



В. Н. Марков

Современное логическое программирование на языке Visual Prolog 7.5



Материалы
на www.bhv.ru



УДК 004.438 Visual Prolog
ББК 32.973.26-018.1
М26

Марков В. Н.

М26 Современное логическое программирование на языке Visual Prolog 7.5:
учебник. — СПб.: БХВ-Петербург, 2016. — 544 с.: ил. —
(Учебная литература для вузов)

ISBN 978-5-9775-3487-1

В учебнике излагается полный набор классических и новейших инструментов логического программирования, а также парадигмы функционального, обобщенного, императивного и объектно-ориентированного программирования, органически вошедшие в Visual Prolog 7.5. Рассматриваются основные способы представления и обработки графов, деревьев и массивов, инструменты профессионального программирования. Приводятся примеры разработки символьных преобразователей, калькуляторов, интерпретаторов языков программирования, игровых моделей и т. п. Книга содержит практикум по программированию и описание основных классов Visual Prolog. Учебник предназначен для изучения дисциплин «Логическое программирование» и «Функциональное и логическое программирование».

*Для студентов, преподавателей
и разработчиков интеллектуальных информационных систем.*

УДК 004.438 Visual Prolog
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Евгений Рыбаков</i>
Зав. редакцией	<i>Екатерина Капалыгина</i>
Редактор	<i>Григорий Добин</i>
Компьютерная верстка	<i>Ольги Сергиенко</i>
Корректор	<i>Зинаида Дмитриева</i>
Дизайн серии	<i>Инны Тачиной</i>
Оформление обложки	<i>Марины Дамбиевой</i>
Фото	<i>Кирилла Сергеева</i>

Подписано в печать 30.09.15.
Формат 70×100¹/₁₆. Печать офсетная. Усл. печ. л. 43,86.
Тираж 700 экз. Заказ №
"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.
Первая Академическая типография "Наука"
199034, Санкт-Петербург, 9 линия, 12/28

ISBN 978-5-9775-3487-1

© Марков В. Н., 2016
© Оформление, издательство "БХВ-Петербург", 2016

Оглавление

Предисловие	13
О содержании	14
Благодарности	15
О языке	16
О приоритетах	17
Поддержка	17
ЧАСТЬ I. ОСНОВНЫЕ ЭЛЕМЕНТЫ ЯЗЫКА VISUAL PROLOG	19
Глава 1. Лексика языка Visual Prolog	21
1.1. Алфавит	21
1.2. Комментарии	21
1.3. Символы разметки текста	22
1.4. Лексемы	22
1.4.1. Ключевые слова	22
1.4.2. Знаки пунктуации	22
1.4.3. Операции	23
1.4.4. Идентификаторы	23
1.4.5. Литералы	23
Целые числа	24
Вещественные числа	24
Символы	25
Строки	25
Двоичные данные	26
Списки	27
Составной терм	27
Глава 2. Термы	28
2.1. Понятие термов	28
2.2. Переменные	29
2.3. Списки	30
2.4. Операция сопоставления с образцом	31
2.4.1. Сопоставление термов	32
2.4.2. Сопоставление списков	33

2.5. Операция унификации.....	34
2.6. Операция «должно унифицироваться»	36
2.7. Неразрушающее присваивание в Visual Prolog.....	37
Глава 3. Операции языка	38
3.1. Арифметические операции и операции сравнения и сопоставления	38
3.2. Целочисленное деление	39
Глава 4. Типы, домены, подтипы	40
4.1. Типы данных	40
4.2. Встроенные домены	41
4.3. Домены-синонимы.....	45
4.4. Подтипы и категориальный полиморфизм.....	45
4.5. Домены, определяемые пользователем	46
4.5.1. Целочисленные подтипы	46
4.5.2. Вещественные подтипы	47
4.5.3. Списочные домены.....	48
4.5.4. Составные домены.....	48
Глава 5. Константы.....	50
5.1. Объявление пользовательских констант.....	50
5.2. Встроенные константы.....	51
ЧАСТЬ II. ЯЗЫК VISUAL PROLOG.....	53
Глава 6. Предикаты	55
6.1. Понятие предиката	55
6.2. Логические операции над предикатами.....	56
6.3. Логические формулы.....	58
6.4. Правила вывода предикатов	59
6.4.1. Вывод предиката из другого предиката.....	59
6.4.2. Формулы вычисления предикатов	61
6.5. Факты.....	64
6.6. Машина логического вывода.....	65
6.7. Работа механизма поиска с возвратом.....	66
6.8. Чистый Пролог.....	74
Глава 7. Предикаты в Visual Prolog.....	75
7.1. Объявление и определение предикатов	75
7.1.1. Режимы детерминизма.....	75
7.1.2. Шаблон потоков	77
7.1.3. Определение предикатов.....	78
7.2. Объявление и определение функций.....	80
7.3. Объявление и определение фактов	82
7.3.1. Режимы детерминизма факта	82
7.3.2. Определение фактов.....	83
7.3.3. Факты-переменные.....	83
7.4. Операции над разделами базы фактов	87
7.5. Операции над фактами.....	88

7.6. Встроенные предикаты Visual Prolog	89
7.6.1. Предикаты для работы с фактами базы данных.....	89
7.6.2. Предикаты контроля потока параметров.....	90
7.6.3. Предикаты локализации места выполнения программы в исходном тексте.....	91
7.6.4. Предикаты контроля компиляции исходного текста.....	91
7.6.5. Предикат сравнения термов.....	92
7.6.6. Предикаты преобразования типов.....	92
7.6.7. Предикаты обработки эллипсиса	96
7.6.8. Предикаты получения размера домена	97
7.6.9. Предикат получения размера термина	98
7.6.10. Предикаты объявления/проверки домена переменной.....	98
7.6.11. Предикат обработки ошибки	99
7.6.12. Предикаты управления выполнением программы.....	99
Глава 8. Модули.....	101
8.1. Область видимости	102
8.2. Структура модуля	102
8.3. Использование модулей в проекте	104
8.3.1. Библиотека констант и доменов	104
8.3.2. Библиотеки предикатов.....	105
8.3.3. Глобальные переменные проекта.....	105
Глава 9. Отсечение и отрицание	108
9.1. Принцип работы отсечения	108
9.1.1. Область видимости отсечения.....	109
9.1.2. Использование отсечений.....	109
9.2. Зеленые и красные отсечения.....	111
9.3. Динамическое отсечение	112
9.4. Отрицание	113
Глава 10. Циклы с откатом	115
10.1. Структура цикла с откатом	115
10.2. Реализация циклов с откатом	117
10.3. Использование изменяемых переменных в циклах с откатом.....	122
10.4. Циклы с откатом на основе отрицания.....	125
Глава 11. Рекурсия.....	127
11.1. Структура рекурсии.....	127
11.2. Реализация рекурсии	130
11.3. Мемоизация.....	138
Глава 12. Ввод/вывод	140
12.1. Ввод/вывод в консольном приложении.....	140
12.1.1. Основные функции ввода с клавиатуры	140
12.1.2. Основные предикаты вывода на экран	141
12.1.3. Использование предиката <i>hasDomain</i>	142
12.2. Файловый ввод/вывод	143
12.3. Поточковый ввод/вывод.....	144
12.3.1. Поточковый ввод/вывод в цикле с откатом.....	146
12.3.2. Поточковый ввод/вывод в рекурсивном цикле	147

12.4. Строковые потоки.....	148
2.4.1. Поток ввода строк.....	149
12.4.2. Поток вывода строк.....	151

Глава 13. Списки 153

13.1. Представление списков в памяти компьютера.....	153
13.2. Встроенные операции над списками.....	154
13.2.1. Объявление списочных доменов.....	157
13.3. Реализация очереди и дека.....	157
13.4. Принципы рекурсивной обработки списков.....	158
13.4.1. Построение списков из элементов.....	158
13.4.2. Построение реверсивных списков из элементов.....	159
13.4.3. Сканирование списка.....	160
13.4.4. Модификация списка.....	161
13.4.5. Синхронная обработка списков.....	161
13.5. Примеры рекурсивной обработки списков.....	162
13.5.1. Определение длины списка.....	162
13.5.2. Построение списка.....	164
13.5.3. Соединение списков.....	165
13.5.4. Реверс списка.....	167
13.6. Ввод/вывод списков целиком.....	167
13.6.1. Терминальный ввод/вывод списков.....	167
13.6.2. Файловый ввод/вывод списков.....	168
13.6.3. Поточковый ввод/вывод списков.....	168
13.7. Поэлементный ввод/вывод списков.....	169
13.7.1. Поэлементный ввод/вывод списков в цикле с откатом.....	169
13.7.2. Поэлементный ввод/вывод списков в рекурсивном цикле.....	170
13.8. Предикат выборки элементов списка.....	171
13.9. Коллектор списков.....	172
13.10. Представление базы фактов списками фактов.....	174
13.10.1. Когерентность базы фактов и списка фактов.....	174
13.10.2. Домен фактов внутренней базы данных.....	175
13.10.3. Создание списка фактов внутренней базы данных.....	175
13.10.4. Восстановление внутренней базы данных из списка фактов.....	175
13.10.5. Предикаты преобразования внутренней базы данных в список фактов и обратно.....	176

Глава 14. Параметрический полиморфизм..... 178

14.1. Полиморфизм параметров предикатов.....	178
14.2. Полиморфизм параметров доменов.....	179
14.2.1. Полиморфный списочный домен класса <i>core</i>	182

Глава 15. Эллипсис 183

Глава 16. Предикаты второго порядка и анонимные предикаты..... 185

16.1. Предикатные и функциональные домены.....	185
16.2. Анонимные предикаты и функции.....	189
16.2.1. Определения анонимных предикатов.....	190

16.2.2. Использование анонимных предикатов и функций	190
16.2.3. Замыкание	195
16.2.4. Каррирование	195
16.3. Высокоуровневые предикаты и функции класса <i>list</i>	197

Глава 17. Императивные конструкции 205

17.1. Разрушающее присваивание	205
17.2. Ветвление	206
17.3. Условные выражения	207
17.4. Цикл <i>foreach</i>	207
17.5. Циклы с заданным числом повторений	210

Глава 18. Обработка исключительных ситуаций..... 211

18.1. Явный вызов исключений	211
18.2. Обработка исключений	213

Глава 19. Классы 218

19.1. Структура класса	218
19.1.1. Параметры состояния класса и объекта	221
19.2. Объекты	221
19.2.1. Создание объектов	221
19.2.2. Объектные предикаты	222
19.2.3. Объектные свойства	223
19.2.4. Удаление объектов	225
19.3. Классы	226
19.3.1. Конструкторы	226
19.3.2. Состояние класса	227
19.4. Наследование кода и поддержка интерфейсов	228
19.5. Сохранение объектов	230
19.6. Операции над всеми живущими объектами	231
19.7. Примеры использования классов	232

Глава 20. Обобщенное программирование 239

20.1. Обобщенные интерфейсы	239
20.2. Обобщенные классы	240
20.3. Обобщенные реализации	241
20.4. Пример обобщенной очереди	242

ЧАСТЬ III. СРЕДСТВА

ПРОФЕССИОНАЛЬНОГО ПРОГРАММИРОВАНИЯ 245

Глава 21. Многопоточность 247

21.1. Основные операции с потоками	247
21.1.1. Создание потоков	247
21.1.2. Завершение потоков	248
21.1.3. Приостановка и возобновление потоков	248
21.1.4. Примеры создания потока	249
21.2. Мониторы	251
21.3. Защита	253

Глава 22. Доступ к API-функциям Windows	256
22.1. Описание типов данных API-функций доменами Visual Prolog.....	256
22.2. Объявления предикатов для вызова API-функций.....	258
22.3. Использование пакета <i>windowsAPI</i>	260
22.4. Использование API-функций из библиотеки Windows.....	262
Глава 23. Разработка и использование DLL	265
23.1. Создание DLL-проекта.....	266
23.2. Описание экспортируемых предикатов и функций DLL-проекта.....	267
23.3. Использование DLL-проекта.....	268
Глава 24. Отладка приложений	271
24.1. Отладчик Visual Prolog.....	271
24.2. Просмотр значений переменных средствами языка.....	274
24.3. Контроль стека и кучи.....	275
ЧАСТЬ IV. ПРЕДСТАВЛЕНИЕ И ОБРАБОТКА ДАННЫХ В VISUAL PROLOG	277
Глава 25. Графы	279
25.1. Представление ориентированных графов.....	279
25.2. Представление неориентированных графов.....	282
25.3. Поиск «сначала вглубь».....	283
25.4. Поиск «сначала вширь».....	285
Глава 26. Деревья	288
26.1. Представление деревьев в Visual Prolog.....	288
26.2. Операции над деревьями.....	290
26.2.1. Добавление вершины.....	290
26.2.2. Поиск вершины.....	291
26.2.3. Удаление вершины.....	291
26.3. Красно-черные деревья.....	294
Глава 27. Массивы	299
27.1. Класс <i>binary</i>	299
27.1.1. Создание массивов <i>binary</i>	299
27.1.2. Основные операции над массивами <i>binary</i>	300
27.2. Класс <i>arrayM</i>	301
27.2.1. Создание одномерных массивов.....	301
27.2.2. Основные операции над одномерными массивами.....	301
27.3. Класс <i>array2M</i>	303
27.3.1. Создание двумерных массивов.....	303
27.3.2. Основные операции над двумерными массивами.....	303
27.4. Класс <i>arrayM_boolean</i>	304
27.4.1. Создание одномерных булевых массивов.....	304
27.4.2. Основные операции над одномерными булевыми массивами.....	304
27.5. Способы обработки массивов.....	305

Глава 28. Символьные преобразования	307
28.1. Этапы анализа текстов	307
28.2. Основы анализа текстов на Visual Prolog	308
28.2.1. Простой лексический анализ	308
28.2.2. Простой синтаксический анализ	308
28.3. Анализ математических выражений	309
28.4. Парсер математических выражений с произвольной грамматикой	315
28.5. Символьное дифференцирование выражений	322
28.6. Калькулятор	329
Способ 1	329
Способ 2	330
Способ 3	331
28.7. Задания для самостоятельного решения	333
Глава 29. Интерпретатор программ	335
29.1. Лексический анализ	335
29.2. Синтаксический анализ	341
29.3. Интерпретатор программ	346
29.4. Задания для самостоятельного решения	353
Глава 30. Практические рекомендации	355
30.1. Выбор способа представления и обработки данных	355
30.2. Управление памятью в Visual Prolog 7	356
30.2.1. Стек вызовов	356
Управляемый детерминизм	357
Задание произвольного размера стека вызовов	361
30.2.2. Динамическая память	363
Отказ от глобального стека в Visual Prolog 7	364
Многопоточность	365
30.2.3. Системная память	365
ЧАСТЬ V. ПРАКТИКУМ ПО ПРОГРАММИРОВАНИЮ	367
Глава 31. Введение в Visual Prolog	369
31.1. Создание консольного проекта	369
31.2. Запуск программы	373
31.3. Расширение области видимости	376
31.4. Управление выводом в консоли	376
31.5. Использование классов <i>PFC</i>	377
Глава 32. Поиск с откатом на фактах	382
32.1. Цвета автомобилей	382
32.2. Точки на плоскости	385
32.3. Двенадцать месяцев	390
32.4. Зеленые и красные отсечения	390
Глава 33. Поиск с откатом на правилах	392
33.1. Родственные отношения	392
33.2. Моделирование комбинационных схем	397

33.3. Фигуры на плоскости	400
33.4. Ребус	405
Глава 34. Рекурсивные правила	408
34.1. Арифметика.....	408
34.2. Ряды	413
34.3. Длинная арифметика	415
34.4. Перевод чисел из одной системы счисления в другую	416
34.5. Обработка строк.....	423
Глава 35. Рекурсивные правила на списках.....	429
35.1. Списки	429
35.2. Математика	434
35.3. Треугольник Паскаля	435
35.4. Длинная арифметика	437
35.5. Преобразователь чисел из одной системы счисления в другую.....	439
35.6. Монеты	439
35.7. Домино	441
35.8. Ребус	442
35.9. Ребус с произвольными словами.....	444
35.10. Ребус с произвольными словами произвольной длины	446
35.11. Расстановка N ферзей на доске $N \times N$	448
35.12. Кувшины.....	452
35.13. Строки. Баланс скобок	454
35.14. Строки. Транслитерация	455
35.15. Списки. Поиск в ширину подписка в списке.....	456
Глава 36. Внутренняя база данных.....	459
36.1. Лексика русского языка	459
36.2. Искусственная жизнь.....	462
36.3. Представление базы данных списком фактов.....	466
Глава 37. Задачи на графах	468
Глава 38. Задачи на деревьях.....	473
Глава 39. Задачи на массивах	478
39.1. Решето Эратосфена	478
39.2. Сравнение частотных букварей текстов	481
ПРИЛОЖЕНИЯ	485
Приложение 1. Описание свойств и предикатов класса <i>console</i>.....	487
Свойства	487
Предикаты	487
Приложение 2. Описание предикатов класса <i>file</i>	491
Приложение 3. Описание конструкторов класса <i>InputStream_file</i>.....	495

Приложение 4. Описание конструкторов класса <i>outputStream_file</i>.....	497
Приложение 5. Пакет <i>stream</i>.....	500
Описание предикатов класса <i>inputStream</i>	500
Описание предикатов класса <i>outputStream</i>	501
Описание предикатов класса <i>stream</i>	501
Приложение 6. Меню Visual Prolog.....	503
Приложение 7. Быстрые клавиши меню Visual Prolog.....	509
Приложение 8. Панель инструментов Visual Prolog.....	512
Приложение 9. Описание предикатов класса <i>std</i>	514
Приложение 10. Описание класса <i>string</i>	517
Константа	517
Домены	517
Предикаты	517
Приложение 11. Описание электронного архива, сопровождающего книгу.....	534
Предметный указатель	535

Главы, помещенные в электронный архив

ЧАСТЬ VI. ПРОСТЫЕ ГРАФИЧЕСКИЕ ПРИЛОЖЕНИЯ

Глава 40. Часы и секундомер.....	1
40.1. Создание графического приложения	1
40.2. Конструирование формы	3
40.3. Разработка программного кода	7
40.3.1. Программирование часов.....	8
40.3.2. Программирование секундомера	10
Глава 41. Искусственная жизнь	13
41.1. Конструирование формы	13
41.2. Разработка программного кода	17
41.2.1. Константы, домены и факты класса.....	17
41.2.2. Размещение игрового поля на форме	18
41.2.3. Случайное заполнение колонии и сброс.....	22
41.2.4. Программирование зависимых переключателей	23
41.2.5. Старт/стоп таймера.....	24
41.2.6. Обработчик «тиков» таймера.....	24
41.2.7. Описание ручного режима моделирования.....	26
41.2.8. Редактирование колонии мышью.....	26
41.2.9. Создание и моделирование колонии.....	27

Глава 42. Машина Тьюринга.....	29
42.1. Создание формы графического приложения.....	29
42.2. Создание меню.....	30
42.3. Создание панели инструментов.....	35
42.4. Разработка графического интерфейса формы.....	42
42.5. Разработка статического класса <i>TuringMachine</i>	46
42.5.1. Описание предикатов класса <i>TuringMachine</i>	49
42.6. Разработка программного кода для пунктов меню.....	50
42.6.1. Меню <i>Файл</i>	51
42.6.2. Меню <i>Состояние ДМТ</i>	53
42.6.3. Меню <i>Запуск</i>	54
42.6.4. Меню <i>Справка</i>	57
42.7. Испытание проекта.....	58
Приложение 12. Описание графических элементов управления	62

ГЛАВА 1



Лексика языка Visual Prolog

Лексика представляет собой описание неделимых сущностей языка программирования, из которых, как из кирпичиков, строятся все вычислительные конструкции этого языка. неделимыми они являются потому, что воспринимаются компилятором языка как цельные фрагменты текста, хотя и состоят из последовательности символов алфавита. Такие неделимые сущности называются *лексемами*.

Некоторые лексемы — например, числа, ключевые слова и строки известны многим начинающим программистам. О других лексемах — двоичных данных, списках знают лишь опытные. В этой главе мы рассмотрим только лексемы Visual Prolog, а способы построения из них языковых конструкций будут описаны в следующих главах.

1.1. Алфавит

Алфавит Visual Prolog включает буквы английского и национального (русского) алфавитов, цифры, знаки пунктуации, операторы и ключевые слова. В языке есть различия между идентификаторами, начинающимися с заглавной и строчной буквы.

1.2. Комментарии

Для указания многострочных комментариев используются открывающие символы `/*` и закрывающие символы `*/`. Для указания однострочных комментариев служит знак процента `%`, действие которого распространяется до конца строки.

Пример комментариев:

```
/* Многострочные  
   комментарии */  
% Однострочные комментарии
```

1.3. Символы разметки текста

Таковыми символами являются пробелы, табуляции и символы перехода на новую строку. Комментарии и символы разметки во время компиляции не анализируются.

1.4. Лексемы

В качестве лексем языка Visual Prolog выступают ключевые слова, знаки пунктуации, операции, идентификаторы и литералы.

1.4.1. Ключевые слова

Ключевые слова разделяются на старшие и младшие. Это деление условное и предназначено только для различения цвета ключевых слов при их отображении в редакторе.

К старшим ключевым словам относятся:

class	domains	inherits	predicates
clauses	end	interface	properties
constants	facts	monitor	resolve
constructors	goal	namespace	supports
delegate	implement	open	

Младшие ключевые слова:

align	digits	failure	language	quot
and	div	finally	mod	rem
anyflow	do	foreach	multi	single
as	else	from	nondeterm	then
bitsize	elseif	guard	or	to
catch	erroneous	if	orelse	try
determ	externally	in	procedure	

Все ключевые слова, кроме `as` и `language`, зарезервированы, и использовать их в качестве имен, вводимых программистом в исходный текст программы, не допускается. Ключевое слово `end` всегда комбинируется с другими ключевыми словами:

end class	end implement	end interface
end if	end foreach	end try

1.4.2. Знаки пунктуации

Знаками пунктуации являются:

; ! , . # [] | () { } : :- ::

Многосимвольные знаки `:-` и `::` не должны разделяться пробелами.

1.4.3. Операции

Операции определяют вычисления, которые должны быть выполнены над указанными данными. Все операции бинарные, но плюс и минус могут быть и унарными.

```
+   -   /   *   ^   =   div   mod   quot   rem
<   >   <>  ><  <=  >=  :=   ==
```

Не допускается разрывать пробелами многосимвольные имена операций:

```
<>  ><  <=  >=  :=  ==  div  mod  quot  rem
```

1.4.4. Идентификаторы

В языке различают четыре вида идентификаторов: строчные, заглавные, анонимные и эллипсис.

Строчный идентификатор — последовательность букв, цифр и знаков подчеркивания, начинающаяся со строчной буквы.

Примеры правильных строчных идентификаторов:

```
S      жук001      w_ю_      xMary      й9я8ы
```

Примеры неправильных строчных идентификаторов:

```
8ver      % начинается с цифры
soap#1    % содержит символ #
Soap      % начинается с заглавной буквы
my prog   % содержит пробел
facts     % ключевое слово
```

Заглавный идентификатор — последовательность букв, цифр и знаков подчеркивания, начинающаяся с заглавной буквы или со знака подчеркивания.

Примеры правильных заглавных идентификаторов:

```
S      Жук_001      _Mary      Facts
```

Примеры неправильных заглавных идентификаторов:

```
8ver      % начинается с цифры
Soap$1    % содержит символ $
soap      % начинается со строчной буквы
My prog   % содержит пробел
```

Анонимный идентификатор — знак подчеркивания: `_`.

Эллипсис — знак многоточия: `...`

1.4.5. Литералы

В качестве литералов выступают целые и вещественные числа, символы, строки, двоичные данные, списки и составные термы.

Целые числа

Формат целых чисел зависит от системы счисления, в которой представлено число (табл. 1.1).

Таблица 1.1. Представление десятичных, восьмеричных и шестнадцатеричных чисел

Система счисления	Знак (опционально)	Префикс	Допустимые цифры	Примеры
Десятичное целое	±	Нет префикса	0123456789	355 +90 -897
Восьмеричное целое	±	0o	01234567	0o777 +0o121 -0o67
Шестнадцатеричное целое	±	0x	0123456789 AaBbCcDdEeFf	0xABC +0xc82 -0x3ff5

Вещественные числа

Вещественные числа представляются только в десятичной системе счисления (табл. 1.2, 1.3).

Таблица 1.2. Представление вещественных чисел

Мантисса				Порядок (опционально)		
Знак числа (опционально)	Цифры целой части	Десятичный раз-делитель	Цифры дробной части	Префикс показателя степени	Знак показателя степени	Цифры показателя степени
		опционально				
±	0123456789	.	0123456789	e или E	±	0123456789

Таблица 1.3. Примеры вещественных чисел

Число	Пояснение
-123.789	число без степени
+123.456E-5	+123.456·10 ⁻⁵
-0.123456e+12	-0.123456·10 ¹²

СИМВОЛЫ

Символ — любой печатаемый на клавиатуре символ, заключенный в апострофы, или ESC-последовательность, также заключенная в апострофы. *ESC-последовательность* (ESCAPE-последовательность) — это последовательность управляющих символов, начинающаяся с обратной наклонной черты — знака шиллинга \ (иногда этот знак называют *обратным слешем*). Именно обратная наклонная черта и является признаком ESC-последовательности:

- '\t' — табуляция (tabulation);
- '\n' — переход на новую строку (new line);
- '\r' — возврат в начало строки (return);
- '\uxxxx' — Unicode-символ, имеющий шестнадцатеричный код xxxx.

Кодировку Unicode-символов можно посмотреть, например, в приложении MS Word, используя меню **Вставка | Символ**. ESC-последовательность может определять не только действия, но и символы. Примеры символов и ESC-последовательностей представлены в табл. 1.4.

Таблица 1.4. Примеры символов

Символ	Пояснение
'f'	Буква f
'\u0066'	Буква f, заданная двухбайтовым шестнадцатеричным кодом
'Ж'	Буква Ж
'\u0416'	Буква Ж, заданная двухбайтовым шестнадцатеричным кодом
'\n'	Переход на новую строку
'\''	Апостроф '
'\\'	Обратная наклонная черта \
'\"'	Двойная кавычка "
'/'	Наклонная черта /
'\u2194'	Символ ↔
'\u203c'	Символ !!

С помощью Unicode-символов можно задавать символы, которых нет на клавиатуре.

Строки

Строка — один или более символов, заключенных в двойные кавычки. Строка не должна разрываться клавишей <Enter>. Строки могут содержать символы, представляемые ESC-последовательностями. Строки с префиксом @ могут многократно разрываться клавишей <Enter> и, кроме того, выполнение ESC-последова-

тельностью в такой строке подавляется. В табл. 1.5 показаны особенности обработки строк с префиксом @.

Таблица 1.5. Особенности обработки строк при наличии префикса @

Строка	Вывод на экран
"Visual prolog"	Visual prolog
"Visual\tprolog"	Visual prolog
@ "Visual prolog"	Visual prolog
@ "Visual\tprolog"	Visual\tprolog
"Visual prolog"	Ошибка компиляции
@ "Visual prolog"	Visual prolog

Префикс @ удобно использовать, например, при задании в программе пути к файлу, который обычно содержит несколько знаков обратной наклонной черты. С его помощью также удобно определять многострочный текст без многократного ввода ESC-последовательности \n.

Кроме того, префикс @ полезно применять для определения строк, заданных без кавычек и обрамленных парными символами: открывающим и закрывающим. В этом случае компилятор будет проверять наличие закрывающего символа, если был использован открывающий символ. В табл. 1.6 представлены допустимые открывающие и закрывающие символы.

Таблица 1.6. Открывающие и закрывающие парные символы

Парные символы		Парные символы	
Открывающий	Закрывающий	Открывающий	Закрывающий
@ ()	@)	(
@ {	}	@ }	{
@ []	@]	[
@ <	>	@ >	<

Например, выровненный по центру абзац HTML-текста можно задать в виде строки:

```
Str = @[<p align="center">Некоторый текст</p>]
```

Двоичные данные

Двоичные данные — последовательность байтов, заключенная в квадратные скобки, перед которыми указывается знак \$. Байты могут быть представлены десятич-

ными, восьмеричными или шестнадцатеричными целыми числами в диапазоне от 0 до 255, или арифметическими выражениями, значения которых может быть вычислено во время компиляции (табл. 1.7).

Таблица 1.7. Примеры двоичных данных

Двоичные данные	Вывод на экран
\$ [10, 20, 30]	\$ [0A, 14, 1E]
\$ [10, 0x20, 0o30]	\$ [0A, 20, 18]
\$ [4+4, 2*0x10]	\$ [08, 20]

Двоичные данные используются в Visual Prolog не только для связи с другими языками и для низкоуровневой обработки данных, но и в целях безопасного хранения и обмена данными произвольного типа, а также в качестве массива данных с произвольным доступом к элементам.

Списки

Синтаксически, список представляет собой заключенную в квадратные скобки последовательность элементов одного типа, разделенных запятой. Список может быть также выражен головой списка, роль которого играет первый элемент, и хвостом списка, отделенным от головы вертикальной чертой. Хвост списка является списком. Варианты записи списков приведены в табл. 1.8.

Таблица 1.8. Варианты записи списков

Список	Описание
[]	Пустой список
[10, 20, 30]	Список целых чисел
["123", "abcd"]	Список строк
[10, "abcd"]	Неправильный список
[3 W]	Список, состоящий из головы — числа 3 и хвоста, выраженного переменной W
[3 [45, 12]]	Список, состоящий из головы — числа 3 и хвоста, представленного списком [45, 12]
[[8, 45], [], [90]]	Список, состоящий из трех вложенных списков, один из которых пустой
[[]]	Непустой список, содержащий один пустой список

Составной терм

Составной терм определяется пользователем и служит для представления в программе структур данных. Составные термы описаны в главе 2.

ГЛАВА 2



Термы

2.1. Понятие термов

К *простым* термам относятся числа, символы, строки и идентификаторы, описанные в лексике языка.

Составные термы определяются через простые термы и составные термы, в том числе и через самих себя. Иерархия термов изображена на рис. 2.1.

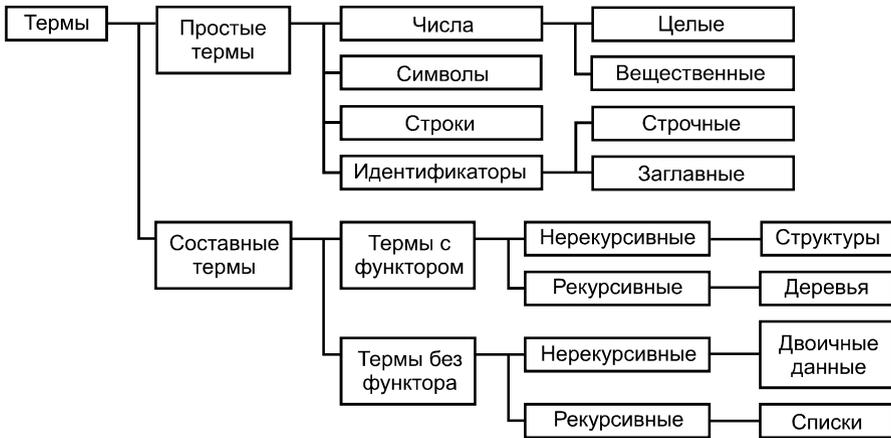


Рис. 2.1. Иерархия термов

Определение составного терма в общем случае рекурсивно. Согласно своему названию такой терм является именованной структурой, составленной из нескольких сущностей, как показано на рис. 2.2.

Имя терма называется *функтором* и обозначается строчным идентификатором. Количество аргументов называется *арностью*. Терм на рис. 2.2 является *N-арным* термом, иногда его называют *N-местным* термом, что есть то же самое. Каждый аргумент, в свою очередь, сам является простым или составным термом. Если один или несколько аргументов терма выражаются этим же термом, то говорят, что такой терм определен *рекурсивно*. В комментариях к логическим программам термы

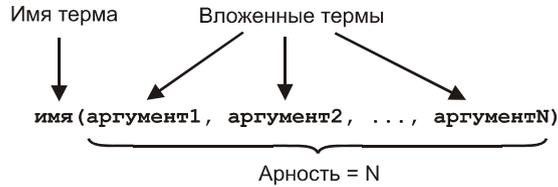


Рис. 2.2. Составной терм

иногда обозначают в виде имени и арности $\text{имя}/N$, опуская для краткости аргументы.

Составные термы могут не иметь аргументов. Такие термы содержат пустые скобки и называются *нульарными* термами.

Например, терм $f(999, \text{arm})$ имеет функтор f , является двухарным, первый аргумент — число 999, второй аргумент — строчный идентификатор arm , терм определен рекурсивно и имеет сокращенное обозначение $f/2$.

Терм $\text{spice}()$ является нульарным термом и в документации имеет сокращенное обозначение $\text{spice}/0$.

Рекурсивный терм $z(\text{babilon}, z(\text{inter}, \text{Near}, 8), 43)$ является трехарным. Первый аргумент — строчный идентификатор babilon , второй аргумент — *вложенный* терм с этим же функтором и арностью, третий аргумент — число 43. Вложенный терм $z(\text{inter}, \text{Near}, 8)$ содержит в качестве второго аргумента переменную Near , т. к. это заглавный идентификатор.

2.2. Переменные

Переменные в Прологе пишутся заглавными идентификаторами и могут быть либо свободными, либо связанными. Пока переменная *не связана* каким-либо значением, она называется *свободной*. Как только свободная переменная *связывается* каким-либо значением, она становится *связанной* и после этого не может изменить своего значения, будто является константой. Такой механизм связывания переменных называется *неразрушающим* присваиванием, в отличие от *разрушающего* присваивания, используемого в императивных языках и допускающего многократное изменение значения переменной.

Термин «переменная» в применении к Прологу достаточно условен и является данью традиции, привнесенной из императивных языков. Переменная в Прологе не меняет свое значение и, будучи свободной, она приобретает его только один раз. Чтобы связанная переменная приобрела другое значение, ее надо «отвязать» от текущего значения с помощью процесса, называемого *откатом*, т. е. произвести своеобразное «развоплощение». Механизм отката будет описан в *главе 6*. К настоящему времени в теории языков программирования не придуман такой термин, который бы адекватно отражал суть «переменных» Пролога с однократным связыванием. Поэтому мы будем использовать термин «переменная», надеясь на правильное его понимания читателем.

Термы, в которые не входят свободные переменные, называются *основными*. Термы, содержащие свободные переменные, называются *неосновными*.

Терм $f(999, \text{arm})$ является основным, а терм $z(\text{inter}, \text{Near}, 8)$ — неосновным, терм $z(\text{babilon}, z(\text{inter}, \text{Near}, 8), 43)$ также является неосновным, поскольку один из его аргументов — неосновной терм.

В Прологе используются и *анонимные* переменные — в тех случаях, где согласно синтаксису должна находиться переменная, но ее значение в вычислениях не используется. Анонимные переменные выражаются знаком подчеркивания или идентификатором, начинающимся со знака подчеркивания.

Мы рассмотрели *префиксные* термы, т. е. термы, функтор которых расположен слева от аргументов, перечисленных в скобках. В Прологе используются также *инфиксные* двухарные термы, у которых функтор располагается между аргументами и обозначается не строчным идентификатором, а символом арифметической или логической операции. Такие термы призваны повысить читабельность исходного текста программы. Например, вместо префиксного термина проверки неравенства $>(A, 5)$ в Прологе используют инфиксную запись этого термина $A > 5$. Такая запись более привычна для человека и, несмотря на отсутствие скобочной нотации и строчного идентификатора в качестве функтора, является допустимым в Прологе термом.

2.3. Списки

Наряду с префиксными и инфиксными терминами в Пролог для обозначения односвязных списков введен безфункторный терм, описанный в *главе 1*. *Список* есть специальный вид двоичного безфункторного рекурсивного термина, у которого первый аргумент — это голова списка, а второй аргумент является хвостом списка. Разделителем между ними служит вертикальная черта. Все элементы списка должны принадлежать к одному типу.

Списочный терм правоассоциативен. Это означает, что список, содержащий три элемента, — например, 10, 11 и 12, с теоретической точки зрения представляет собой терм $[10 \mid [11 \mid [12 \mid []]]]$. Такая запись списка трудна для восприятия, поэтому для упрощения записи и облегчения восприятия списки обозначаются кратко — в виде перечисления через запятую элементов, — например, $[10, 11, 12]$. Подобные упрощения синтаксиса в теории языков программирования называют «синтаксическим сахаром». Так, синтаксическим сахаром являются следующие варианты записи этого списка:

```
[10 | [11,12]]
[10, 11 | [12]]
[10, 11, 12 | []]
```

Далее представлены неправильные записи списков:

- $[1, 2|3]$ — хвост списка должен быть списком $[3]$, а не элементом 3;
- $[1, "aaa", 7]$ — элементы списка должны принадлежать одному типу;
- $[1 \mid [2] \mid [3]]$ — список имеет два хвоста, а допускается только один.

Если элементами списка являются тоже списки, то они называются *вложенными* списками, или списками первого уровня вложенности. Список может иметь произвольное число уровней вложенности.

Вертикальная черта, обозначающая разделитель между головой и хвостом, является операцией префикса списка. Префикс списка может быть как конструктором списка, так и селектором списка.

Конструктор собирает новый список из данного ему элемента, служащего головой нового списка, и данного ему списка, служащего хвостом нового списка. Например, если имеется элемент 5 и список [10, 20, 30], то новый список можно построить так: [5 | [10, 20, 30]]. Эту же операцию можно выразить формально:

```
A = 5,
X = [10, 20, 30],
NewList = [A|X].
```

В результате:

```
NewList = [5, 10, 20, 30].
```

Селектор разделяет заданный список на первый элемент, т. е. голову списка, и остаток списка, т. е. хвост. Например, если задан список [5, 10, 20, 30], то его можно разделить на голову и хвост, используя две свободные переменные, следующим образом:

```
[A|X] = [5, 10, 20, 30].
```

Результатом будет являться голова 5 и остаток списка [10, 20, 30]:

```
A = 5,
X = [10, 20, 30].
```

Селектор не применим к пустому списку [], т. к. пустой список не разделяется на голову и хвост. Однако список, содержащий лишь один элемент, разделяется на голову — этот самый единственный элемент, и хвост — пустой список.

И конструктор, и селектор синтаксически обозначаются одинаково — вертикальной чертой. Различие заключается в том, связаны или свободны переменные, между которыми стоит вертикальная черта. Если переменные связаны, то конструируется новый список. Если переменные свободны, то разделяется тот список, который приравнивается к этим свободным переменным, разделенным вертикальной чертой. Такое приравнивание в Прологе выполняет операцию сопоставления с образцом. Эта же операция сопоставления с образцом обрабатывает те случаи сопоставления, когда список содержит как свободные, так и связанные переменные.

2.4. Операция сопоставления с образцом

Сопоставление с образцом (pattern matching) — это операция сопоставления двух термов с целью поиска множества подстановок. Под *подстановкой* понимается замена всех вхождений переменной в терме каким-либо основным термом. *Множество подстановок* — это замена всех переменных терма основными термами.

Общий терм, полученный после замены переменных их значениями, должен быть основным, т. е. не должен содержать свободных переменных. Такое ограничение, наряду с отсутствием проверки вхождения переменной в терм, с которым сопоставляется эта переменная, не только повышает скорость выполнения кода, но и увеличивает его надежность за счет глобального анализа потока данных на этапе компиляции.

2.4.1. Сопоставление термов

Сопоставление термов выполняется за три этапа:

1. Проверка равенства функторов сопоставляемых термов.
2. Проверка равенства аргументности сопоставляемых термов.
3. Поиск множества подстановок.

Если подстановки всех переменных найдены и являются основными термами, то сопоставление с образцом считается успешным, иначе — сопоставление с образцом считается неуспешным, и замена переменных найденными значениями не производится.

Примеры сопоставления с образцом двух термов приведены в табл. 2.1.

Таблица 2.1. Примеры сопоставления с образцом

№	Терм 1	Терм 2	Сопоставление	Подстановки	Общий терм
1	X	24	Успешно	X=24	24 — основной
2	X	f(99)	Успешно	X=f(99)	f(99) — основной
3	ace()	X	Успешно	X=ace()	ace() — основной
4	ace()	ace(88)	Ошибка этапа компиляции	Нет	Нет
5	ace(88)	face(88)	Неуспешно	Нет	Нет
6	X	Y	Ошибка этапа компиляции	Нет	Нет
7	t(X, 45, Y)	t(2, Z, 7)	Успешно	X=2, Y=7, Z=45	t(2, 45, 7) — основной
8	t(X, 45, Y)	t(Z, Z, 8)	Ошибка этапа компиляции	Нет	Нет
9	t(X, 45, Y)	t(Z, 45, 8)	Ошибка этапа компиляции	Нет	Нет
10	t(X, 45, Y)	t(Z, X, W)	Ошибка этапа компиляции	Нет	Нет
11	t(X, 45, Y)	g(2, 45, 7)	Неуспешно	Нет	Нет
12	t(X, 45, Y)	t(X, 45)	Ошибка этапа компиляции	Нет	Нет

Таблица 2.1 (окончание)

№	Терм 1	Терм 2	Сопоставление	Подстановки	Общий терм
13	$t(X, 45, 8)$	$t(2, Z, Z)$	Неуспешно	Нет	Нет
14	X	$s(X)$	Ошибка этапа компиляции	Нет	Нет
15	$ace(88)$	$ace(88)$	Успешно	Нет	$ace(88)$ — основной

В строке 4 табл. 2.1 Visual Prolog допускает использование в программе двух термов с одинаковым именем, но с разной арностью. Такие термы считаются разными, что видно в строках 4 и 12.

Сопоставление двух термов, указанных в строке 8, не будет скомпилировано, т. к. в ходе глобального анализа потока данных компилятор обнаружит, что одной из промежуточных подстановок является подстановка $X=Z$, т. е. связывание двух свободных переменных, что не допускается алгоритмом сопоставления с образцом.

Сопоставление термина с анонимной переменной всегда успешно.

Операция сопоставления с образцом выражается явно знаком равенства — например, $t(X, 45, Y) = t(2, Z, 7)$. С другой стороны, сопоставление с образцом выполняется без явного указания знака равенства — например, при вызове функции с заданными параметрами. Параметры вызова сопоставляются с соответствующими параметрами, заданными в определении функции.

Операция сопоставления с образцом безразлична к расположению свободных переменных относительно знака равенства. Поэтому операции $X=7$ и $7=X$ эквивалентны.

2.4.2. Сопоставление списков

Списки являются безфункторными термами, поэтому сопоставление списков выполняется аналогично сопоставлению термов — последовательным сопоставлением элементов двух списков между собой. Если в списках есть вложенные списки, то выполняется сопоставление их элементов. В табл. 2.2 приведены примеры сопоставления двух списков.

Таблица 2.2. Примеры сопоставления с образцом для списков

№	Список 1	Список 2	Сопоставление с образцом	Подстановки	Общий терм
1	X	$[10, 20, 30]$	Успешно	$X=[10, 20, 30]$	$[10, 20, 30]$
2	$[X, 20, 30]$	$[10, 20, 30]$	Успешно	$X=10$	$[10, 20, 30]$
3	$[X, 20, 30]$	$[10, Y, 30]$	Успешно	$X=10, Y=20$	$[10, 20, 30]$
4	$[X, 20, 30]$	$[10, Y, 50]$	Неуспешно	Нет	Нет
5	$[X, 20, 30]$	$[X, 20]$	Неуспешно	Нет	Нет