на которые есть билеты экономкласса.

Вариант №29

Предметная область – расписание движения самолетов. Каждый рейс может быть описан структурой: название авиакомпании, номер рейса, пункт отлета, пункт прилета, время отлета, время прилета, список пунктов промежуточных посадок, список тарифов. Тариф может быть описан структурой: тип класса салона, цена.

Реализовать следующие типы запросов:

1. Найти авиакомпанию, у которой минимальная продолжительность полета по заданному маршруту.
2. Подсчитать число рейсов, совершающих промежуточную посадку в заданном пункте.
3. Найти все авиакомпании, у которых есть билеты бизнес-класса.
4. Найти все рейсы, прилетающие в указанный пункт до заданного времени.
5. Найти все рейсы, летающие по заданному маршруту.

Вариант №30

Предметная область – расписание движения самолетов. Каждый рейс может быть описан структурой: название авиакомпании, номер рейса, пункт отлета, пункт прилета, время отлета, время прилета, список пунктов промежуточных посадок, список тарифов. Тариф может быть описан структурой: тип класса салона, цена.

Реализовать следующие типы запросов:

1. Найти авиакомпанию, у которой максимальное число тарифов на заданный маршрут.
2. Подсчитать число рейсов, улетающих из заданного города до указанного времени.
3. Найти все авиакомпании, у которых есть билеты класса люкс.
4. Найти все рейсы, у которых цена билета ниже заданной.
5. Найти все авиакомпании, у которых время полета меньше заданного.

7.3 Задания для лабораторной работы на тему: решение логических головоломок

Вариант №1

Три друга заняли первое, второе и третье места на школьной олимпиаде. Друзья учатся в разных классах, зовут их по-разному и они участвовали в олимпиадах по разным предметам. Алексей участвовал в олимпиаде по химии и выступил лучше, чем девятиклассник. Семиклассник Сергей выступил лучше участника олимпиады по физике. Участник олимпиады по математике занял первое место. Кто из них в каком классе учится и кто в какой олимпиаде участвовал?

Вариант №2

Трое хищников – волк, медведь и лиса пошли на охоту. У них разный возраст и они охотились на разную дичь. Медведь любит рыбу и

поймал дичи больше, чем тот, кому 2 года. Трехлетний волк поймал больше, чем любитель зайцев. Тот, кто любит кур, поймал больше всех. Одному из них 4 года. Определите, у кого какой возраст и кто какую дичь предпочитает.

Вариант №3

Три друга – Иван, Дмитрий и Степан преподают биологию, физику и химию в школах Москвы, Санкт-Петербурга и Киева. Иван – не в Москве, Дмитрий – не в Санкт-Петербурге. Москвич не преподаёт физику. Тот, кто живёт в Санкт-Петербурге, преподаёт химию. Дмитрий не преподаёт биологию. Кто, в каком городе живёт и что преподаёт?

Вариант №4

Три ученика – Коля, Миша и Андрей сидят в классе за партами первого ряда. У них разный цвет волос и они любят разные предметы. Миша любит физику и сидит ближе к классной доске, чем рыжий. Блондин Коля сидит ближе к классной доске, чем любитель литературы. Тот, кто любит математику, сидит за первой партой. Один из них брюнет. Определите у кого какой цвет волос и кто какой предмет предпочитает.

Вариант №5

Три друга заняли первое, второе и третье места в соревнованиях универсиады. Друзья разной национальности, зовут их по-разному и любят они разные виды спорта. Майкл предпочитает баскетбол и играет лучше, чем американец. Израильтянин Саймон играет лучше теннисиста. Игрок в крикет занял первое место. Кто из них австралиец? Каким видом спорта увлекается Ричард?

Вариант №6

Кондратьев, Давыдов и Фёдоров живут на одной улице. Один из них - столяр, другой – маляр, третий – водопроводчик. У столяра самый большой дом из троих и у него нет автомобиля. Федорову нравится машина Кондратьева и его дом меньше, чем у маляра. Определите, кто чем занимается, у кого есть машина.

Вариант №7

Браун, Гриффит, Клеменс и Грин - четверо студентов университетов разных стран приехали на международный фестиваль молодёжи и студентов. Один из них – канадец, второй – американец, третий – англичанин, четвёртый – австралиец. Браун и Клеменс были на концерте, в котором принимал участие их знакомый англичанин. Гриффит и австралиец знакомы, так как пели дуэтом под аккомпанемент их знакомого американца. Австралиец пригласил к себе в гости своего знакомого Грина и собирается

пригласить своего знакомого Брауна. Определите национальности студентов.

Вариант №8

В одном театре работают четыре актёра: Смирнов, Снегов, Морев и Никитин. Один из них играет роль Отелло, другой – короля Лира, третий – Ромео, четвёртый – Гамлета. Смирнов – не Отелло и не Гамлет. Морев – не Ромео и не Отелло. Никитин – не Гамлет, не Отелло. Снегов не играет ни Гамлета, ни Ромео. Если Морев играет Гамлета, то Смирнов не играет короля Лира. Кто из актёров кого играет?

Вариант №9

Пол, Джон и Джордж – три музыканта. Один из них – гитарист, другой – ударник, третий – пианист. Каждый из них выступает в составе одной из рок-групп - «Ромашка», «Василёк» или «Колокольчик». Тот, кто выступает в составе «Ромашки», не гитарист. Джон выступает в составе не «Василька». Пол – выступает в составе не «Ромашки». Ударник выступает в составе рок-группы «Василёк». Джон не пианист. На каком инструменте играет каждый из трёх музыкантов и в составе какой рок-группы выступает?

Вариант №10

Лена, Коля, Марина и Саша работают на одном предприятии конструктором, технологом, экономистом и дизайнером. Лена и Марина учились вместе с экономистом. Коля и дизайнер в обеденный перерыв часто играют в теннис с технологом. Дизайнер постоянно сталкивается по работе с Леной и иногда с Сашей. Кто кем работает?

Вариант №11

Трое мальчиков Костя, Фома и Марат дружили с тремя девочками – Женей, Светой и Мариной. Но вскоре компания разделилась на пары, потому что оказалось, что Света ненавидит ходить на лыжах. Костя, Женин брат, часто катается со своей подружкой на лыжах. А Фома бежит на свидание к Костиной сестре. С кем же проводит время Марат?

Вариант №12

В комнате женского общежития живут четыре девушки: Мира, Мона, Мод и Мэри. Как-то раз они собрались все вместе, причём каждая занималась своим делом. Одна делала маникюр, другая – расчёсывала волосы, третья – прихорашивалась перед зеркалом, а четвёртая – читала. Точно о них известно следующее: Мира не делала маникюр и не читала. Мод не прихорашивалась перед зеркалом и не делала маникюр. Мэри не читала и не занималась маникюром. Мона не читала и не прихорашивалась перед

зеркалом. Если Мод читала, то Мэри прихорашивалась перед зеркалом. Кто из девушек делала маникюр, кто расчёсывала волосы, кто прихорашивалась перед зеркалом, а кто читала?

Вариант №13

«Динамо», «Торпедо» и «Спартак» - команды-участники розыгрыша кубка по футболу из Челябинска, Краснодара и Самары. Команда из Челябинска забила меньше всех мячей. «Динамо» забила мячей больше, чем команда из Краснодара, а «Торпедо» меньше «Спартака». За какие города выступают названные команды?

Вариант №14

В одном купе поезда ехали три пассажира – мистер Джонс, мистер Смит и мистер Робинсон. Один из них житель Чикаго, другой живёт в Детройте, третий – в Окленде. За разговорами выяснилось, что мистер Джонс из Чикаго старше, чем тот, у кого нет детей. Житель Детройта самый старый. Мистер Смит, у которого двое детей, старше, чем житель Окленда. У одного из них только один ребенок. Определите, кто из какого города и у кого сколько детей.

Вариант №15

Билл, Джон и Ричард играют в одном оркестре. Они владеют разными музыкальными инструментами и выступают в костюмах разных цветов. Джон играет на саксофоне и находится ближе к дирижёру, чем тот, кто выступает в белом костюме. Билл на концерт одевает чёрный костюм и сидит ближе к дирижёру, чем флейтист. Альтист сидит к дирижёру ближе всех. Один из друзей приходит на концерт в костюме синего цвета. Определите, кто какими инструментами владеет, и в каком костюме выступает.

Вариант №16

Петров, Семёнов и Арбузов – пилот, штурман и бортинженер. Один из них летает на самолёте ТУ-134, другой на АН-24, третий на ИЛ-76. Тот, кто летает на ТУ-134, не штурман. Семёнов работает не на АН-24. Петров – не на ТУ-134. Бортинженер летает на АН-24. Семёнов не пилот. У кого какая профессия и какой самолёт?

Вариант №17

Андреев, Борисов и Николаев – поэт, композитор и художник. У поэта нет бороды и он зарабатывает меньше всех. Андреев считает, что Николаеву очень идет борода. Андреев зарабатывает больше композитора. Определите, кто является поэтом, кто композитором, кто художником и у кого есть борода.

Вариант №18

За столиком в кафе познакомились три девушки: Алёна, Светлана и Марианна. Одна из них работает парикмахером в Салоне красоты, другая – модельером в Доме мод, третья – медсестрой в санатории. Также выяснилось, что парикмахер – самая молодая из троих, Марианна старше модельера, а Светлана младше Алёны. Определите профессии девушек.

Вариант №19

В одном институте учатся три друга – Липатов, Кондратьев и Зайцев. Один из них – будущий физик, второй – математик, третий – программист. У физика нет ни братьев, ни сестёр. Он самый младший из друзей. Зайцеву нравится сестра Липатова и он старше математика. Определите будущие профессии друзей.

Вариант №20

У Васи, Вани, Славы и Вовы живут дома собака, кошка, морская свинка и попугай. Вася и Слава знакомы с хозяином кошки. Ваня и хозяин собаки часто ходят в кино с хозяином попугая. Хозяин собаки постоянно встречается в школе с Васей и иногда с Вовой. У кого какое животное?

Вариант №21

Трое коллег – Костя, Митя и Артем пошли в ресторан. У них разное семейное положение и они любят разные блюда. Митя любит рыбу и получает больше, чем тот, кто холостой. Разведенный Костя получает больше, чем любитель пельменей. Тот, кто любит плов, получает больше всех. Один из них женат. Определите, у кого какое семейное положение, и кто какое блюдо предпочитает.

Вариант №22

В книжный магазин пришли четыре подруги: Соколова, Ястребова, Орлова и Голубева. Одна из них искала книгу «Поющие в терновнике», другая – «Финансист», третья – «Охоту на овец», четвёртая – «Заводной апельсин». Соколова не искала книги «Поющие в терновнике» и «Заводной апельсин». Орлова – не искала книги «Охота на овец» и «Поющие в терновнике». Голубева не искала книги «Заводной апельсин» и «Поющие в терновнике». Если Ястребова искала книгу «Поющие в терновнике», то Голубева искала книгу «Финансист». Ястребова не искала книги «Заводной апельсин» и «Охота на овец». Какая девушка какую книгу искала?

Вариант №23

Боб, Джек и Рональд часто ходят в кино. Они любят разные жанры и предпочитают разные кинотеатры. Джек любит комедии и садится ближе, чем тот, кто ходит в кинотеатр «Звезда». Боб ходит в кинотеатр «Самара»

и сидит ближе, чем любитель триллеров. Тот, кто любит боевики, сидит к экрану ближе всех. Один из друзей ходит в кинотеатр «Огонек». Определите, кто какой жанр любит и в какой кинотеатр предпочитает ходить.

Вариант №24

Кукушкина, Петухова и Галкина – модельер, менеджер и страховой агент. Одна из них водит «форд», другая «рено», третья «ауди». Тот, кто ездит на «форде», не «менеджер». Петухова ездит не на «рено». Кукушкина – не на «форде». Страховой агент ездит на «рено». Петухова не модельер. У кого какая профессия и какой автомобиль?

Вариант №25

Три семейные пары– Ивановы, Петровы и Сидоровы купили путевки за 2000$, 3000$ и 5000$ в Турцию, Италию и Испанию. Ивановы поехали не в Турцию, Петровы – не в Италию. Те, кто поехали в Турцию, не платили за путевку 5000$. Те, кто поехали в Италию, заплатили 3000$. Петровы не платили за путевку 2000$. Кто в какой стране отдыхал и кто сколько заплатил за путевку?

Вариант №26

«Тройка» думает, что «туз» не в своём уме. «четвёрка» думает, что «тройка» и «двойка» обе не могут быть не в своём уме. «пятёрка» думает, что «туз» и «четвёрка» либо оба не в своём уме, либо оба в своём уме. «шестёрка» думает, что «туз» и «двойка» оба в своём уме. «семёрка» думает, что «пятёрка» не в своём уме. «валет» думает, что «шестёрка» и «семёрка» обе не могут быть не в своём уме. В своём ли уме «валет»?

Вариант №27

Король думает, что королева думает, что она не в своём уме. В своём ли уме король?

Вариант №28

Антон, Максим и Дима учатся в художественной школе. Каждый ученик художественной школы рисует или портреты, или пейзажи, или натюрморты и у каждого ученика один любимый художник - И.Репин, И.Шишкин или А.Рублёв. Антон рисует портреты и учится лучше, чем тот, кто любит И.Репина. Максим любит И.Шишкина и учится лучше, чем тот, кто рисует пейзажи. Тот, кто рисует натюрморты, учится лучше всех. Один из учеников любит А.Рублёва. Определите, кто из учеников художественной школы что рисует и у кого какой любимый художник.

Вариант №29

На экскурсию в Эрмитаж приехали трое туристов – Кузнецов, Смирнов и Попов. Один из них из Казани, другой – из Самары, третий – из Мо-

сквы. Выяснилось, что житель Казани купил билет самым первым из троих. Кузнецов из Самары купил билет раньше, чем тот, кто живет в гостинице №1. Смирнов, живущий в гостинице №2, купил билет раньше, чем москвич. Один из них живет в гостинице №3. Определите, кто из какого города и кто в какой гостинице живет.

Вариант №30

Тони, Майкл и Джон – спортсмены. Один из них боксер, второй футболист, третий – пловец. Один из них любит снег, второй любит дождь, третий любит солнечную погоду. Тот, кто любит солнечную погоду, не боксер. Тони не любит снег. Майкл – не любит солнечную погоду. Футболист любит снег. Тони не пловец. У кого какая любимая погода и кто каким спортом занимается?

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. Адаменко, А.Н. Логическое программирование и Visual Prolog / А.Н. Адаменко, А. Кучуков. – Спб.: БХВ – Петербург, 2003.

2. Братко, И. Алгоритмы искусственного интеллекта на языке Prolog / И. Братко. М.: Вильямс, 2004.

3. Стерлинг, Л. Искусство программирования на языке Пролог / Л. Стерлинг, Э. Шапиро; пер. с англ. – М.: Мир, 1990.

Учебное издание

Солдатова Ольга Петровна
Лёзина Ирина Викторовна

Программирование на языке ПРОЛОГ

Методические указания к лабораторным работам

Редактор Н. С. К у п р и я н о в а
Компьютерная верстка Т. Е. П о л о в н е в а

Подписано в печать 23.12.2008. Формат 60x84 1/16.
Бумага офсетная. Печать офсетная.
Печ. л. 3.25.
Тираж 50 экз. Заказ .

Самарский государственный
аэрокосмический университет.
443086, Самара, Московское шоссе, 34.

Изд-во Самарского государственного
аэрокосмического университета.
443086 Самара, Московское шоссе, 34.