

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
АЛТАЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет математики и информационных технологий
Кафедра информатики

ФУНКЦИОНАЛЬНОЕ И ЛОГИЧЕСКОЕ ПРОГРАММИРОВАНИЕ

Учебно-методическое пособие



Барнаул

Издательство
Алтайского государственного
университета
2016

Составитель:

канд. физ.-мат. наук, доцент ***О.Н. Половикова***

Рецензенты:

канд. физ.-мат. наук, доцент ***М.А. Рязанов***

канд. физ.-мат. наук, доцент ***А.А. Шайдунов***

Курс «Функциональные и логические языки» читается для студентов на факультете математики и информационных технологий Алтайского госуниверситета. Одна из основных задач курса – сформировать у студентов необходимую теоретическую базу и практические навыки для создания программных продуктов с использованием одного или нескольких языков декларативного программирования. В качестве практической базы используется среда Visual Prolog и интерпретатор Common Lisp.

План УМД 2016 г., п. 45

Подписано в печать 28.06.2016. Формат 60x84/16

Усл.-печ. л. 2. Тираж 50 экз. Заказ №129

Типография Алтайского государственного университета:

656049, Барнаул, ул. Димитрова, 66

СОДЕРЖАНИЕ

ЛЕКЦИЯ 1. БАЗА ПОНЯТИЙ ДЕКЛАРАТИВНОГО ЯЗЫКА	4
ЛЕКЦИЯ 2. ПРОГРАММА НА ЯЗЫКЕ VISUAL PROLOG	10
ЛЕКЦИЯ 3. ФУНКЦИОНАЛЬНЫЙ ЯЗЫК LISP. СПИСКИ.....	15
ЛЕКЦИЯ 4. ФУНКЦИИ ОБРАБОТКИ СПИСКОВ	19
ЗАДАНИЯ ДЛЯ ПРАКТИЧЕСКИХ ЗАНЯТИЙ.....	22
Лабораторная работа № 1	22
Лабораторная работа № 2	23
Лабораторная работа № 3	23
Лабораторная работа № 4	24
Лабораторная работа № 5	25
Лабораторная работа № 6	26
ИНДИВИДУАЛЬНЫЕ ЗАДАНИЯ ПО ВАРИАНТАМ	27
1. Поиск решения с возвратом	27
2. Работа со списками	35
3. Структуры данных	37
4. Списки, функции, управляющие структуры	42
БИБЛИОГРАФИЧЕСКИЙ СПИСОК	44

Лекция 1. База понятий декларативного языка

Функция

Можно интуитивно дать определение понятию – **функция**. Можно сказать, что **функция** – это отображение из множества всевозможных значений аргумента во множество результатов вычислений. Функция имеет **функтор**, то есть имя, и **аргументы**. Примеры функций: $\sin(x)$; логическая функция $X \rightarrow Y$, которая возвращает истину или ложь.

Чтобы применить функцию, следует подставить в переменные, или аргументы, конкретные значения. Для функции можно определить **домен** (т.е. область её определения) – это множество всевозможных значений аргумента. **Образ** – это множество результатов вычислений. Для функции *синус* доменом является *множество действительных чисел*.

Предикаты

Предикатами являются функции, домены которых отображаются в множество $\{true, false\}$. (Образ состоит из двух значений). Предикатами являются следующие функции: $X > Y$, $X \rightarrow Y$, $X \downarrow Y$, $X' = Y$.

Предикат с одним аргументом задаёт или определяет свойство этого аргумента. Предикат с двумя и более аргументами определяет отношение, которое может быть выражено значениями: *true* или *false* в зависимости от характеристик аргументов.

Так как множество базовых предикатов не могут в полной мере описать и задать всю специфику работы системы для конкретной предметной области, то требуется использовать язык программирования с предикатами, который позволяет определять и использовать новые отношения.

Например, следующие предикаты востребованы в разных предметных областях:

Сегодня_выгодно_покупать_Евро (курс евро, стоимость нефти, динамика курса,)

Сегодня_будет_дождь (Влажность, температура воздуха и т.д.)

Декларативные языки

Для построения различных предикатов, а так же для решения задач на их основе созданы декларативные языки. Т.е. языки, с помощью которых можно задекларировать (описать) различные отношения между объектами, а также построить поиск решения. Декларативные языки программирования развиваются по двум веткам: логические языки и функциональные языки. К логическим языкам относится язык Prolog и все его диалекты, а к функциональным – язык Lisp.

Традиционно под программой понимают последовательность операторов (команд, выполняемых компьютером). Этот стиль программирования принято называть императивным. Программируя в императивном стиле, программист должен объяснить компьютеру, как нужно решать задачу. Другой стиль программирования – так называемый декларативный стиль, в котором программа состоит из совокупности утверждений, описывающих фрагмент предметной области или сложившуюся ситуацию. Программируя в декларативном стиле, программист должен описать, что нужно решать.

История

Язык программирования Visual Prolog – декларативный язык программирования общего назначения. Это усовершенствование языка Prolog (от “PROgramming in LOGic”).

Теоретические основы языка Prolog заложил в 1972 г. Роберт Ковальский (г. Эдинбург). В 1977 г. Уоррен и Перейра (г. Эдинбург) создали эффективный компилятор языка Prolog для ЭВМ, который послужил прототипом для многих последующих реализаций этого языка. Следует заметить, что компилятор был написан на самом языке Prolog. Эта реализация языка Prolog получила название – *эдинбургская* версия. Алгоритм, использованный при его реализации, послужил прототипом для многих последующих диалектов. Как правило, если какая-либо Prolog-система и не поддерживает *эдинбургский* Prolog, то в ее состав входит подсистема, переводящая программу в *эдинбургский* диалект.

В 1981 г. в Японии стартовал проект *Института по разработке методов создания компьютеров нового поколения*. Авторы проекта надеялись, что у них получится не подстраивать мышление человека под принципы функционирования компьютеров, а приблизить работу компьютера к тем моделям, по которым работает мышление человека. Одним из направлений данного проекта было развитие логического языка Prolog.

Сегодня существует много реализаций языка Prolog. Наиболее известные из них следующие: VisualProlog, BinProlog, AMZI-Prolog, CProlog, MicroProlog, StrawberryProlog, SWIProlog, UNSWProlog, Акторный Пролог и т.д. Разработчиком Visual Prolog является Prolog Development Center (Дания), а руководит командой разработчиков – Томас Линдер Пулс.

В нашей стране были разработаны такие версии языка Prolog, как Пролог-Д, Акторный Пролог, а также Флэнг. Следует заметить, что в последнее время растёт интерес к языку Акторный Пролог, это связано с развитием идеи Semantic Web. В рамках реализации проекта

Semantic Web востребованы технологии и средства для построения программных агентов – специализированных программных модулей для выполнения сервисов в рамках глобальной сети Internet. Программные агенты, решая свои конкретные задачи, при этом могут договариваться между собой без участия человека-пользователя, используя свои модели знаний. Пользователь может получить любую комбинированную услугу средствами взаимодействия агентов между собой. Возможности языка Акторного Пролога позволяют разрабатывать и программировать подобных агентов.

Стандарт языка был принят в 1995 г. Prolog – один из старейших языков логического программирования. Сегодня он используется в системах обработки естественных языков, исследованиях искусственного интеллекта, экспертных системах, онтологиях и других предметных областях, для которых естественно применение логической парадигмы.

Prolog был создан под влиянием более раннего языка Planner и позаимствовал из него идею обратного логического вывода (последовательный вызов процедур, исходя из целей). Prolog реализован практически для всех известных операционных систем и платформ: Unix, Windows, OS и для мобильных платформ.

Предложения Хорна

Программа на логическом языке (называется *базой знаний*) состоит из **предложений Хорна** (или **утверждений**), каждое предложение **Хорна** заканчивается точкой. Предложения Хорна в программе на языке VisualProlog ещё разделены по разделам (см. следующую лекцию).

Предложение – это формула вида:

Голова :- *Тело*. ($A :- B_1, \dots, B_n$.)

A называется **заголовком** или **головой** предложения, а B_1, \dots, B_n – **телом**. Каждое B_i является предложением, т.е. функцией, у которой *домен* принимает любые значения, а *образом* является множество $\{true, false\}$. Читается как: «чтобы доказать *голову*, следует доказать *тело*». *Тело* предложения состоит из нескольких предикатов (целей), скомбинированных с помощью конъюнкции и/или дизъюнкции. Как правило, конъюнкция предикатов оформляется символом “запятая” (B_k, B_{k+1}), дизъюнкция – символом “точка с запятой” ($B_k ; B_{k+1}$).

Предложения с пустым телом называются *фактами* и эквивалентны предложениям вида:

Голова :- *true*.

Можно считать, что *факты* – это предложение, у которого *тело* пустое.

Пример базы знаний

Задача 1. Определить и показать стипендию для каждого студента заданной группы. Стипендия начисляется в зависимости от балла студента, по следующему принципу:

- ниже 50 баллов – стипендии нет;
- до 60 баллов начисляется базовая стипендия – 100 ед.
- выше 60 баллов – к базовой стипендии добавляется ещё 10%.

Для решения задачи построим следующие предикаты:

1) предложение (факт):

студент (имя, группа, балл)

Область определения данного отношения можно задать декартовым произведением:

$string \times string \times integer$.

2) Предложение:

стипендия (балл, размер стипендии)

Область определения данного отношения можно задать декартовым произведением:

$integer \times real$

Область значений (образ) данных предложений включает два значения $\{true, false\}$. Базу знаний можно задать следующим набором фактов и предикатов:

студент("Миша", "443", 50).

студент("Лена", "441", 49).

студент("Коля", "441", 57).

студент("Оля", "442", 39).

студент("Галя", "441", 79).

студент("Петя", "442", 76).

стипендия (N, 0) :- N < 50.

стипендия (N, S) :- N > 70, S = 100 * 1.1.

стипендия (N, 100) :- not(N < 50), not(N > 70).

Поиск решения

Целью выполнения программы на Prolog является оценивание целевого предиката, который записан в разделе *goal*. Для Visual Prolog такой целью выступает предложение *run*.

Имея этот предикат и набор правил и фактов, заданных в программе, Prolog-система пытается найти привязки (значения) переменных, при которых целевой предикат принимает значение истина. В случае успешности поиска Prolog-система выдает ответ *Yes* (Успех). В

процессе поиска подходящего согласования всех предикатов целевого предложения выполняется последовательный обход всего дерева решений. Если на каком-то этапе доказательства целевого предиката невозможно доказать составляющие его предложения, то происходит откат на один шаг назад, повторно доказывается предшествующее предложение, но уже с другими значениями переменных, на следующих фактах и правилах.

Если Prolog-система просмотрела все возможные варианты согласования предикатов целевого предложения и не нашла привязку переменных, при которой целевой предикат принимает значение *истина*, в этом случае программой в качестве ответа будет выдано значение *No*. Такой исход следует интерпретировать, как **невозможность** согласовать предикаты целевого предложения.

Рассмотрим *поиск решения* Prolog-системой целевого предиката заданного по базе знаний к задаче 1 (см. выше).

Задача 2. Найти студента со стипендией выше базовой.

Сформируем цель по созданным ранее предложениям, а также используя встроенные предикаты для ввода-вывода.

```
goal:- студент(A, N, B),  
вывод_данных("\n Имя:", A, " гр:" , N, " балл:" , B),  
стипендия(B, S), S>100,  
вывод_данных("\n\t Имя:", A, " гр:",N, " балл:", B, " ", S).
```

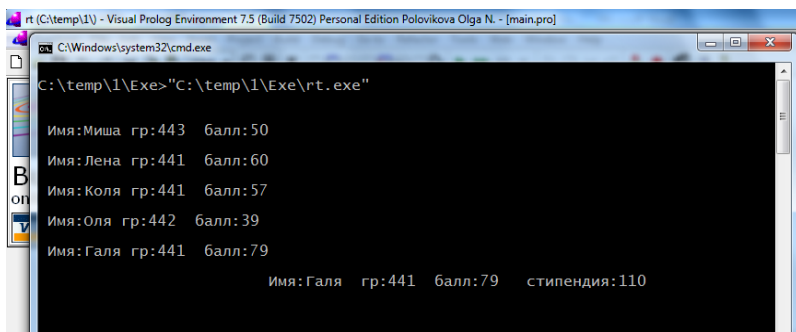
В процессе доказательства данного целевого предиката будут последовательно согласовываться предикаты: *студент*, *стипендия*, *S>100* (предикаты для ввода и вывода всегда согласуются со значением *истина*) и связываться соответствующие переменные: *A*, *N*, *B*, *S*. В результате при выборе первого факта *студент("Миша", "443", 50)* базы знания, переменные *A*, *N*, *B* связываются с соответствующими значениями: *"Миша"*, *"443"*, *50*.

Далее происходит согласование предложения стипендия и связывание несвязанной ранее переменной *S* (со значением *100*). После этого согласуется предикат *S>100*, результатом операции будет значение – *ложь*. Поэтому происходит откат на предыдущий шаг к повторному согласованию предиката *стипендия*, переменная *S* при этом теряет своё значение и становится несогласованной. Так как невозможно другое связывание переменной *S* (при фиксированном значении переменной *B*) в предложении *стипендия*, то происходит откат ещё на один шаг назад – к согласованию факта *студент*, переменные *A*, *N*, *B* теряют свои значения.

При повторном доказательстве факта *студент* переменные *A*, *N*, *B* могут согласовываться только с набором ранее неиспользуемых

данных, поэтому будет выбран следующий факт *студент("Лена", "441", 49)*, а переменные связаны с соответствующими значениями из данного факта.

По такому сценарию происходит согласование предикатов *стипендия*, $S > 100$ и связывание переменной S . Результатов согласования предиката $S > 100$ будет значение – *ложь*. Затем выполняются *откаты*, Prolog-системой будет рассматриваться факт *студент("Коля", "441", 57)*. И так далее, пока система не откатится до факта *студент("Гая", "441", 79)*, предикатов *стипендия*, $S > 100$ *успешно согласуются*, переменная S связывается со значением *110*. Программа связала со значениями все переменные, согласовала все предикаты целевого утверждения, поэтому процесс поиска решение успешно завершен. Одно решение найдено. На следующем рисунке представлено окно вывода программы.



The screenshot shows a Windows command prompt window titled "rt (C:\temp\1\1) - Visual Prolog Environment 7.5 (Build 7502) Personal Edition Polovikova Olga N. - [main.pro]". The command executed is "C:\temp\1\Exe>"C:\temp\1\Exe\rt.exe"". The output lists five students with their names, group numbers, and scores, followed by a final line showing a student with a stipend.

Имя	гр	балл
Имя: Миша	гр: 443	балл: 50
Имя: Лена	гр: 441	балл: 60
Имя: Коля	гр: 441	балл: 57
Имя: Оля	гр: 442	балл: 39
Имя: Гая	гр: 441	балл: 79

Имя: Гая гр: 441 балл: 79 стипендия: 110

Рис. 1 Консольное окно программы в Visual Prolog

Замечание. Отметим, что в зависимости от Prolog-системы, отвечающей за поиск решения заданной цели, на экран будут выведены либо все данные о студентах, удовлетворяющие критерию, либо данные только одного студента (первого в базе данных).

Visual Prolog «по умолчанию» находит только **одно решение**, удовлетворяющее критериям поиска целевого предиката. Поиск решения согласно указанной цели к задаче 2 завершится, как только будет найден первый студент из базы знаний, у которого стипендия выше базовой. Чтобы вывести всех студентов, удовлетворяющих критерию, следует указать Prolog-системе, что «Найденное решение неверно, следует искать ещё». Такой искусственный прием поиска всех возможных комбинаций, удовлетворяющих заданному критерию, возможен благодаря использованию предиката *fail*, который всегда согласуется Prolog-

системой со значением ложь *{false}*. Поэтому его использование позволяет многократно согласовывать всю цепочку предикатов целевого утверждения, многократно связывать используемые переменные.

Замечание. Чтобы быть языком общего назначения, Visual Prolog должен предоставлять ряд сервисных функций, например, процедур ввода/вывода. Они реализованы как предикаты без специального логического смысла, которые всегда оцениваются как истинные и выполняют свои сервисные функции как побочный эффект оценивания.

Примеры: *write, read, nl, readline, readChar*. Данные функции реализованы в пакете *console* и *stdio*.

Лекция 2. Программа на языке Visual Prolog

Предложения Хорна в программе на языке Visual Prolog разбиты по разделам с использованием определенных ключевых слов.

Программа на Visual Prolog состоит из кода языка Prolog, который разбит по разделам соответствующими ключевыми словами. Каждое ключевое слово информирует компилятор кода о свойствах генерации. Например, есть ключевые слова, которые различают для предикатов их объявления от их определений. Каждый раздел определяется ключевым словом. Существует правило – нет ключевого слова, которое означает окончание определенной секции. Наличие другого ключевого слова указывает на окончание в предыдущем разделе и начало следующего.

Исключением из этого правила являются ключевые слова *implement* (реализация) и *end implement* (конец реализации). Код, содержащийся между этими двумя ключевыми словами, используется для определенного класса и для кода главной программы.

Задача. Напишите программу, которая по определенным характеристикам людей, находит всех бабушек. База знаний определяет свойства людей, включая их родственные отношения.

Для решения задачи сформируем новый **домен** – *человек* и предикат – *бабушка*. Рассмотрим следующий код программы, где вывод всех возможных исходов возможен за счёт использования цикла *foreach...end foreach*.

```
implement main  
open core, console
```

```
domains
```

```
человек=человек(string, char, integer).  
% человек (имя, пол, возраст)
```

```
class facts  
родитель: (человек, человек).
```

```
clauses  
родитель(человек ("Лена", 'ж', 40) , человек ("Галя", 'ж', 7)).  
родитель(человек ("Лена", 'ж', 40) , человек ("Дима", 'м', 12)).  
родитель(человек ("Оля", 'ж', 60) , человек ("Лена", 'ж', 40)).  
родитель(человек ("Коля", 'м', 39) , человек ("Галя", 'ж', 7)).  
родитель(человек ("Коля", 'м', 39) , человек ("Дима", 'м', 12)).  
родитель(человек ("Оля", 'ж', 60) , человек ("Саша", 'м', 36)).  
родитель(человек ("Саша", 'м', 36) , человек ("Петя", 'м', 11)).  
родитель(человек ("Мира", 'ж', 34) , человек ("Петя", 'м', 11)).
```

```
class predicates  
бабушка: ( человек) nondeterm (o).
```

```
clauses  
бабушка(X):- X= человек( _, 'ж', _),  
родитель(X, Y), родитель(Y, Z),
```

```
clauses  
run():-init(), foreach бабушка(X) do write(X), nl end foreach,  
_=readChar().
```

```
end implement main
```

```
goal  
console::run(main::run).
```

Замечание. Использование цикла **foreach...end foreach** позволяет добиться схожего результата, как и при применении встроенного предиката *fail*. Происходит перебор возможных исходов некоторой цепочки предикатов и связывание соответствующих переменных.

Ключевое слово **open** используется для расширения области видимости класса. Оно должно использоваться только после ключевого слова **implement** (реализация).

Ключевое слово **constants** обозначает раздел с описанием констант. Константы имеют символические имена, которым соответствуют значения. При компиляции имя заменяется значением.

Ключевое слово **domains** предваряет описание новых доменов. Каждый домен содержит описание нового типа данных. Если используются только стандартные типы, то этого раздела в программе нет.

Ключевое слово **facts** обозначает раздел, который объявляет факты, которые будут использоваться позже в коде программы. В этом разделе расположены предикаты внутренней базы данных, которые истинны. Каждый факт объявляется с именем, которое используется для обозначения факта, и аргументами вместе с доменами, которым эти аргументы принадлежат.

Ключевое слово **predicates** обозначает раздел, который содержит только объявления предикатов, которые будут позднее определены в разделе **clauses** (предложения) кода. Однажды объявленные имена, используемые для этих предикатов вместе с аргументами и доменами, к которым принадлежат аргументы, будут указываться в этом разделе.

Ключевое слово **clauses** обозначает раздел, который содержит определения ранее объявленных предикатов. Из всех разделов, которые присутствуют в коде Visual Prolog, этот раздел наиболее близко имитирует традиционную программу Prolog. Сюда записываются факты и правила, которыми будет оперировать VisualProlog, пытаясь решить цель программы.

Ключевое слово **goal** обозначает раздел, который определяет главную точку входа в программу на языке Visual Prolog. Раздел объявлен. Здесь формулируется цель программы: запуск исполняемого файла.

Рассмотрим некоторые разделы подробнее.

Раздел доменов

Если для описания предметной области требуются типы данных, отличающиеся от стандартных для Visual Prolog. В этом случае следует задать новые домены и поместить их в код.

Форма задания домена следующая:

domains

<имя1>, <имя2> = <имя известного домена>

Каждое строка задания доменов завершается символом новой строки. Объявление новых доменов, благодаря присваиванию осмысленных имен типам аргументов, помогает документировать описываемые предикаты.

Домены позволяют задавать разные имена различным видам данных, которые, в противном случае, будут выглядеть абсолютно одинаково. В программах Visual Prolog объекты в отношениях (аргументы предикатов) принадлежат доменам, причем это могут быть как стандартные, так и описанные пользователем специальные домены.

Кроме того, что осмысленные имена доменов помогают документировать код программы, специальные домены ещё используются для описания структур данных, которые отсутствуют в Visual Prolog.

Аргументы с типами из специальных доменов не могут смешиваться между собой, даже если эти домены одинаковы. Если переменная в предложении используется более чем в одном предикате, она должна быть одинаково объявлена в каждом из них.

Раздел констант

В коде на языке Visual Prolog можно объявлять и использовать *символические константы*. Для объявления констант следует использовать следующий синтаксис:

`constants`

`<имя> = <Макроопределение>.`

Имя для символической константы должно быть оформлено в нижнем регистре, а *макроопределение* – это значение для константы. Каждое макроопределение завершается точкой. На одной строке может быть только одно описание константы.

Примеры описания констант

`constants`

`n = 100.`

`str = "Good day".`

`summa = n*0.7+500.`

Перед компиляцией программы будет произведено замещение каждой константы соответствующим значением.

Символические константы подчиняются следующим правилам:

1. Описание константы не может ссылаться само на себя, это приведет к сообщению об ошибке.
2. В описаниях констант не различается верхний и нижний регистры.
3. В программе может быть несколько разделов с константами, любое объявление константы должно производиться перед ее использованием.
4. Идентификаторы констант являются глобальными и могут объявляться только один раз.

Раздел предикатов

Вспомним, что предикатами являются функции, домены которых отображаются в множество: $\{true, false\}$. В Visual Prolog существуют встроенные предикаты, которые объявлять не надо. Приведём примеры встроенных предикатов:

$X > Y$ есть *true*, если X больше, чем Y , иначе возвращается *false*;

$X < Y$ есть *true*, если X меньше, чем Y , иначе *false*;

$X = Y$ есть *true*, если X равен Y , иначе *false*.

Если в разделе *clauses* программы на языке Visual Prolog описан собственный предикат, то его необходимо предварительно объявить в разделе *predicates* (предикатов). При объявлении предиката сообщается, к каким доменам (типам) принадлежат его аргументы.

Необходимо придерживаться следующего синтаксиса:

имя_предиката (тип_Aрг1, тип_Aрг2,..., тип_AргN).

Каждое построение предиката должно быть оформлено с новой строки и завершаться точкой. Доменами (типами) аргументов предиката могут быть либо стандартные домены, либо новые домены, объявленные в разделе *domains*.

Примеры предикатов:

predicates

стипендия (integer, real).% см. решение задачи 1

birthday : (человек, date).

% домен человек описан для решения задачи 3

Арность (размерность) предиката – это количество аргументов, которые он принимает. Можно объявить в одной программе два предиката с одним и тем же именем, но различной арностью. В разделах *predicates* и *clauses* версии предикатов с одним именем и разной арностью должны записываться вместе; за исключением этого ограничения, различная арность всегда понимается как полное различие предикатов.

Раздел предложений

В раздел *clauses* (предложений) помещаются все факты и правила, составляющие программу. Напомним, что все предложения для каждого конкретного предиката в разделе *clauses* должны располагаться вместе. Последовательность предложений, описывающих один предикат, называется **процедурой**.

Поиск решения для заданного целевого утверждения Visual Prolog начинает с первого предложения раздела *clauses*. Prolog-система просматривает каждый факт и каждое правило, стремясь найти сопоставление соответствующим предикатам из цели. По мере продвижения вниз по разделу *clauses*, Prolog-система устанавливает внутренний указатель на первое предложение, являющееся частью пути, ведущего к решению. Если следующее предложение не является частью этого логического пути, то Visual Prolog возвращается к установленному указателю и ищет очередное подходящее сопоставление, перемещая указатель на него (этот процесс называется **поиск с возвратом**). Prolog-система просматривает дерево решений заданного целевого утверждения, выполняя поиск с возвратом.

Раздел фактов

Программа на Visual Prolog представляет собой набор фактов и правил. Иногда в процессе работы программы бывает необходимо модифицировать (изменить, удалить или добавить) некоторые из фактов, с которыми она работает. В этом случае факты рассматриваются как динамическая или внутренняя база данных, которая при выполнении программы может изменяться. Часть фактов можно загружать из текстовых файлов, другую часть может строить Prolog-система в зависимости от ситуации. Для объявления фактов из постоянной или из динамической (изменяющейся) базы данных используют специальный раздел *facts*.

Каждое задание факта начинается с новой строки и завершается точкой. Форма задания факта:

`facts`

`<имя отношения> : (список объектов).`

Фактом можно записать отношение между объектами или свойства объекта. Ниже представлен пример описания раздела с фактом, который выражает свойство объекта:

`facts`

`студент : (string, string, integer).`

В лекции рассмотрены разделы программы на языке Visual Prolog. Каждый раздел начинается с ключевого слова, которое информирует компилятор о свойствах генерации кода.

Лекция 3. Функциональный язык Lisp. Списки

Функциональное программирование возникло в 1962 г. вместе с созданием Дж. Маккарти языка программирования Lisp. В качестве примера использования этого языка не для написания систем искусственного интеллекта можно выделить систему инженерного проектирования AUTOCAD, где всё математическое обеспечение написано на языке Lisp, также в этой системе есть свой встроенный диалект языка Lisp (AutoLisp), который позволяет упростить и автоматизировать процесс создания инженерных проектов.

Язык Lisp – один из первых языков обработки данных в символической форме. Его название происходит от английских слов «list processing» – «обработка списков». В Lisp программа и обрабатываемые ею данные представляются в одной и той же форме – в форме списка. Таким образом, программа может обрабатывать и преобразовывать другую программу и даже саму себя.

Функциональный подход программирования, на котором базируется Lisp, можно выразить следующим принципом: вся обработка ин-

формации и получение искомого результата могут быть представлены в виде вложенных и/или рекурсивных вызовов функций, выполняющих некоторые действия, так что значение одной функции используется как аргумент другой. Решение задачи строится на последовательном вызове функций, когда результат одной функции становится аргументом следующей и т.д., пока не будет получен конечный результат.

Программы языка Lisp строятся из определений функций. Определения состоят из управляющих структур, организующих вычисления, и из вложенных вызовов функций. Основными методами функционального программирования являются композиция и рекурсия. Все это представляет собой реализацию идей теории рекурсивных функций.

Программы на языке Lisp строятся из простейших неделимых элементов, называемых **атомами**. Символы и числа представляют собой атомы, из них состоят все остальные структуры. Символы могут состоять как из прописных, так и из строчных букв, **прописные и строчные буквы отождествляются** и представляются прописными буквами. Кроме символов в программах на языке Lisp используются **числа**, которые, как и символы, записываются при помощи ограниченной пробелами последовательности знаков.

Числа не являются символами, так как число не может представлять другие объекты, кроме своего числового значения. Как и в других языках программирования, в Lisp для различных целей используется много различных типов чисел. Числа (значения) могут быть целыми (например, 543), десятичными (например, 3.789), или представляться мантиссой и порядком (например, 1.0243E-6), или представляться дробями (например, 4/5 и -17/2).

Логические значения T и NIL

Символы *T* и *NIL* используются в языке Lisp для специальных целей: *T* – обозначает логическое значение истина (true), а *NIL* – логическое значение ложь (false). (В некоторых системах для обозначения логического значения ложь используется специальный символ *F* (false). Символом *NIL* обозначается также и пустой список. Символы *T* и *NIL* имеют всегда одно и то же фиксированное встроенное значение. Их нельзя использовать в другом качестве.

Константы и переменные

Числа и логические значения *T* и *NIL* являются *константами* (constant), остальные символы – *переменными* (variable), которые используются для обозначения других объектов языка Lisp. Кроме этого,

язык Lisp (его диалект Common Lisp) умеет оперировать со специальными *глобальными переменными*, имеющие встроенные значения.

В языке Lisp (в диалекте Common Lisp) предусмотрена специальная директива **DEFCONSTANT**, используемая для превращения любого символа в константу. Символ, определенный как константа, может обозначать лишь то значение, которое ему предписано.

Символы и числа представляют собой простейшие объекты, из которых строятся остальные структуры. Поэтому их называют **атомарными объектами** или просто **атомами (atom)**.

Списки

Атомы и списки (list) – это основные типы данных языка Lisp. В естественном языке под **списком** понимают некоторый перечень необязательно однотипных элементов. **Списком** на языке Lisp называется упорядоченная последовательность, элементами которой являются либо *атомы*, либо *списки (подсписки)*. Списки заключаются в круглые списки, а их элементы разделяются пробелами.

Например, следующая последовательность атомов и подсписков является списком:

(a b (c d) e).

Список, в котором нет ни одного элемента, называется **пустым** списком и обозначается пустыми круглыми скобками: () или специальным символом *NIL*.

Пустой список несёт особую смысловую нагрузку. Это как пустое множество в теории множеств, которое необходимо для корректного определения разнообразных операций над элементами множества.

Имеется и альтернативный способ записи списков – с использованием точечной нотации. Точкой отделяют начальный элемент списка – его голову от остальной части списка – хвоста: (*голова*. *хвост*).

(a1 a2 ... aN) = (a1. (a2.... (aN. Nil)...)).

Списки в Лиспе – средство (декларативный способ) представления знаний. Например, с помощью вложенных списков можно охарактеризовать сам язык:

(язык Lisp

(автор Маккарти)

(парадигмы функциональная объектно-ориентированная)

(диалекты (Common Lisp) (Home lisp) (Auto Lisp) ...)

...)

Различная интерпретация списков

Одним из основных отличий языка Lisp от традиционных языков программирования является запись в Лиспе в виде списков не только, данных, но и блоков программы (функций).

Например, список: $(+ 2 3)$ можно интерпретировать в зависимости от окружения и использования либо как действие, дающее в результате число 5, либо как список, состоящий из трех элементов. Если нужно интерпретировать это выражение как список с данными, необходимо заблокировать его вычисление. Для обозначения списка, используемого как данные, т.е. для блокировки вычислений в языке Lisp определена функция *QUOTE*:

$(\text{QUOTE } (a\ b\ c))$ или $'(a\ b\ c)$

Запись вида: $(a\ b\ c)$ интерпретируется как вызов функции с именем a и фактическими параметрами b, c .

Функции

Функция с точки зрения языка Lisp – отображение между множествами. Различают *определение* функции и *вызов* (т.е. применение функции). В языке Lisp как для вызова функции, так и для определения действий функции принята единообразная префиксная форма записи: имя функции или имя действия, сами аргументы записываются внутри скобок:

$(\text{defun } (f\ x\ y)\ (+\ x\ y))$

$(*\ (+\ 5\ (f\ 4.5\ -7/8))\ 4)$

Интерпретатор Лиспа

Lisp напоминает естественный язык, так как обычно используется в диалоге (интерактивно) и в режиме интерпретации (interpretation). **Интерпретатор** языка Lisp работает по следующей схеме:

1. Символ подчеркивания "_" показывает, что ждет очередное выражение от пользователя.
2. Когда пользователь заканчивает ввод какого-либо вызова функции или другого выражения, то интерпретатор вычисляет и выдает значение этого выражения.

Иерархия вызовов

В конструкцию какой-либо функции могут входить другие функции или вычисляемые выражения. Пример:

$_ (*\ (+\ 1\ 2)\ (+\ 3\ 4))$

Вычислять значения выражения, интерпретатор языка Lisp сначала пытается слева направо вычислить значения аргументов внешнего вызова. Если первый аргумент является константой или иным объектом, значение которого можно определить непосредственно, то вы-

числение аргументов вычисляемой функции можно продолжить. Если аргументом будет вложенный вызов функции или другая вычисляемая форма, то перед ее вычислением определяются значения ее аргументов и т.д.

Лекция 4. Функции обработки списков

В языке Lisp для обработки символьных выражений (атомов и списков) используют следующие базовые функции: *CAR*, *CDR*, *CONS*, *ATOM*, *EQ* и другие.

Функции по принципу их использования можно разделить на функции **разделения** (разбора), **создания** и **проверки**:

Использование	Вызов	Результат
Разделение	(<i>CAR</i> список)	атом или список
	(<i>CDR</i> список)	список
Создание	(<i>CONS</i> список список)	список
	(<i>CONS</i> атом список)	список
Проверка	(<i>ATOM</i> s-выражение)	Т или NIL
	(<i>EQ</i> символ символ)	Т или NIL

Данные функции можно представить и следующим образом:

1. Функции-селекторы *CAR* и *CDR*.
2. Функцию-конструктор *CONS*.
3. Предикаты *ATOM*, *EQ*, *EQUAL*.

Селекторы

Селектором называется функция, осуществляющая выборку элемента объекта данных. Функция *CAR* возвращает в качестве значения голову списка. Функция *CDR* возвращает в качестве значения хвост списка.

Примеры:

Список	<i>CAR</i>	<i>CDR</i>
(a (b) (cd ab))	a	((b) (cd ab))
((a b) c) (b c))	((a b) c)	((b c))
(a)	a	nil
()	nil	nil

Конструктор

Конструктором называется функция, осуществляющая построение объекта данных. Функция *CONS* строит новый список из переданных ей в качестве аргументов произвольного списка или атома и спис-

ка. При этом первый аргумент включается в список-результат в качестве головы, второй – в качестве хвоста.

Примеры:

Ввод	Результат
(CONS (a) (b c))	((a) b c)
(CONS a NIL)	(a)
(CONS (a) (b c))	((a) b c)

Связь между конструкторами и селекторами

Функции *CAR* и *CDR* являются обратными для конструктора *CONS*. Пример:

```
_(CONS (CAR `(a b c d))(CDR `(a b c d)))
```

дает результат:

```
`(a b c d)
```

Композиция селекторов

Путем комбинации селекторов *CAR* и *CDR* можно выделять произвольный элемент списка. В языке Lisp допускается сокращенная запись композиции нескольких селекторов в виде одного вызова функции.

Например, для списка списков: *‘((a b)(1 2)(3 4 5 6)(7 8 9) 10)* необходимо выделить второй элемент третьего подсписка.

Решение

Выделим третий подсписок:

```
_(CAR (CDR (CDR `(a b)(1 2)(3 4 5 6)(7 8 9) 10))))
```

Результат:

```
`(3 4 5 6)
```

Выделим второй элемент третьего подсписка:

```
_(CAR (CDR (CAR (CDR (CDR `(a b)(1 2)(3 4 5 6)(7 8 9) 10))))))
```

Результат:

```
4
```

Сокращенная запись:

```
_ (CADAR (CDDDR `(a b)(1 2)(3 4 5 6)(7 8 9) 10)))
```

Результат:

```
4
```

Конструирование списков

Кроме функций низкого уровня *CAR*, *CDR* и *CONS* для построения списков в языке Lisp существуют функции высокого уровня. Функция *LIST* создает список из произвольных элементов. Вызов функции *LIST* эквивалентен суперпозиции вызовов *CONS*, причем вторым аргументом последнего вызова является пустой список *NIL*. Пустой список служит основой для наращивания нового списка.

Пример:

```
_(LIST 'a 'b 'c 'd)
```

Эта инструкция эквивалентна следующему выражению:

```
_(CONS 'a (CONS 'b (CONS 'c (CONS 'd nil))))
```

Для реализуемой с помощью селекторов и их суперпозиции выборки элементов списков в языке Lisp (в диалекте Common Lisp) существуют функции выделения заданного элемента: *FIRST*, *SECOND*, *THIRD*, *FOURTH* и другие, а также общая функция *NTH*, выделяющая *n*-й элемент списка:

(*NTH* *n* список).

Замечание. Функция *NTH* отсекает элемент списка, начиная с нулевого элемента.

Пример:

```
_(NTH 5 '(1 2 3 4 5))
```

Результат:

NIL

Пример:

```
_(NTH 4 '(1 2 3 4 5))
```

Результат:

5

Последний элемент списка можно выделить с помощью функции *LAST*. При этом вызов (*LAST* список) возвращает последний непустой хвост списка *список*. Пример:

```
_(CAR(LAST '(A B C)))
```

Результат:

C

В данном примере использование селекторной функции позволило получить атом *C* из списка *'(C)*.

Задания для практических занятий

Среда Visual Prolog

Лабораторная работа № 1

Задача 1. Построить базу фактов для описания свойств людей и отношений между ними: родитель-ребёнок. Например, за основу можно взять следующие предикаты:

человек (имя, пол, возраст)

родитель (имя родителя, имя ребёнка)

Дополнительно следует построить правила для определения отношений между родственниками: родной брат/сестра, бабушка/дедушка, тётя/дядя. Используя сформированную базу фактов и правил, научиться находить ответы на разные вопросы, например:

- Найти родного брата/сестру для мальчика Коли.
- Найти всех дедушек моложе 55 лет.
- Найти всех тёть, у которых несколько родных племянников.

Задача 2. Построить предикат для вычисления факториала заданного числа.

Задача 3. Составить программу для вычисления *НОД*. Если заданы два целых числа X и Y , то их наибольший общий делитель можно найти, руководствуясь следующими тремя правилами:

1. Если X и Y равны, то *НОД* равен X .
2. Если $X < Y$, то *НОД* равен наибольшему общему делителю X и разности $Y - X$.
3. Если $Y > X$, то формулировка аналогична правилу 2, если X и Y поменять местами.

Задача 4. Разработать процедуру предложений для разбиения предложения на слова.

Задача 5. Составить программу, которая выводит на экран все арабские цифры, символьная запись которых есть в задаваемом предложении.

Задача 6. Построить процедуру предложений для реализации операции удаления заданного элемента из списка.

Задача 7. Для заданного списка чисел построить процедуру предложений для определения элемента с максимальным по модулю значением.

Задача 8. Для заданного списка строк построить новый список строк, в который включить только те элементы списка, длина которых больше некоторого значения N .

Лабораторная работа № 2

Задача 9. Построить процедуру предложений (*include*) для определения принадлежности элемента списку, проверить работу данной процедуры (пример, элемент 34 принадлежит списку [23, red, 34, 43], а элемент *green* не принадлежит списку [red, yellow, blue, white]).

Задача 10. Построить процедуру предложений (*connect*) для реализации операции сцепления (соединения) двух списков, результатом выполнения которой будет получен третий список, например: *connect([a, b, c, d], [aa, bb, cc], L)*, где *L* исходящий параметр-результат: [a, b, c, d, aa, bb, cc].

Лабораторная работа № 3

Задача 11. База данных содержит информацию о забронированных посадочных местах на некоторый рейс. Посадочные места, которые можно бронировать, определены списком *L1*. Потенциальные пассажиры, для которых нужно забронировать места, заданы списком *L2*. Места, которые необходимо освободить от бронирования, определены списком *L3*. Разработать программу, которая должна выполнить следующие действия:

1. Освободить забронированные посадочные места из списка *L3*.
2. Забронировать каждому пассажиру из списка *L2* место оставшихся позиций списка *L1*.

Учесть, что все изменения должны действовать только на момент работы программы.

Подсказка

Бронирование места с номером *X* пассажиром *Y* может определяться в базе данных фактом *mesto(Y, X)*. Процедура *rasp(L2, L1)* должна рекурсивно добавлять во временную БД факты вида *mesto(Y, X)*, где *Y* элемент списка *L1*, *X* элемент списка *L2*. Процедура *rasp* может быть реализована следующим образом:

```
rasp([],_).
rasp([X|Y],[A|B]):- not(mesto(_,A)),
добавить_факт_в_БД(mesto(X,A)), rasp(Y,B).
rasp([X|Y],[A|B]):- mesto(_,A), rasp([X|Y],B).
```

Задача 12. В текстовом файле хранятся строки произвольной длины. Построить функции для вычисления максимальной и минимальной длины строк файла.

Задача 13. В текстовом файле хранятся факты для временной БД вида:

firma(Уникальный номер, Название, Телефон)
products(Уникальный номер фирмы, [список названий товаров
из ассортимента фирмы]).

Построить программу, которая будет обеспечивать заполнение временной БД, а также решать следующие подзадачи:

1. Для заданного названия товара определить и вывести на экран названия фирм с телефонами, в ассортименте которых присутствует данный товар.
2. Для заданного названия фирмы определить и вывести на экран полный ассортимент товаров.

Для интерпретатора Common Lisp

Лабораторная работа № 4

Задача 14. Напишите выражение, вычисляющее среднее арифметическое чисел 23, 5, 43 и 17.

Задача 15. Найти сумму 100 первых членов арифметической прогрессии, если $a_2=2/7$ и $a_3=4/7$. Формулу записать одним выражением.

$$S_n = \sum_{i=1}^n a_i = \frac{a_1 + a_n}{2} n = \frac{2a_1 + d(n-1)}{2} n$$

Задача 16. Определите значения следующих выражений:

- a) '(+2 (* 3 4))
- b) (+ 2 '(* 3 4))
- c) (+ 2 ('* 3 4))
- d) (+ 2 (* 3 '4))
- e) (quote 'quote)
- f) (quote 2)
- g) '(quote NIL)

Задача 17. Вычислите:

- a) расстояние между двумя точками $A(10, 2.5e+1, 3.0e-2)$ и $B(-5, 3/7, -10e-3)$ в Декартовой системе координат $OXYZ$,
- б) расстояние между двумя точками A и B в Декартовой системе координат OXY , если координаты этих точек задаются случайным образом.

Задача 18. Вычислить площадь круга, радиус которого должен задавать пользователь.

Задача 19. Запишите последовательность вызовов *CAR* и *CDR*, которая позволит получить символ *cat* для следующих списков. Упростите эти вызовы с помощью сокращенной формы вызова селекторов.

- a) (1 2 (cat) 3 4)
- b) ((1) (2 7 cat) (3 (4)))
- c) ((1 (2 (3 (4 (cat))))))

Задача 20. Вычислите значения следующих выражений. Проверьте результат интерпретатором.

- a) (cons nil '(cdr car list))
- b) (cons nil nil)
- c) (cons '(nil) '(nil))
- d) (cons (car '(a b)) (cdr '(a b)))
- e) (car '(car (a b c)))
- f) (cdr (car (cdr '(a b c))))
- g) (list (list 'a 'b) '(car (c d)))

Задача 21. Постройте список следующего вида $((+ a b) (- c d))$, где числа a и c задает пользователь, b и d задаются случайным образом.

Задача 22. Постройте список следующего вида $((L) A (B) C (D))$, где L, A, B, C, D – это числа, которые являются результатами возведения в степень некоторых чисел, задаваемых пользователем. При этом степени, в которые возводятся числа, должны определяться случайным образом.

Лабораторная работа № 5

Задача 23. Каковы будут значения следующих вызовов при условии, что значением X является Y , а значением Y – X :

- a) (set x y)
- b) (setq x y)
- c) (setq x y)

Задача 24. Каково будет значение атома A после следующих вызовов:

- a) (set (setq a 'a) 'b)
- b) (set (setq b 'a) (setq a 'c))
- c) (set b a)

Задача 25. Вычислите значения следующих выражений:

- a) '(car '(a b c))
- b) (eval '(car '(a b c)))
- c) (eval (car '(a b c)))
- d) (eval (quote (quote quote)))
- e) (quote (eval (quote (quote quote))))

Задача 26. Какие из следующих вызовов возвращают значение T (истина)?

- a) (atom '(cdr nil))

- b) (equal '(a b) (cons '(a) '(b)))
- c) (atom (* 2 (+ 2 3)))
- d) (null (null t))
- e) (eq nil (null nil))
- f) (eq 2.0 2)
- g) (equal 2.0 2)
- h) (= 2.0 2)
- i) (equalp 2.0 2)
- j) (equalp (atom nil) (car '(())))

Задача 27. Построить функцию, которая первый формальный параметр возводит в степень максимального значения из второго и третьего параметра.

Задача 28. Построить функцию для вычисления факториала некоторого положительного числа.

Задача 29. Построить функцию для вычисления НОД для двух целых положительных чисел.

Задача 30. Построить функцию-предикат от двух аргументов, которая возвращает истину, если первый аргумент является элементом первого уровня списка, задаваемого вторым аргументом.

Задача 31. Построить функцию, которая вычисляет сумму заданного списка атомов-чисел.

Лабораторная работа № 6

Задача 32. Используя предложения (форму) *COND*, следует запрограммировать функцию *WHEN_* от двух аргументов: *условие*, *выражение в случае истинности условия*. Пример вызова:

`_(WHEN_ '(or (eq a t) (eq b t)) 't)`

Задача 33. Запрограммируйте с помощью предложения *DOTIMES* итеративную версию функции для вычисления факториала.

Задача 34. Определите функцию (*production x y*), вычисляющую произведение двух целых положительных чисел (мультипликативные операции использовать нельзя).

Задача 35. Функция (*LENGTH x*) является встроенной функцией, которая возвращает в качестве значения длину списка (количество элементов на верхнем уровне). Определите функцию *LENGTH_* сначала рекурсивно с помощью предложения *COND* и затем итеративно с помощью предложений для организации циклов.

Задача 36. В математике числа Фибоначчи образуют ряд: 0, 1, 1, 2, 3, 5, 8, Эту последовательность можно определить с помощью следующей функции *fib*:

fib(*n*) = 0, если *n*=0

$fib(n) = 1,$ если $n=1$
 $fib(n) = fib(n-1)*fib(n-2),$ если $n>1$.

Определите функцию (*fib n*), **итеративно** и (а затем **рекурсивно**) вычисляющую *n*-й элемент ряда Фибоначчи.

Задача 37. Используя предикат сравнение «на больше» и условное предложение, необходимо описать функцию, которая возвращает из трех числовых аргументов значение среднего по величине числа:

`_(average 5 7 3) % ответ 5`

Задача 38. Нарисуйте следующие списки при помощи списочных ячеек и стрелок:

- a) (a)
- b) (a (b (c) d))
- c) (nil (2. c) . d)

Задача 39. Постройте список `'((a b c) a b c)`, у которого голова и хвост ссылаются на физически идентичные ячейки.

Задача 40. Вычислите значения следующих выражений:

- a) `(rplacd '(a) 'b)`
- b) `(rplaca '(a) 'b)`
- c) `(rplacd (cddr '(a b x)) 'c)`
- d) `(rplacd '(nil) nil)`
- e) `(rplacd '(nil) '(nil))`

Задача 41. Что делает следующая функция?

```

(defun fun (x) (cond ((null x) x)
                     (t (rplacd x (fun (cdr x)))))
)

```

Индивидуальные задания по вариантам

Для визуальной среды Visual Prolog

1. Поиск решения с возвратом

Вариант 1. В небольшой фирме работает (10-15) сотрудников. Каждый сотрудник характеризуется персональными данными (фамилия, имя, год приема на работу), занимает определенную должность и в зависимости от этого оклад (бухгалтер 300 ед., директор 500 ед., оператор 250 ед., инженер 400 ед.). Сотрудники работают либо в основном офисе, либо в филиале (операторы и бухгалтера в основном офисе, остальные в филиале).

Создать программу на языке Пролог, которая будет отвечать на следующие вопросы:

1. Вывести список сотрудников, работающих в филиале.
2. Вывести список операторов.
3. Для заданной фамилии сотрудника определить зарплату, которая складывается из суммы оклада и премии, премия вычисляется из расчета 100 ед., если сотрудник проработал в фирме больше 5 лет, 40 ед. иначе.

Вариант 2. В ведомственной группе детского сада (10-15 детей) у каждого ребенка хотя бы один родитель работает либо в спортивной школе, либо в школе олимпийского резерва. Каждый ребенок характеризуется личными данными: фамилия, имя, пол, год рождения. Каждый месяц родители оплачивают пребывание ребенка в детском саду.

Если оба родителя работают в спортивной школе, тогда ежемесячная оплата за детский сад на каждого ребенка составляет 50% от номинальной стоимости. Если только один родитель работает в спортивной школе, то льгота на каждого ребенка составляет 30%. Для остальных родителей льгота составляет 10% на каждого ребенка, если в семье более одного ребенка посещают данную группу. Остальные родители ежемесячно оплачивают полную номинальную стоимость.

Создать программу на языке Пролог, которая будет отвечать на следующие вопросы:

- a. Вывести список детей, у которых родной(ая) брат(сестра) посещают эту группу детского сада.
- b. Вывести список детей, у которых оба родителя работают в спортивной школе.
- c. Для заданного ребенка и заданной номинальной стоимости пребывания в детском саду определить родительскую оплату.

Вариант 3. В конструкторском бюро работает от 10 до 15 инженеров, каждый характеризуется персональными данными: фамилия, имя,

должность (инженер, старший инженер, конструктор), ученая степень (без степени, кандидат наук, доктор наук). Конструкторы первые шесть месяцев года работают в институте, остальные – в бюро. Старшие инженеры в институте работают только в зимнее календарное время, остальное время – в бюро. Инженеры в бюро работают только в зимнее и летнее календарное время, остальное время – в институте.

Оплата в институте составляет: 250, 300 и 500 ед. соответственно для инженера, старшего инженера и конструктора. В бюро оплата меньше на 15% для инженеров всех должностей, кроме докторов наук. Создать программу на языке Пролог, которая будет отвечать на следующие вопросы:

1. По указанному номеру месяца года вывести список сотрудников с персональными данными и местом работы.
2. Вывести список докторов или кандидатов наук, которые работают в должности инженера.
3. Для заданного сотрудника и заданного месяца года определить его оплату.

Вариант 4. Авиакомпания обслуживает рейсы (10-15 рейсов), которые характеризуются следующими параметрами: город отправления, город прибытия, время в пути. В каждом городе прибытия своя стоимость аренды аэропорта. Государство дополнительно с каждого рейса взимает стоимость использования воздушного коридора, которая зависит от типа города прибытия: южного направления – 20% к стоимости аренды аэропорта, северного направления – 10% к стоимости аренды, зарубежные города – 100 ед.

Создать программу на языке Пролог, которая будет отвечать на следующие вопросы:

1. Вывести список городов, в которые можно долететь из указанного города максимально с одной пересадкой, рассчитать суммарное время в пути.
2. Для заданного города вывести список всех городов, в которые можно долететь рейсами авиакомпании со стоимостью аренды и воздушного коридора.
3. Для заданного города вывести список всех прибывающих в него прямых рейсов и рейсов с одной пересадкой.

Вариант 5. Авиакомпания обслуживает рейсы (10-15 рейсов), которые характеризуются следующими параметрами: номер рейса, город отправления, город прибытия, время отправления в часах (14:30 это есть 14+30/60), время в пути в часах (2 часа 20 минут это есть 2+20/60), номер дня недели отправления рейса.

1. Вывести список всех прямых рейсов в указанный город прибытия на указанный день недели.
2. Вывести список всех возможных городов прибытия из указанного города с одной пересадкой, необходимо чтобы время ожидания пассажиров в аэропорту не превышало 5 часов.
3. Вывести список всех городов отправления, из которых можно прилететь в указанный город с одной пересадкой, необходимо, чтобы время ожидания пассажиров в аэропорту не превышало 4 часов.

Вариант 6. Фирма принимает заказы на изготовление трех видов деталей (f106, af106,bf107). Каждое утро в понедельник диспетчер формирует базу заказов (10-15 заказов), каждый заказ оформляется следующими характеристиками: номер заказа, вид детали, объем в некоторых единицах, срок выполнения – номер дня недели (от 1 до 7). Для всех трёх видов деталей технология изготовления одинакова: вначале делают заготовку, потом её обрабатывают, время выполнения каждого этапа зависит от вида детали: 3ч и 2ч, 2ч и 1ч, 4ч и 1ч. на одну единицу объёма соответственно для деталей f106, af106,bf107.

Создать программу на языке Пролог, которая будет отвечать на следующие вопросы:

1. Вывести список деталей, у которых время выполнения заготовки превышает 12 часов.
2. Вывести список деталей, которые нельзя изготовить в срок (рабочее время сотрудников 8 часов в день).
3. Вывести список деталей указанного вида, которые можно изготовить за указанное количество рабочих дней раньше от установленного срока.

Вариант 7. За одной из кафедр вуза закреплено от 10 до 15 учебных дисциплин. Каждая дисциплина в вузе характеризуется: названием, курсом

(1-6), преподавателем, объемом часов. Оклад преподавателя зависит от занимаемой должности и его ученой степени. Любому преподавателю начисляется один из 5 окладов:

ст. преподаватель без научной степени – 40 ед.

ст. преподаватель и кандидат наук – 50 ед.

доцент и кандидат наук – 60 ед.

профессор и кандидат наук – 80 ед.

профессор и доктор наук – 100 ед.

Создать программу на языке Пролог, которая будет отвечать на следующие вопросы:

1. Для заданного значения вывести все дисциплины, объем которых превосходит указанный предел и которые преподают доктора наук.

2. Вывести всех преподавателей кандидатов наук или без научной степени, которые работают в должности старшего преподавателя и ведут более одной дисциплины.

3. Вывести список преподавателей с персональными данными и окладами, которые преподают на заданном курсе.

Вариант 8. В расписании ежедневных рейсовых автобусов между двумя городами (Ачинск и Канск, время в пути на автобусе – 1 час) указаны: номер маршрута, время отправления (8:20 можно рассматривать как 8+20/60), город прибытия, водитель. Водители, обслуживающие маршруты, работают в одной из трех компаний (ТрансАчинск, ТранспортКанск, КАчинские), первая компания зарегистрирована в Ачинске, две других – в Канске. Каждый водитель имеет свою категорию (1, 2, 3), которая характеризует его профессиональные качества, а также стаж работы. Зарегистрированные компании в городе Ачинске премию сотрудникам начисляют в случае обслуживания рейса после 18 часов. А канские компании начисляют премию, если у водителя 1 категория или 2 категория и стаж больше 15 лет.

Создать программу на языке Пролог, которая будет отвечать на следующие вопросы:

1. Вывести список маршрутов, которые обслуживаются водителями заданной категории и стаж которых больше указанного значения.

2. Создать список маршрутов по парам, для которых промежутки времени между прибытием автобуса одного маршрута в один из городов и отправлением другого маршрута из этого города находится в интервале от 5 до 15 минут.

3. Вывести список водителей, которым положена премия.

Вариант 9. Ежедневно участковые терапевты обслуживают от 10 до 15 вызовов. Информация по каждому вызову фиксируется терапевтом в электронном журнале: номер, терапевт, пациент, время вызова, время прихода к пациенту, диагноз, степень тяжести (легкая, средняя, тяжёлая). В электронном журнале у каждого пациента есть карта с персональными данными, у каждого терапевта есть специальный паспорт, в котором указаны категория (1, 2, 3), стаж работы и другие профессиональные данные врача.

Создать программу на языке Пролог, которая будет отвечать на следующие вопросы:

1. Составить список пациентов с тяжелой формой болезни, которые старше 60 лет или у которых диагноз «грипп».

2. Вывести список вызовов (включить профессиональные данные обслужившего терапевта), для которых временной интервал между вызовом и приходом врача больше заданного количества минут.

3. Вывести список терапевтов заданной категории младше 40 лет, которые обслуживали детей в возрасте от 12 до 15 лет с диагнозом «ОРВИ» со средней или тяжелой формой заболевания.

Вариант 10. В сети Internet опубликована электронная доска объявлений по операциям с недвижимостью (только квартиры). Используются два вида объявлений: предложение и объявление-спрос. Предложение оформляется по следующему формату: порядковый номер, идентификатор автора, тип предложения (аренда, продажа), улица, этаж, цена. Каждая улица находится в определенном районе.

Объявление-спрос оформляется по следующему образцу: порядковый номер, идентификатор автора, тип предложения (аренда, покупка), район, мин. этаж, макс. этаж, макс. цена). По каждому объявлению хранится закрытая информация об авторе: идентификатор автора, фамилия, электронный адрес. Создать программу на языке Пролог, которая будет отвечать на следующие вопросы:

1. Составить список всех объявлений заданного автора.
2. Вывести список всех объявлений спроса с электронным адресом автора, приемлемых для предложения, которое задается номером.
3. Вывести список всех объявлений предложения, приемлемых для заданного объявления-спроса (задается номер спроса).

Вариант 11. Регистрируясь на электронной бирже труда, каждый пользователь (физическое или юридическое лицо) заполняет о себе следующие данные: тип пользователя (физическое или юридическое лицо), телефон, фамилия для контакта или название организации. Для юридического лица указывается название организации, для физического лица – фамилия человека для контакта. Все юридические лица дополнительно заполняют форму с описанием вида деятельности организации: вид деятельности (коммерческая, муниципальная или государственная структура).

Зарегистрированный пользователь может выставить объявление. Биржа труда принимает объявления двух видов: вакансии, поиск работы. Каждая вакансия характеризуется следующими параметрами: название организации, должность, оклад. Каждое объявление по поиску работы оформляется со следующими параметрами: должность, вид деятельности, минимальная заработная плата, фамилия человека для контакта.

Создать программу на языке Пролог, которая будет отвечать на следующие вопросы:

1. Вывести все объявления от муниципальных или государственных структур.

2. Вывести список всех объявлений о вакансиях с указанием контактных телефонов для заданного объявления по поиску работы (объявление по поиску задается фамилией человека).

3. Вывести список всех объявлений по поиску работы с указанием всех характеристик (в том числе и контактного телефона) для заданного объявления о вакансии.

Вариант 12. На городской телефонной станции каждый телефонный вызов фиксируется следующими параметрами: номер вызывающего абонента, номер вызываемого абонента, время вызова (дробное число от 0 до 24, например, если вызов был в 15:10, то это значение будет равно $15+10/60$), время разговора в секундах. Информация о вызовах накапливается за сутки, а затем архивируется.

Все телефонные номера абонентов зарегистрированы за определенным лицом: фамилия, вид тарифа. Существует три вида тарифов:

а) экономный: вызов до 10 сек по цене 0 усл.ед., каждая следующая секунда 0.03 усл.ед.

б) псевдо-безлимитный: вызов до 1 мин. по цене 0.05 усл.ед. за 1 сек., каждая следующая секунда по цене 0.01 усл.ед.

в) советский: время вызова округляется до минуты (в большую сторону), каждая минута оплачивается по цене 1 усл.ед.

Если телефон вызывающего абонента зарегистрирован как льготный, то на «советский» вид тарифа абонент получает 50% скидку.

Создать программу на языке Пролог, которая будет отвечать на следующие вопросы:

1. Вывести список вызовов на заданную фамилию (вызывающего абонента) с указанием стоимости вызова и фамилии вызываемого абонента.

2. Вывести список льготных вызовов с (указанием фамилии вызывающего абонента и стоимости вызова) для заданной фамилии вызываемого абонента.

3. Вывести список вызовов с 14 до 15 часов длительностью менее 5 секунд, вызывающий абонент которых выбрал псевдо-безлимитный или экономный виды тарифов.

Вариант 13. В течение дня всё движение городского транспорта фиксируется на электронном табло у диспетчера (а потом архивируется). На пульт к дежурному поступает следующая информация: номер маршрута, водитель, время отправления, время прибытия, количество нарушений ПДД. Для всех номеров маршрутов есть специальный регламент по времени, определяющий минимальное и максимальное время прохождения маршрута, который, в общем случае, может зави-

сеть от времени дня. Например, до обеда одни значения, регламентирующие время прохождения маршрута, а после обеда – другие.

Все водители работают либо в компании ГорТранс, либо в ТрансГор. Если водитель при прохождении маршрута нарушил правила или более одного раза нарушил регламент по времени, то он заносится в «черный список».

Создать программу на языке Пролог, которая будет отвечать на следующие вопросы:

1. Вывести черный список водителей.
2. Вывести всю информацию о выполненных маршрутах водителями заданной транспортной компании, указать было/не было нарушений по регламенту.
3. Для заданной компании вывести список водителей, которые вышли на маршрут, имея на выполняемом ранее маршруте более одного нарушения ПДД.

Вариант 14. В муниципальной аптеке хранится вся необходимая информация о медикаментах: название препарата, кол-во поступивших ед., количество проданных ед., фирма-провизор. Аптека работает с несколькими фирмами-провизорами (Фарм, Алфит, БийскВитамины, ...). По каждой фирме-провизоре хранится их контактная информация: контактное лицо, телефон.

Как только врач выписывает льготный рецепт (с указанием: фамилия пациента, диагноз, название препарата, кол-во ед.), вся информация по рецепту поступает в аптеку. Создать программу на языке Пролог, которая будет отвечать на следующие вопросы:

1. Для заданного препарата вывести полную информацию о наличии и о провизоре.
2. Для заданного провизора вывести список препаратов, которые закончились или заканчиваются (менее 10 ед.), но рецептов на которые пока нет.
3. Вывести список препаратов, которых нет, или есть в недостаточном количестве для выдачи пациентам по рецептам.

Вариант 15. Школьное расписание определяет время проведения каждого учебного занятия (номер урока от 1 до 5) и день недели (от 1 до 6), название предмета, класс (кл_1_А, кл_2_В и т.д.), учителя (который будет проводить занятие). Дополнительно по каждому классу хранится информация о классном руководителе и по количеству учеников в классе. Каждый учитель характеризуется педагогическим стажем и категорией (вторая, первая, высшая).

Создать программу на языке Пролог, которая будет отвечать на следующие вопросы:

1. Вывести список занятий, по которым в расписании есть накладки: у одного учителя одновременно два разных занятия, у одного класса одновременно два разных занятия.

2. Вывести список классов, у которых не ведут занятия классные руководители.

3. Вывести список занятий, проводимых на последнем уроке (6 урок), учителями высшей категории с педагогическим стажем более 10 лет для классов, где количество учеников меньше заданного числа.

Вариант 16. Все игры спортивного клуба по хоккею фиксируются в базе данных: первая команда, вторая команда, количество забитых шайб первой командой, количество забитых шайб второй командой, дата проведения матча. Команды клуба выступают в трёх дивизионах (1, 2, 3), у каждой команды есть главный тренер.

Создать программу на языке Пролог, которая будет отвечать на следующие вопросы:

1. Вывести список всех игр заданного месяца.

2. Вывести список команд первого дивизиона, у которых есть проигранные матчи.

3. Вывести список игр, в которых выступали подопечные заданного главного тренера, и которые закончились ничьей.

2. Работа со списками

В данную лабораторную работу включены задания на выполнение различных операций со списком элементов (термов). В качестве элементов списка могут быть атомы, строки (произвольный набор символов в двойных кавычках) или числа (целые или вещественные). Элементы списка могут быть получены с использованием функций: 1) консольного ввода-вывода или 2) работы с текстовыми файлами.

Вариант 1. Вывести на экран элементы списка (строки), в символьной записи которых более одного слова (последовательность символов не пробелов до или после символа пробел).

Вариант 2. Для заданного списка строк построить новый список термов – целых чисел по следующему принципу: на i -том месте нового списка расположено число – длина i -той строки исходного списка.

Вариант 3. Для заданного списка строк построить новый список, элементы которого расположены в зеркальном отражении относительно исходного списка, например для списка $[a, b, c, d]$ нужно получить $[d, c, b, a]$.

Вариант 4. Для двух заданных списков чисел получить третий список путём слияния первых двух списков по следующему принципу: из первого списка в новый третий список попадают элементы строго

больше заданного значения, из второго списка – те элементы, которые равны или меньше заданного значения.

Вариант 5. Из заданного списка строк сформировать новый список (не нарушая последовательности элементов), в который следует включить только те строки из данного списка, где первым символом является буква. В формируемом списке строки должны начинаться с заглавной буквы, если в их символьной записи нет символов арифметических операций (+, -, *).

Вариант 6. Для заданного списка чисел сформировать новый список отклонений от среднего арифметического элементов списка. Например, для списка [10, 12, 40, -24] среднее арифметическое равно: $(10+12+40+-24)/4=9.5$, элементы нового списка вычисляются следующим образом: [9.5-10/, 9.5-12/, 9.5-40/, 9.5- -24/].

Вариант 7. Для заданного списка строк построить новый список, не нарушая порядка по следующему принципу: если в строке исходного списка есть символ равно (“=”), то строка для нового списка должна быть построена сцеплением двух подстрок после первого знака равно и до первого знака равно. Например, для следующей строки заданного списка: “*abc=cd-e*” в новом списке будет соответствовать строка “*cd-eabc*”. Если в строке заданного списка нет символа равно, то в новой список данная строка перейдет без исключения.

Вариант 8. Для заданного массива целых чисел сформировать два списка (не нарушая порядка), в первый список включить чётные числа заданного массива, во второй – нечётные числа.

Вариант 9. Для заданного списка чисел сформировать два новых списка, в первый включить целые числа (дробная часть отсутствует), во второй – все остальные числа.

Вариант 10. Для заданного списка строк построить новый список по следующему принципу: в новом списке вначале расположены строки, в символьной записи которых есть символ пробел, затем все остальные. Порядок строк, в символьной записи которых есть символ пробел, а также остальных строк в новом списке не должен измениться.

Вариант 11. Для заданной строки символов создать список строк, в котором каждая строка должна состоять из одного символа заданной строки (порядок символов в строке должен соответствовать порядку элементов списка, составленных из символов строки).

Вариант 12. Для заданного списка строк-слов сформировать строку-предложение, в которую записаны строки-слова списка, разделенные пробелами. Из списка строк-слов в строку-предложение включить только те элементы, в символьной записи которых нет пробелов.

Вариант 13. Для списка строк сформировать массив односимвольных строк. Если первым символом строки списка является буква, тогда эта буква будет соответствующим элементом массива, если первый символ строки – не буква, тогда элементом массива будет строка, состоящая из символа подчеркивания.

Вариант 14. Для списка строк, в котором хранятся словосочетания английского языка (два слова, принадлежащих разным частям речи, разделенных пробелами), сформировать два массива строк. В первый массив включать слова из списка строк, у которых меньше длина. Во второй массив включать оставшиеся слова из заданного списка строк. Из каждой строки списка одно слово должно попасть в первый массив (короткое), другое слово – во второй массив.

Вариант 15. Заданы два списка целых чисел с равным количеством элементов. Построить третий список вещественных чисел, в котором каждый элемент равен арифметическому среднему значению двух чисел, расположенных на соответствующих позициях в заданных списках.

Вариант 16. Заданы два списка строк с произвольным количеством элементов. Построить третий список строк из элементов первого списка по следующему правилу: элемент первого списка следует включить в новый список (в третий), если данная строка является подстрокой для любого элемента из второго списка строк.

3. Структуры данных

Текст задания II. В БД деканата содержится следующая информация о кафедрах факультета:

Название кафедры;

Список преподавателей (первый преподаватель в списке – заведующий кафедрой);

Список учебных курсов, закрепленных за кафедрой.

Каждый преподаватель описывается: ФИО, дата рождения, ученая степень, количество публикаций, стаж работы.

Каждый учебный предмет характеризуется: название, курс (1-6), учебные часы, тип (региональный, федеральный).

Вариант 1. Построить правило, которое выводит на экран список следующего содержания: название кафедры, заведующий, количество докторов наук, суммарное количество публикаций, среднее количество учебных часов выделяемых на предмет.

Вариант 2. Построить правило, которое для заданного курса (1-6) формирует и выводит список следующего содержания: название кафедры, за которой закреплен предмет, преподаваемый на данном кур-

се, средний возраст преподавателей кафедры, суммарное количество учебных часов по кафедре.

Текст задания III. В Алтайском лесничестве каждая природная зона характеризуется:

Кадастровым номером;

Типом природной зоны (рекреационный, закрытый, общий);

Названием района;

Списком земельных угодий (единиц);

Списком животных, обитающих в природной зоне.

Каждая земельная единица характеризуется: тип почвы (чернозем, каштановые, песчаник т.д.), средняя толщина плодородного слоя, площадь земельной единицы.

Каждое животное описывается: названием, количеством особей.

Вариант 3. Построить правило, которое выводит на экран список следующего содержания: кадастровый номер природной зоны, тип природной зоны, усредненная толщина природного слоя по всем земельным единицам, суммарное количество животных, суммарная площадь черноземных почв.

Вариант 4. Для заданного названия животного вывести список природных зон, в которых это животное обитает, со следующей информацией: количество земельных единиц в природной зоне, среднее количество особей животных по всем видам.

Текст задания IV. Торговый склад по обслуживанию сети магазинов хранит в БД следующую информацию о заказах:

Номер заказа;

Информация о поступлении заказа: принимающий менеджер, дата поступления;

Информация о выполнении заказа: отправляющий менеджер, дата отгрузки;

Список товаров первой категории,

Список товаров второй категории

Идентификатор магазина (откуда заказ).

Каждый товар содержит следующую информацию: название, цена за ед., количество единиц, тип единиц (штуки, упаковки и др.).

Кроме информации о заказах в базе хранятся данные о менеджерах (ФИО и контактная информация) и о магазинах, с которыми работает торговый склад: идентификатор магазина, адрес.

Вариант 5. Построить правило, которое выводит на экран список следующего содержания: номер заказа, дата поступления, дата отгрузки, контактная информация отправляющего менеджера, стоимость

товаров первой категории, стоимость товаров второй категории, общее количество товаров.

Вариант 6. Построить правило, которое для заданного названия товара из второй категории выводит на экран следующую информацию: номер заказа, в котором этот товар содержится, цену за единицу, количество ед., тип единиц, ФИО и контактная информация принимающего заказ менеджера.

Текст задания V. В базе данных лаборатории хранится следующая информация об исследуемых водоемах:

Название водоема;

Тип (озеро, болото, пруд и т.д.);

Координаты водоема;

Список проб воды;

Список биологических организмов, обитаемых в водоеме.

Каждая проба воды описывается: дата взятия пробы, процент содержания тяжелых металлов.

Каждый биологический организм характеризуется: название, количество в 1 литре воды, степень вредности по шкале от 1 до 5.

Координаты водоема определяют физическое расположение объекта на карте: долгота и широта самой северной точки водоема.

Вариант 7. Построить правило, которое выводит на экран список следующего содержания: название водоёма, тип, количество биологических организмов в 1 л., средний процент содержания тяжелых металлов по всем пробам.

Вариант 8. Для заданного биологического организма вывести список водоемов (название, координаты), в которых он содержится, с описанием самой последней по дате пробы воды.

Текст задания VI. В базе данных ГАИ города ежедневно собирается следующая информация о транспортных развязках:

Идентификационный номер;

Тип (федеральный, региональный, городской);

Номер района города;

Список автодорожных происшествий;

Список проезжающих транспортных средств.

Каждое автодорожное происшествие характеризуется: номер протокола, время данного события, количество пострадавших.

Каждое транспортное средство о себе оставляет следующую информацию: госномер, тип транспортного средства (легковой автомобиль, автобус, грузовик), время проезда.

По каждому району города хранится информация: номер района, название, адрес районной инспекции.

Вариант 9. Построить правило, которое выводит на экран список следующего содержания: идентификационный номер транспортной развязки, название района и адрес районной инспекции, количество автодорожных происшествий, суммарное количество пострадавших, суммарное количество проехавших автобусов.

Вариант 10. Построить правило, которое для заданного регистрационного номера автомобиля выводит следующий список: идентификационный номер транспортной развязки, по которой проезжало транспортное средство, среднее количество транспортных средств по трем видам проехавших по этой развязке, среднее количество пострадавших на этой развязке.

Текст задания VII. В базе данных городской дежурной части ежесуточно собирается следующая информация об уличных происшествиях в районах города:

Название района;

Идентификатор дежурного сотрудника;

Номенклатурный номер звуковой записи;

Список разбоев и грабежей;

Список убийств.

Каждый дежурный сотрудник характеризуется следующими данными: идентификатор, ФИО, звание, телефон для связи.

Все телефонные разговоры записываются в специальный файл, который оформляется в базе следующими параметрами: номенклатурный номер записи, название файл-сервера, список звуковых дорожек (например: [109a, 109f, 110a]).

Каждый разбой или грабёж оформляется по следующему шаблону: рег. номер, информация о пострадавшем: (ФИО, контактный телефон), время, предварительная сумма ущерба. Фамилии, имена, отчества пострадавших могут повторяться.

Каждое убийство оформляется по шаблону: рег. номер, причина смерти (удушьё, потеря крови и т.д.), время.

Вариант 11. Построить правило, которое выводит на экран список следующего содержания: название района, ФИО дежурного сотрудника, список звуковых дорожек, общая сумма предварительного ущерба по грабежам и разбоям, количество убийств, где причина смерти «удушьё».

Вариант 12. Построить правило, которое для заданной фамилии пострадавшего строит список следующего содержания:

- 1) название района, в списке грабежей и разбоев которого фигурирует заданная фамилия;
- 2) идентификатор дежурного сотрудника из района;

- 3) количество звуковых дорожек в файле, который хранит звуковую информацию о районе;
- 4) количество убийств и разбоев в данном районе.

Текст задания VIII. В коммерческой образовательной организации, занимающейся обучением персонала, хранится следующая информация по проведенным курсам:

Порядковый номер курса;

Номер специализации;

Дата начала занятий;

Дата окончания занятий;

Название организации (для персонала, которой организован курс);

Список слушателей;

Список изучаемых предметов.

Для каждой специализации дополнительно хранится следующая информация: номер, название, количество учебных часов.

Каждый слушатель из списка характеризуется: ФИО, возраст, занимаемая должность (экономист, бухгалтер).

Каждый изучаемый предмет описывается следующими параметрами: название, ФИО преподавателя, количество учебных часов, тариф оплаты за учебный час.

Вариант 13. Построить правило, которое выводит на экран список следующего содержания: порядковый номер курса, продолжительность курса в днях, название специализации, название организации, количество слушателей в должности экономиста, общая сумма оплаты за курс.

Вариант 14. Построить правило, которое для заданного названия предмета, выводит список следующего содержания: порядковый номер курса, в изучении которого есть данный предмет, количество учебных часов, отводимых на данную специализацию, количество слушателей, количество изучаемых предметов.

Текст задания IX. Диспетчер районного отделения скорой помощи ежедневно собирает в базу данных следующую информацию по работе отделения:

Порядковый номер суток по книге учета;

Идентификатор дежурного;

Дата;

Список больных;

Список дежурных бригад.

Каждый дежурный характеризуется: идентификатор, ФИО, должность, телефон для связи.

По каждому больному из списка фиксируется следующая информация: ФИО, возраст, предварительный диагноз, адрес.

Каждая дежурная бригада оформляется по следующему шаблону: госномер скорой машины, врач: ФИО, должность, ФИО медсестры/медбрата.

Вариант 15. Построить правило, которое выводит на экран список следующего содержания: дата, на которую собрана информация, порядковый номер суток по книге учета, ФИО и должность дежурного, количество больных с предварительным диагнозом «грипп», средний возраст больных в эти сутки.

Вариант 16. Построить правило, которое для заданного предварительного диагноза выводит список следующего содержания: порядковый номер суток по книге учета, в которые зафиксирован данный предварительный диагноз, телефон для связи работающего в эти сутки дежурного, дата, количество дежурных бригад, где врач педиатр.

Интерпретатор языка Common Lisp

4. Списки, функции, управляющие структуры

При выполнении заданий нельзя использовать встроенные функции стандарта Common Lisp, функционал которых аналогичен требованиям задания.

Вариант 1. Определите функцию, возвращающую сумму первого и последнего элементов списка вещественных чисел.

Вариант 2. Определите функцию, удаляющую из списка каждый второй элемент.

Вариант 3. Определите функцию-предикат, которая проверяет, является ли её аргумент многоуровневым списком.

Вариант 4. Определить функцию, которая для двух одинаково упорядоченных списков вещественных чисел выполняет их слияние, не нарушая порядка элементов. (Сортировку выполнять нельзя).

Вариант 5. Определить функцию, которая для числовой выборки вещественных чисел, задаваемой одноуровневым списком, вычисляет максимальный и минимальный элемент.

Вариант 6. Определить функцию, которая для одноуровневого списка строк формирует итоговую строку сцеплением всех элементов списка. Элементы в итоговой строке разделяются знаком: двоеточие.

Вариант 7. Определите функцию, которая сливает два заданных списка в один список. Пример:

```
_(function ' (a (b a) () c) ' ( (a) 34 g (a vv) 89)
```

Результат:

```
' (a (a) (b a) 34 () g c (a vv) 89)
```

Вариант 8. Определить функцию, которая для заданного списка формирует новый список по правилу: в новый список включены только те элементы, которые сами являются списками. Пример:

```
_(function ' (a (b a) () c)
```

Результат:

```
' ( (b a) () )
```

Вариант 9. Определите функцию, которая преобразует список $(a\ b\ c\ d\ e\ f\ g)$, меняя уровень элементов на нечётных позициях: $((a)\ b\ (c)\ d\ (e)\ f\ (g))$.

Вариант 10. Определить функцию для поиска *НОД* двух положительных целых чисел.

Вариант 11. Определите функцию, удаляющую из списка каждый второй элемент (чётная позиция), который является списком. Если на чётной позиции списка расположен атомарный объект, его следует оставить.

Вариант 12. Определить функцию, которая для двух числовых выборок, задаваемых двумя списками, определяет их коэффициент корреляции.

Вариант 13. Определить функцию, которая для числовой выборки, задаваемой списком, определяет дисперсию.

Вариант 14. Определить функцию с двумя аргументами: *список строк, подстрока*. Данная функция должна для заданного списка строк формировать новый список, в котором на соответствующей позиции должно быть целое число – количество вхождений указанной подстроки в соответствующую строку списка.

Библиографический список

1. Иванов Д.А. Функциональное программирование и не только. – М., 2016.
2. Грэм П. ANSI Common Lisp. – М.: Символ-Плюс, 2012.
3. Цуканова Н.И., Дмитриева Т.А. Логическое программирование на языке Visual Prolog. – М.: Горячая линия – Телеком, 2008.
4. Цуканова Н.И., Дмитриева Т.А. Теория и практика логического программирования на языке Visual Prolog 7: учеб. пособие. – М.: Горячая линия – Телеком, 2013.
5. Шрайнер П.А. Основы программирования на языке Пролог. – М., 2005.