Посвящаю третье издание этой книги моей матери, самому доброму человеку из всех, кого я знаю, и моему отцу, который во время Второй Мировой войны сумел убежать из концентрационного лагеря, прорыв подземный туннель, о чем он написал в своем романе “Телескоп”.

Из предисловия Патрика Уинстона ко второму изданию

Никогда не забуду того волнения, которое я испытал, увидев в действии свою первую программу, написанную в стиле Prolog. Эта программа была частью знаменитой системы Shrdlu Терри Винограда (Terry Winograd). Решатель задач, встроенный в систему, работал в "мире блоков" и заставлял руку робота (точнее, ее модель) перемещать кубики на экране дисплея, решая хитроумные задачи, поставленные оператором.

Решатель задач в мире блоков Винограда был написан на языке Microplanner, который, как мы теперь понимаем, был своего рода языком Prolog в миниатюре. Несмотря на все недостатки языка Microplanner, вся работа решателя задач в мире блоков была явно нацелена на достижение определенных целей, поскольку любой язык типа Prolog побуждает программистов мыслить в терминах целей. Благодаря использованию таких целенаправленных процедур, как "схватить", "освободить верх блока", "убрать в сторону", "переместить" и "отпустить", действия этой ясной, понятной и краткой программы казались удивительно интеллектуальными.

Решатель задач в мире блоков Винограда навсегда изменил мое представление о программировании. Я даже переписал этот решатель задач в мире блоков на языке Lisp и привел в своем учебнике по Lisp, поскольку эта программа неизменно впечатляла меня мощью заложенной в ней философии целевого программирования и той легкостью, с какой создаются целенаправленные программы.

Однако обучение целевому программированию на примерах программ Lisp можно сравнить с чтением произведений Шекспира не на английском языке. Это позволяет получить определенное впечатление, но эстетическое воздействие становится гораздо слабее, чем при восприятии оригинала. Поэтому лучшим способом обучения целевому программированию является чтение и написание программ на языке Prolog, поскольку он как раз и предназначен для программирования в терминах целей.

Вообще говоря, развитие языков программирования происходит на пути перехода от языков низкого уровня, в которых программист определяет, как должны быть выполнены те или иные действия, к языкам высокого уровня, позволяющим просто указать, что должно быть сделано. Например, с появлением языка Fortran программисты избавились от необходимости общаться с компьютером на прокрустовом языке адресов и регистров. Теперь они уже могли выражать свои мысли на естественном (или почти естественном) языке, не считая той небольшой уступки, что для этого должны были использоваться ограниченные, 80-колонные перфокарты.

Однако Fortran и почти все другие языки программирования все еще остаются языками процедурного типа, которые требуют точно указывать весь процесс решения задачи. На мой взгляд, современный Lisp достиг абсолютного предела развития языков этого типа, поскольку Lisp (вернее, его версия Common Lisp) обладает непревзойденными выразительными возможностями, но качество создаваемых на нем программ полностью зависит от того, насколько сможет ими воспользоваться сам программист. А с появлением языка Prolog, с другой стороны, наметился четко выраженный переход к использованию языков декларативного типа, которые побуждают программиста описывать ситуации и формулировать задачи, а не регламентировать во всех подробностях процедуры решения этих задач.

Отсюда следует, насколько важен вводный курс по языку Prolog для всех студентов, изучающих вычислительную технику и программирование, — ведь просто не существует лучшего способа понять, что представляет собой декларативное программирование.

Многие страницы этой книги могут служить хорошей иллюстрацией того различия, которое существует между процедурным и декларативным стилями программистского мышления. Например, в первой главе это различие иллюстрируется на задачах, относящихся к семейным отношениям. Язык Prolog позволяет явно и естественно определить понятие "дед" (grandfather), указав, что дед — это отец (father) одного из родителей (parent). На этом языке определение предиката grandfather выглядит так:

```
grandfather( X, Z) :- father( X, Y), parent( Y, Z).
```

И сразу после ввода в систему Prolog определения предиката grandfather ей можно задать вопрос, например, каково имя каждого из дедов Патрика. Ниже приведен вопрос, который снова сформулирован в системе обозначений Prolog, наряду с типичным ответом.

```
?- grandfather( X, patrick).
X = james;
X = carl
```

Именно система Prolog должна найти способ решения этой задачи, просматривая в базе данных все, что касается отношений father и parent. Программист указывает только, что дано и на какой вопрос должен быть получен ответ. Он в большей степени обязан предоставить системе нужную информацию, чем алгоритмы, с помощью которых ведется обработка этой информации.

Поняв, что очень важно изучить Prolog, нужно решить, как лучше это сделать. Я убежден, что изучение языка программирования во многом сходно с изучением естественного языка, Так, например, в первом случае можно воспользоваться справочным руководством, а во втором — словарем. Но никто не изучает язык по словарю, поскольку слова — это только часть знаний, необходимых для овладения языком. Для этого необходимо также узнать, по каким правилам формируются осмысленные сочетания слов, а затем приобщиться к высшим достижениям искусства речи под руководством тех, кто владеет литературным стилем.

Точно так же, никто не изучает язык программирования только по справочному руководству, поскольку в нем почти ничего не говорится о том, как используют элементарные конструкции языка те, кто им виртуозно владеет. Поэтому нужен учебник, а признаком хорошего учебника является изобилие примеров, ведь в них сосредоточен богатый опыт, на котором мы в основном и учимся.

В этой книге первый пример появляется уже на первой странице, а далее на читателя как из рога изобилия обрушивается поток примеров программ на языке Prolog, написанных программистом-энтузиастом, горячим приверженцем декларативной идеологии программирования. После тщательного изучения этих примеров читатель не только узнает, как действует система Prolog, но и станет обладателем личной коллекции образцов программ, готовых к употреблению: он может разбирать эти программы на части, приспосабливать каждую часть к своей задаче, а затем снова собирать их вместе, получая при этом новые программы. Такое усвоение предшествующего опыта можно считать первым шагом на пути к овладению мастерством программирования.

Полезным побочным эффектом изучения примеров качественных программ является то, что они позволяют не только освоить само программирование, но и многое узнать об интересной области науки. В данной книге такой научной дисциплиной, лежащей в основе большинства примеров, является искусственный интеллект. Читатель ознакомится с такими идеями в области автоматического решения задач, как сведение задач к подзадачам, прямое и обратное построение цепочки рассуждений, получение объяснения последовательности рассуждений и объяснения предпосылок, а также освоит различные методы поиска.

Одним из замечательных свойств языка Prolog является то, что он достаточно прост, чтобы студенты могли его использовать непосредственно в процессе изучения вводного курса по искусственному интеллекту. Я не сомневаюсь, что многие преподаватели включат эту книгу в свои курсы искусственного интеллекта с тем, чтобы студенты смогли сами увидеть, как абстрактные идеи приобретают конкретные и действенные формы.

Полагаю, что среди учебников по языку Prolog эта книга окажется особенно популярной, и не только благодаря наличию качественных примеров, но и в связи с другими ее привлекательными особенностями:

- во всех главах книги имеются тщательно составленные резюме;
- все изучаемые понятия подкрепляются многочисленными упражнениями;
- понятие абстракции данных представлено наиболее наглядно — с помощью процедур доступа к элементам структур;
- обсуждению вопросов стиля и методологии программирования посвящена целая глава;
- со всей откровенностью обозначены все трудности, с которыми приходится сталкиваться в ходе программирования на языке Prolog, а не только привлекательные стороны этого процесса.

Все это говорит о том, что перед нами прекрасно написанная, интересная и полезная книга.

Я до сих пор храню первое издание этой книги в своей библиотеке, где она стоит на одной полке с другими выдающимися учебниками по языкам программирования, поскольку обладает такими превосходными особенностями, как ясность и прямота изложения, а также наличие многочисленных примеров, тщательно составленных резюме и многочисленных упражнений. А поскольку эта книга является учебником по языку программирования, мне особенно нравится сделанный в ней акцент на абстракцию данных, внимательное отношение к выбору стиля программирования и непредвзятый анализ не только преимуществ, но и недостатков языка Prolog.

Предисловие

Язык Prolog

Prolog — это язык программирования, сосредоточенный вокруг небольшого набора основных механизмов, включая сопоставление с образцом, древовидное представление структур данных и автоматический перебор с возвратами. Этот ограниченный набор средств образует удивительно мощную и гибкую среду программирования. Prolog особенно хорошо подходит для решения задач, в которых рассматриваются объекты (в частности, структурированные объекты) и отношения между ними. Например, на языке Prolog совсем не сложно выразить пространственные связи между объектами, допустим, указать, что синий шар находится за зеленым. Столь же просто определить можно более общее правило: если объект `X` ближе к наблюдателю, чем объект `Y`, а `Y` ближе, чем `Z`, то `X` должен быть ближе, чем `Z`. После этого система Prolog получает возможность формировать рассуждения о пространственных связях и их совместимости с общим правилом. Благодаря таким особенностям Prolog становится мощным языком для искусственного интеллекта и нечислового программирования в целом. Можно указать известные примеры символических вычислений, реализация которых на других стандартных языках вызывает необходимость создавать десятки страниц кода, чрезвычайно сложного в изучении. А после реализации тех же алгоритмов на языке Prolog результатом становится кристально ясная программа, которая легко помещается на одной странице.

Основные вехи развития языка Prolog

Prolog стал воплощением идеи использования логики в качестве языка программирования, которая зародилась в начале 1970-х годов, и само его название является сокращением от слов “programming in logic” (программирование в терминах логики). Первыми исследователями, которые занялись разработкой этой идеи, были Роберт Ковальски (Robert Kowalski) из Эдинбурга (теоретические основы), Маартен ван Эмден (Maarten van Emden) из Эдинбурга (экспериментальная демонстрационная система) и Ален Колмероэ (Alain Colmerauer) из Марселя (реализация). Популяризации языка Prolog во многом способствовала эффективная реализация этого языка в середине 1970-х годов Дэвидом Д. Г. Уорреном (David D.H. Warren) из Эдинбурга. К числу новейших достижений в этой области относятся средства программирования на основе логики ограничений (Constraint Logic Programming — CLP), которые обычно реализуются в составе системы Prolog. Средства CLP показали себя на практике как исключительно гибкий инструмент для решения задач составления расписаний и планирования материально-технического снабжения. А в 1996 году был опубликован официальный стандарт ISO языка Prolog.

Наиболее заметные тенденции в истории развития языка Prolog

В развитии языка Prolog наблюдаются очень интересные тенденции. Этот язык быстро приобрел популярность в Европе как инструмент практического программирования. В Японии вокруг языка Prolog были сосредоточены все разработки компьютеров пятого поколения. С другой стороны, в США этот язык в целом был принят с

небольшим опозданием в связи с некоторыми историческими причинами. Одна из них состояла в том, что Соединенные Штаты вначале познакомились с языком Microplanner, который также был близок к идее логического программирования, но неэффективно реализован. Определенная доля низкой популярности Prolog в этой стране объясняется также реакцией на существовавшую вначале "ортодоксальную школу" логического программирования, представители которой настаивали на использовании чистой логики и требовали, чтобы логический подход не был "запятнан" практическими средствами, не относящимися к логике. В прошлом это привело к широкому распространению неверных взглядов на язык Prolog. Например, некоторые считали, что на этом языке можно программировать только рассуждения с выводом от целей к фактам. Но истина заключается в том, что Prolog — универсальный язык программирования и на нем может быть реализован любой алгоритм. Далекая от реальности позиция "ортодоксальной школы" была преодолена практиками языка Prolog, которые приняли более прагматический подход, воспользовавшись плодотворным объединением нового, декларативного подхода с традиционным, процедурным.

Изучение языка Prolog

Поскольку истоки языка Prolog лежат в области математической логики, преподавание этого языка часто начинают с изучения логики. Но, как оказалось, подобный вводный курс, насыщенный математическими дисциплинами, не очень эффективен, если цель состоит в изучении Prolog как инструмента практического программирования. Поэтому в данной книге меньше всего внимания уделяется математическим аспектам, а изложение основывается на изучении искусства использования нескольких базовых механизмов Prolog для решения интересных задач. Обычные языки программирования — процедурно ориентированны, а Prolog является представителем нового поколения программных средств, основанных на применении описательного, или декларативного, подхода. Для его освоения требуется во многом изменить свое представление о способах решения задач, овладеть намного более продуктивными подходами, поэтому обучение программированию на языке Prolog становится увлекательной интеллектуальной деятельностью. Многие полагают, что каждый, кто проходит обучение в области информатики, должен на определенном этапе получить хотя бы некоторое представление о языке Prolog, поскольку этот язык требует использования иного принципа решения задач и позволяет по-другому взглянуть на традиционные языки программирования.

Содержание книги

В части I приведены вводные сведения о языке Prolog и показан процесс разработки программ Prolog. Кроме того, в нее включено описание методов обработки таких важных структур данных, как деревья и графы, поскольку эти методы находят широкое распространение. В части II показано применение языка Prolog во многих областях искусственного интеллекта, включая решение задач и эвристический поиск, программирование в ограничениях, представление знаний и экспертные системы, планирование, машинное обучение, качественные рассуждения, обработка текста на различных языках и ведение игр. Методы искусственного интеллекта описываются и разрабатываются до такой степени детализации, которая позволяет успешно реализовать их на языке Prolog и получить законченные программы. Затем эти программы могут использоваться в качестве структурных блоков для сложных приложений. В заключительной главе, посвященной метапрограммированию, показано, как можно применять Prolog для реализации других языков и принципов программирования, включая объектно-ориентированное программирование, программирова-

ние, управляемое шаблонами, а также написание интерпретаторов Prolog на языке Prolog. Во всей книге акцент делается на создание наиболее удобных для понимания программ; в этих программах исключены все затрудняющие понимание "программистские трюки", которые основаны на средствах системы, зависящих от конкретной реализации.

Различия между вторым и третьим изданиями

Весь материал книги пересмотрен и обновлен. Введены новые главы, в которых рассматриваются следующие темы:

- программирование на основе логики ограничений (Constraint Logic Programming — CLP);
- индуктивное логическое программирование;
- качественные рассуждения.

В числе других важных изменений можно назвать следующие:

- описание сетей доверия (байесовских сетей) в главе по представлению знаний и экспертным системам;
- описание программ поиска по заданному критерию (IDA*, RBFS), характеризующихся низкими требованиями к памяти, в главе по эвристическому поиску;
- существенные дополнения в главе, посвященной машинному обучению;
- описание дополнительных методов повышения эффективности программ в главе, посвященной стилю и методам программирования.

Во всей книге больше внимания уделено различиям между реализациями Prolog и, в случае необходимости, даны конкретные ссылки на стандарт Prolog (см. также приложение А).

Для кого предназначена эта книга

Данная книга предназначена для тех, кто проходит обучение в области языка Prolog и искусственного интеллекта. Ее можно использовать в составе курса по языку Prolog или курса по искусственному интеллекту, в котором принципы искусственного интеллекта иллюстрируются на основе Prolog. Предполагается, что читатель имеет основной запас общих знаний в области компьютерных наук, но наличие знаний в области искусственного интеллекта не требуется. Не обязательна также значительная подготовка в области программирования; напротив, богатый опыт и приверженность обычному процедурному стилю программирования (например, на языке C или Pascal) могут даже стать препятствием к освоению нового способа мышления, который требуется при изучении языка Prolog.

Стандартный синтаксис, используемый в книге

Среди нескольких диалектов Prolog наиболее широкое распространение получил так называемый *эдинбургский синтаксис*, известный также как *синтаксис DEC-10*; этот же диалект стал основой стандарта ISO для языка Prolog. Он применяется и в данной книге. Для обеспечения совместимости с различными реализациями Prolog в этой книге используется лишь относительно небольшое подмножество встроенных средств, которые являются общими для многих версий Prolog.

Как читать эту книгу

Прежде всего необходимо прочитать подряд весь материал части I. Но раздел 2.4 с описанием процедурного смысла программ Prolog является более сложным, поэтому трудный для восприятия материал этого раздела при первом чтении можно пропустить. В главе 4 представлены примеры программ, которые могут быть прочитаны (или пропущены) избирательно. Глава 10, посвященная сложным способам представления деревьев, также может быть пропущена.

В части II читатель получает больше возможностей выбора изучаемого материала, поскольку главы этой части в основном не зависят друг от друга. Но, вполне естественно, некоторые темы все еще должны быть изучены после других, например, те, что опираются на основные стратегии поиска (глава 11). На рис. 1 показана рекомендуемая последовательность чтения глав.

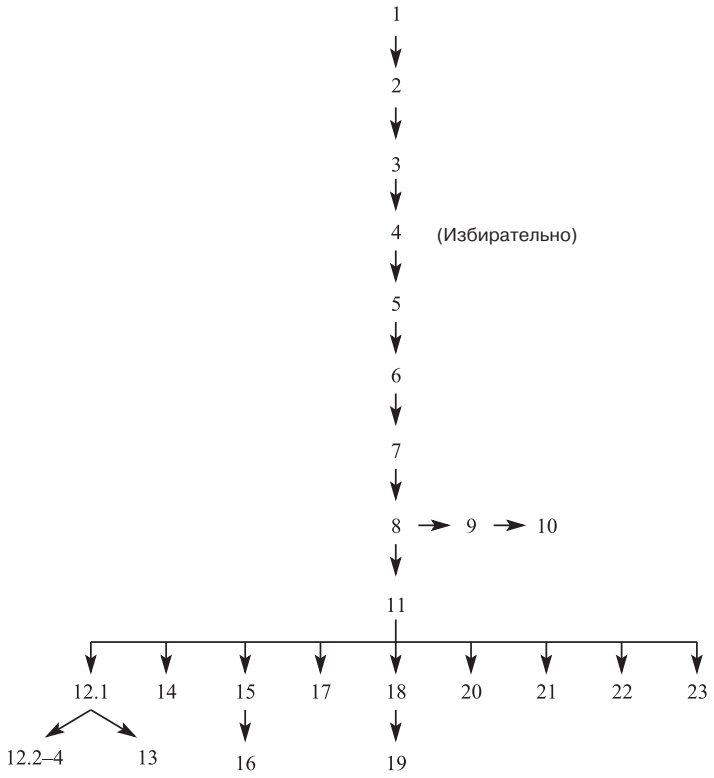

Рис. 1. Рекомендуемая последовательность изучения глав книги

Код программ и материалы курсов

Исходный код всех программ этой книги и соответствующие материалы курсов можно получить на сопровождающем Web-узле (`www.booksites.net/bratko`).

Благодарности

Пробуждению у меня интереса к языку Prolog я обязан Дональду Мичи (Donald Michie). Я очень благодарен Лоренсу Берду (Lawrence Byrd), Фернандо Перейре (Fernando Pereira) и Дэвиду Г. Д. Уоррену (David H. D. Warren), которые в свое время входили в состав группы разработчиков Prolog в Эдинбурге, за их советы по программированию и интересные дискуссии. Книга стала намного лучше под влиянием комментариев и предложений к предыдущим изданиям Эндрю Макгеттрика (Andrew McGettrick) и Патрика Г. Уинстона (Patrick H. Winston). Из числа тех, кто прочитал отдельные части рукописи и внес существенные комментарии, хочу особо отметить Дамиана Бояджиева (Damjan Bojadžiev), Рода Брайстоу (Rod Bristow), Питера Кларка (Peter Clark), Франса Кенена (Frans Coenen), Дэвида К. Додсона (David C. Dodson), Сашо Джероски (Sašo Džeroski), Богдана Филипича (Bogdan Filipič), Вана Фоккинка (Wan Fokkink), Матьяжа Гамса (Matjaž Gams), Питера Г. Гринфилда (Peter G. Greenfield), Марко Гробелника (Marko Grobelnik), Криса Хинде (Chris Hinde), Игоря Кононенко (Igor Kononenko), Матевжа Ковачича (Matevž Kovačič), Эдуардо Моралеса (Eduardo Morales), Игоря Можетича (Igor Možetič), Тимоти Б. Ниблетта (Timothy B. Niblett), Душана Петерца (Dušan Peterc), Уроша Помпе (Uroš Pompe), Роберта Родошека (Robert Rodošek), Агату Сахе (Agata Saje), Клода Саммю (Claude Sammut), Сима Сэя (Cem Say), Ашвина Сринивасана (Ashwin Srinivasan), Дориана Сью (Dorian Sue), Питера Тансига (Peter Tancig), Таню Урбанчич (Tanja Urbančič), Марка Уоллеса (Mark Wallace), Уильяма Уолли (William Walley), Саймона Уэйлгани (Simon Weilguny), Блажа Зупана (Blaž Zupan) и Дарко Зупанича (Darko Zupanič). Выражаю особую признательность Симу Сэю за то, что он проверил значительную часть программ и преподнес мне ценный подарок, обнаружив скрытые ошибки. Некоторые читатели, особенно Г. Улснам (G. Oulsnam) и Изток Тврды (Iztok Tvrdy), помогли выявить ошибки в предыдущих изданиях. Хочу также поблагодарить Карен Мосман (Karen Mosman), Джули Найт (Julie Knight) и Карен Сазерленд (Karen Sutherland), сотрудников издательства Pearson Education, за их плодотворную работу в процессе подготовки этой книги. Саймон Пламтри (Simon Plumtree) и Дебра Майзон-Этерингтон (Debra Myson-Etherington) внесли большой вклад в предыдущие издания. Основная часть иллюстраций была подготовлена Дарко Симершеком (Darko Simeršek). И, наконец, хочу отметить, что эта книга не могла бы появиться в свет без стимулирующего влияния творческой деятельности всего международного сообщества специалистов по логическому программированию.

Издательство Pearson Education выражает свою признательность корпорации Plenum Publishing Corporation за полученное разрешение воспроизвести материал, аналогичный приведенному в главе 10 книги *Human and Machine Problem Solving (1989), K. Gilhooly (ed.)*.

Иван Братко
Январь 2000 года

От редактора перевода

Считаю свои долгом привести несколько рекомендаций, которые, надеюсь, позволят читателю быстрее приступить к работе с этой замечательной книгой.

Выбор дистрибутива Prolog

Принятие стандарта языка Prolog стало стимулом для создания многих независимых реализаций этого языка, в том числе предоставляемых бесплатно для широкого круга пользователей. Кроме того, и для многих коммерческих систем предусмотрены демонстрационные версии, с лицензией на ограниченный, но достаточно большой период времени, а также так называемые *персональные выпуски* (Personal Edition), которые предоставляются бесплатно для личного пользования. Одним из самых интересных продуктов последнего типа является Visual Prolog (`http://www.visual-prolog.com` или `http://www.pdc.dk`). Поэтому читатель может без особых сложностей найти и загрузить наиболее подходящую для себя версию. Постоянно обновляемый обзор новейших реализаций Prolog можно найти по адресу

`http://dmoz.org/Computers/Programming/Languages/Prolog/Implementations/`

На указанной странице отмечен звездочкой бесплатный пакет SWI-Prolog, который действительно представляет наибольший интерес для тех, кто только приступает к изучению этого языка.

Режимы работы интерпретатора Prolog

Все примеры, приведенные в данной книге, могут быть выполнены с помощью интерпретатора Prolog. После вызова на выполнение интерпретатор выводит приглашение "`?-`", которое свидетельствует о том, что работа ведется в режиме выполнения запросов. После ввода запроса интерпретатор обращается к своей базе данных, находит в ней факты и правила, необходимые для ответа на запрос, формирует и выводит ответ. Непосредственно после загрузки в базе данных интерпретатора находятся только стандартные предикаты, обеспечивающие работу системы и выполнение вспомогательных функций. Задача ввода всего того, что требуется для решения практической проблемы (аналога программы на других языках программирования), возлагается на пользователя. В ходе этого система как бы "накапливает знания", поэтому весь этот процесс принято называть *консультированием* (consulting). Для консультирования системы можно либо пользоваться предикатом `assert` (а также его версиями `asserta` и `assertz`), либо ввести программу из файла. Файл с программой может находиться на внешнем носителе информации, и в таком случае для его ввода применяется предикат `consult` с указанием имени файла. Есть также сокращенный способ указания имени файла — в квадратных скобках. Последний способ позволяет также ввести необходимые факты и правила непосредственно в командной строке интерпретатора. После ввода в приглашении имени "псевдофайла" пользовательского терминала `user` система переходит в режим ввода программы с терминала, как показано ниже.

```
?- [user].
|: man(dima).
|: man(kolya).
|: man(petr).
|: end_of_file.
% user://3 compiled 28.12 sec, 80 bytes
```

Для завершения работы в этом режиме необходимо ввести атом `end_of_file.`, с точкой в конце (некоторые системы позволяют завершить ввод с помощью комбинации клавиш, обозначающих конец ввода, таких как <Ctrl+D> или <Ctrl+Z>).

Разработка на языке Prolog с учетом декларативных и процедурных аспектов

Вне всякого сомнения, программа на языке Prolog способна стать "мозгом" чрезвычайно интересных приложений. Но иногда выполнение чисто "телесных" функций (ввод-вывод, преобразование форматов данных, диалог с пользователем и т.д.) целесообразно возложить на другие языки программирования. В настоящее время подобная практика разработки прикладных программных комплексов на основе Prolog находит очень широкое распространение. К счастью, системы Prolog очень хорошо сочетаются с такими системами программирования, как Java, Delphi, C++ и др. Это позволяет оптимальным образом реализовать в разрабатываемом приложении и процедурные, алгоритмические функции, и методы решения декларативных, интеллектуальных задач. Соответствующие инструментальные средства можно легко найти в Web. (Укажите в поисковом запросе "Prolog" и нужный вам язык программирования.)