

А. Мачуговский

Москва

Пролог-Д для БК 0010 и БК 0011М

Введение

Первая версия языка программирования Пролог-Д для компьютера БК-0010 была выпущена в 1987-ом году С. Григорьевым. Версия 1.1 появилась, вероятно, в 1992-ом году. Она прошивалась в ПЗУ и для сохранения (а также загрузки) программ использовала магнитофон. По сравнению с предыдущими версиями, оставляла больше места для стека и для программы пользователя.

В 2019-ом году ПЗУ-версия была адаптирована для запуска в операционных системах MKDOS и ANDOS на БК-0010.01 и БК-0011М с контроллером жёсткого диска SMK. Для работы использует расширенную память контроллера. Адаптированный Пролог-Д можно загрузить и с магнитофона (при этом БК 0011М должна быть переведена в режим БК 0010 предварительной загрузкой в память Монитора от БК 0010).

Редактирование текста

После запуска Пролог-Д находится в режиме «Редактирование».

Текстовый редактор не поддерживает автоповтор при удержании клавиши. Используйте клавишу ПОВТ. Помимо клавиш управления курсором, обрабатываются клавиши «забой», «сдвигка», «раздвигка» и «удаление до конца строки».

Максимально допустимая длина строки - 64 символа.

Нажатие клавиши КТ переводит в режим ожидания команды пользователя. Всего доступно три команды, они вызываются клавишами «Ч» и «З» в русском заглавном регистре и клавишей «?». Если нажата иная клавиша, она обрабатывается как обычно. После обработки команд пользователя возврат в режим редактирования происходит по нажатию любой клавиши. Команды:

? - вызов подсказки (довольно бесполезной).

Ч - чтение программы с кассеты или диска. Завершение чтения никак не обозначается.

З - запись программы. Рекомендуется использовать расширение файла «.PL»

Основы языка

Программу составляют из *фактов* и *правил*. В каждой строке можно указать только один факт или одно правило. Строка должна заканчиваться символом «точка с запятой».

Пример *фактов*:

```
RUSENG ( РАДОСТНЫЙ , GLAD ) ;  
RUSENG ( РАДОСТНЫЙ , SUNNY ) ;  
RUSENG ( СОЛНЕЧНЫЙ , SUNNY ) ;  
RUSENG ( СОЛНЕЧНЫЙ , SOLAR ) ;  
RUSENG ( СОЛНЕЧНЫЙ , SHINY ) ;  
RUSENG ( БЛЕСТЯЩИЙ , SHINY ) ;
```

Текстовые значения записываются в Прологе-Д без кавычек. Регистр учитывается, apple и APPLE - два разных значения. Важно придерживаться *выбранного порядка объектов*. В данном случае первым всегда указано русское слово, вторым - английское.

Пример правила:

```
СИНОНИМ( X, Y ) <- RUSENG( X, Z ), RUSENG( Y, Z );
```

RUSENG и СИНОНИМ - имена созданных программистом *предикатов*. В терминологии Пролога *предикат* - утверждение, задающее отношения между объектами. Первый пример показывает создание предиката RUSENG и наполнение его *фактами* из русско-английского словаря. Во втором примере создан предикат СИНОНИМ, описывающий *правило* нахождения синонимов для русских слов. Это правило гласит: для русских слов X и Y существует такое английское слово Z, что X переводится как Z, и Y переводится как Z.

После имени предиката всегда пишут скобки (даже если предикату не нужно передавать никаких объектов).

Предикат - логическая величина, она принимает значения ДА или НЕТ. Предикат может содержать несколько фактов и/или правил.

Факт всегда выдает значение ДА.

Правило в левой части содержит имя и список объектов. При дальнейшем использовании правила важно *придерживаться выбранного порядка объектов*. Справа от символов <- идет *описание* правила: через запятую перечислены предикаты (факты и правила), их ещё называют *целями*. Когда все они принимают значение ДА, тогда и само правило выдает значение ДА. По сути, символ «,» между предикатами обозначает логическую операцию И.

Описание целей допускает *рекурсию*, то есть при необходимости в описании правила можно использовать имя самого этого правила.

Совокупность фактов и правил называют *базой знаний*.

Вопрос записывают последней строкой программы и обозначают символом "?". Его можно писать слитно со следующим за ним предикатом (фактом или правилом), а можно и через пробел. Пролог-Д ответит на вопрос, учитывая перечисленные выше факты и правила. Программа может содержать только один вопрос. Следующие за ним строки программы игнорируются.

Пример вопроса:

```
? RUSENG( СОЛНЕЧНЫЙ, SAD );
```

В данном примере Пролог-Д проверит предикат RUSENG на содержание факта «значению СОЛНЕЧНЫЙ соответствует значение SAD». Такого факта нет в базе знаний, поэтому ответ на вопрос - НЕТ. Он и появится на экране.

Переменные в Прологе-Д однобуквенные - строчные и заглавные символы от A до Z и от A до Я. Поэтому внести в наш англо-русский словарь значения «I» и «Я», к сожалению, не получится. Область видимости переменной - одна строка. Можно использовать одинаковые имена переменных при создании разных правил. Для простых правил традиционно используют переменные X, Y и Z.

Вопрос с использованием *переменной* Y:

```
? RUSENG( СОЛНЕЧНЫЙ, Y );
```

Пролог-Д рассмотрит записанные в базе знаний факты и напечатает на экране все подходящие значения переменной Y:

```
Y=SUNNY
Y=SOLAR
Y=SHINY
ДРУГИХ РЕШЕНИЙ НЕТ
```

Технически это работает так. Факт RUSENG задает отношение между двумя объектами. Пролог-Д синтезирует новые факты: на место первого объекта подставляет значение «СОЛНЕЧНЫЙ», на место второго объекта по очереди подставляет все возможные (известные из фактов о RUSENG) значения второго объекта. Как только обнаружено, что синтезированный факт совпадает с одним из фактов базы знаний, решение считается найденным. Тогда текущее значение второго аргумента записывается в переменную Y и выдается на экран. После этого поиск решений (синтез и сравнение фактов) продолжается до исчерпания базы знаний, так как верных ответов может быть несколько.

Запуск и выполнение программы

Нажатие клавиши ВВОД на строке с вопросом запускает выполнение программы. После этого Пролог-Д переходит в режим «Выполнение»: экран очищается и выводятся все найденные решения. Нажатие любой клавиши возвращает в режим «Редактирование».

Напомним *описание правила* и приведем вопрос, содержащий это правило:

```
СИНОНИМ(X,Y) <- RUSENG(X,Z), RUSENG(Y,Z);
? СИНОНИМ(СОЛНЕЧНЫЙ,Y);
```

Результат выполнения программы:

```
Y=РАДОСТНЫЙ
Y=БЛЕСТЯЩИЙ
ДРУГИХ РЕШЕНИЙ НЕТ
```

Хотя для пользователя обработка фактов и правил выглядит одинаково, технически второе немного сложнее. *Описание* правила СИНОНИМ использует два факта RUSENG с разными переменными, поэтому вычисляться они должны независимо - в данном случае Пролог-Д синтезирует сразу два факта и сверяет их с фактами из базы знаний. Затем к результатам применяется логическая операция «и». На место переменной X всегда подставляется значение «СОЛНЕЧНЫЙ». На место переменных Y и Z по очереди будут подставлены все известные о RUSENG факты (согласно *описанию* правила СИНОНИМ, на место Y подставляются значения первого объекта RUSENG, на место Z - значения второго объекта RUSENG). Для каждого синтезированного набора фактов Пролог-Д вычислит логическое значение предиката СИНОНИМ и в случае получения истины сочтёт текущее значение переменной Y подходящим ответом. После чего продолжит поиск.

Базовые логические операции

Создадим предикат EQ, который с помощью переменной Z опишет эквивалентность (равенство) двух объектов:

```
EQ(Z,Z);
```

Зададим вопрос:

? EQ(3,3);

Пролог-Д ищет в программе предикат EQ и подставляет в него значения обоих аргументов (числа 3 и 3), после чего проверяет, верно ли, что оба значения можно *одновременно* представить одной и той же переменной Z. Ответ - ДА.

Теперь зададим вопрос:

? EQ(3,4);

Очевидно, значения 3 и 4 не могут быть *одновременно* представлены одной и той же переменной Z, поэтому ответ - НЕТ.

Следующий вопрос:

? NOTEQUAL(3,4);

Предикат с именем NOTEQUAL не описан в программе, Прологу-Д о нем ничего не известно. В таких случаях в Прологе принято выдавать ответ НЕТ.

Отрицание. В Прологе-Д есть встроенный предикат НЕ с одним аргументом. Дополним наше правило СИНОНИМ, чтобы слово не считалось синонимом самому себе. Для этого понадобится предикат EQ и встроенный предикат НЕ:

```
EQ(Z,Z);
СИНОНИМ(X,Y) <- НЕ (EQ(X,Y)), RUSENG(X,Z), RUSENG(Y,Z);
```

Первая цель правила СИНОНИМ гласит, что X и Y не должны быть эквивалентны. Для ускорения работы программы полезно первыми записывать простые цели. Пролог-Д просматривает цели слева направо, и как только встречает цель, выдающую ответ НЕТ, заканчивает обработку значений (и переходит к поиску других решений, если в вопросе содержатся переменные). Сложные для вычисления цели лучше записывать последними, надеясь на то, что до их обработки дело не дойдёт. Впрочем, баланс между быстроедействием и удобочитаемостью программы каждый программист определяет для себя сам.

Или. В Прологе-Д нет отдельного символа для обозначения логической операции «или». Вместо этого отношение «или» задают наполняя предикат несколькими правилами:

```
ДАЛЬНИЙРОДСТВЕННИК(X,Y) <- ТЕТЯ(X,Y);
ДАЛЬНИЙРОДСТВЕННИК(X,Y) <- ДЯДЯ(X,Y);
ДАЛЬНИЙРОДСТВЕННИК(X,Y) <- ПЛЕМЯННИК(X,Y);
```

Отсюда получается: если у человека есть тетя *или* дядя, значит у него есть дальний родственник.

Отсечение. Обозначается знаком «!». Иногда нужно прекратить дальнейший поиск решений, когда одно решение уже найдено. Для примера дополним определение СИНОНИМ ещё одной целью и зададим вопрос:

```
СИНОНИМ(X,Y) <- НЕ (EQ(X,Y)), RUSENG(X,Z), RUSENG(Y,Z), !;
? СИНОНИМ(СОЛНЕЧНЫЙ,Y);
```

Как уже было сказано ранее, Пролог-Д рассматривает цели слева направо. Когда в процессе перебора фактов X и Y принимают равные значения, предикат НЕ(EQ(X,Y)) выдает НЕТ и поиск текущих значений X и Y в словаре RUSENG не производится. Когда же первые три цели выдают значение ДА, Пролог-Д переходит к обработке четвертой цели. А в данном случае это «!» - отсечение. Предикат отсечения заканчивает обработку правила,

целью которого является. Таким образом, правило СИНОНИМ, дополненное оператором отсечения, выдаст только одно, первое найденное, решение. После этого обработка прервётся.

Цель «!» не обязательно должна стоять последней, после неё можно указать и другие цели. Пока они принимают значение ДА, обработка не прерывается. Как только хотя бы одна из целей справа от «!» примет значение НЕТ, обработка прекратится.

Важно заметить, что прекращается поиск решений только для конкретного, обрабатываемого в данный момент правила, а не для всей программы. Таким образом, предикат отсечения удобно использовать, в частности, для выхода из рекурсии.

В реализации Пролога-Д внутреннее устройство *факта* и *правила* одинаково. Факт – это правило, не зависящее ни от каких целей. Поэтому следующие две записи равнозначны:

```
МАМА (ЛЕНА, ВАНЯ) ;
МАМА (ЛЕНА, ВАНЯ) <- ;
```

Это даёт нам возможность применить к факту предикат отсечения, который пояснит, что у человека может быть только одна мама (если найдена одна, других искать не надо):

```
МАМА (ЛЕНА, ВАНЯ) <- ! ;
```

Предикат отсечения нельзя использовать в вопросе. Он допустим только в определении правила.

Арифметические операции

Вся арифметика в Прологе-Д *целочисленная*, переменные могут принимать только положительные 16-разрядные значения в диапазоне от 0 до 65535.

Для понимания арифметических вычислений в Прологе-Д рассмотрим уравнение двух переменных в форме линейной функции:

$$k * x + m = y$$

По какой-то причине страсть к математическим абстракциям привела создателей языка именно к такому виду вычислений: все базовые арифметические операции на Прологе-Д выражаются через линейную функцию. При $k=1$ мы получаем сумму x и m в переменной y . При $m=0$ мы получаем произведение k и x в переменной y . В свою очередь, k содержит результат деления y на x если $m=0$. Ну а разность y и x получается в m если $k=1$.

Для вычисления линейной функции в Прологе-Д есть предикат ВYЧ:

```
ВYЧ (K, X, M, Y) ;
```

Чтобы произвести сложение, вычитание, умножение или деление, программисту нужно составить линейную функцию с соответствующими аргументами.

Допустим, нам нужно узнать результат деления 35 на 7. Линейная функция будет выглядеть так:

$$7 * x + 0 = 35$$

Соответствующая запись на языке Пролог-Д и результат обработки вопроса:

```
? ВYЧ (7, X, 0, 35) ;
X=00005
ДРУГИХ РЕШЕНИЙ НЕТ
```

Иными словами, в Прологе-Д отсутствуют привычные операторы $+$ $-$ $*$ и все вычисления производятся с помощью предиката ВЫЧ.

Создадим более удобный предикат для возведения в квадрат переменной X. Ответ будет помещён в переменную Y:

```
QUAD(X,Y)←ВЫЧ(X,X,0,Y);
```

Иногда по какой-то причине Пролог-Д может выдать три одинаковых ответа на вопрос:

```
? QUAD(3,Y)
Y=00009
Y=00009
Y=00009
ДРУГИХ РЕШЕНИЙ НЕТ
```

Чтобы исправить эту оплошность, добавим в описание предикат отсечения:

```
QUAD(X,Y)←ВЫЧ(X,X,0,Y),!;
```

Теперь ответ будет только один.

Хотя философия языка это подразумевает, в реализации Пролог-Д мы, к сожалению не сможем узнать ответ на вопрос:

```
? QUAD(X,16);
ОШИБКА 34
НЕТ
```

Сравнение. В Прологе-Д нет встроенных средств для сравнения чисел, но их можно создать, используя лишь логику и математическую индукцию – в этом философия и суть Пролога. Помня о том, что Пролог-Д работает с положительными целыми числами, запишем правило: при сравнении с нулем любое число X выдаст ДА, если только само это число X не равно нулю:

```
БОЛЬШЕ(X,0)←НЕ(ЕQ(X,0)),!;
```

Предикат EQ должен быть определен как было показано ранее. Предикат отсечения «!» нужен для дальнейшего использования в ещё одном правиле: если $X > Y$, то верно и неравенство $X-1 > Y-1$. Будем отнимать по единице до тех пор, пока не выполнится одно из условий:

1. Второй аргумент сравнивается с нулем. Тогда сработает записанное выше правило, оно выдаст ответ ДА и остановит перебор благодаря оператору отсечения.

2. Первый объект сравнивается с нулем. Соответствующее правило опишем ниже, оно должно выдавать ответ НЕТ и также останавливать перебор.

Второе правило выглядит так:

```
БОЛЬШЕ(X,Y)←НЕ(ЕQ(X,0)),ВЫЧ(1,U,1,X),ВЫЧ(1,V,1,Y),!,БОЛЬШЕ(U,V);
```

Оно выполняется пока X не равен нулю. В переменную U записываем X-1, в переменную V записываем Y-1. Нам нужно чтобы предикат БОЛЬШЕ вычитал единицы и вызывал сам себя. Выход из рекурсии (череды самовывозов) произойдет когда X сравнивается с нулем. Тогда предикат БОЛЬШЕ справа от символа «!» выдаст НЕТ и на этом обработка закончится.

К сожалению, Пролог-Д не справится с обработкой этим методом больших чисел, так как он потребует глубокого погружения в рекурсию.

Для решения этой проблемы перейдём от вычитания единиц к делению пополам. Правда, Пролог-Д не может решить задачу деления нечётного числа на 2, ведь ответ не принадлежит множеству целых чисел. Поэтому нам придётся определить предикат HALF (половина):

$$\text{HALF}(A, B) \leftarrow \neg \text{ВЫЧ}(2, B, 0, A);$$

$$\text{HALF}(A, B) \leftarrow \neg \text{ВЫЧ}(2, B, 1, A);$$

В зависимости от чётности числа A, одно из правил выдаст ответ НЕТ, зато другое найдёт целое число. Оно и станет решением предиката HALF.

Будем производить деление пополам до тех пор, пока первый объект не станет на единицу больше второго, либо первый объект сравняется с нулем. В первом случае ответ ДА, во втором – НЕТ.

$$\text{БОЛЬШЕ}(X, Y) \leftarrow \neg \text{ВЫЧ}(1, Y, 1, X), !;$$

$$\text{БОЛЬШЕ}(X, Y) \leftarrow \neg (\text{ЕО}(X, 0) \vee \text{HALF}(X, U) \wedge \text{HALF}(Y, V) \wedge \text{БОЛЬШЕ}(U, V));$$

Операция деления выполняется дольше, чем вычитание единицы, но ответ получается за меньшее число шагов (для 16-разрядных чисел максимум за 16 рекурсивных вызовов).

Графические возможности

В Пролог-Д встроен предикат ОТР для рисования отрезков в чёрно-белом режиме 512 на 240 точек. Левому верхнему углу экрана соответствуют координаты 0,0.

$$\text{ОТР}(A, B, X, Y, 1)$$

Предикату ОТР передают 5 объектов. Первые два - горизонтальная и вертикальная координаты начальной точки. Следующие два объекта - координаты конечной точки. Пятый объект должен принимать либо значение 1 (отрисовка отрезка), либо 0 (стирание). Когда координаты начальной и конечной точки совпадают, длина отрезка равна нулю (он не рисуется).

Координата может быть одного из трёх типов:

1. Число
2. Переменная, которая не используется нигде, кроме предиката ОТР
3. Переменная, значение которой вычислено Прологом-Д

В первом случае предикат нарисует один отрезок и вернёт ДА.

Во втором случае Пролог-Д, выполняя предикат ОТР, по очереди подставит на место неопределённой переменной все возможные значения (от 0 до 511 для горизонтальной координаты, от 0 до 239 для вертикальной). Только после того, как нарисовано всё множество отрезков, предикат закончит работу. Ответ ДА.

В третьем случае, если Пролог-Д не нашёл решения для переменной, отрезок не будет нарисован, а предикат ОТР выдаст ответ НЕТ. Если решение найдено, отрезок отобразится, ответ ДА.

После нахождения каждого решения для правила, содержащего один или несколько предикатов ОТР, Пролог-Д показывает текстовый курсор и ожидает нажатия любой клавиши, затем очищает экран и печатает решение. Повторное нажатие клавиши возвращает Пролог-Д к дальнейшему поиску решений.

Примеры трёх вопросов с разными типами координат:

$$? \text{ ОТР}(0, 120, 511, 0, 1);$$

```
? ОТР(0,120,511,Y,1);
? ВЫЧ(A,2,1,511),ОТР(A,120,511,0,1);
```

В первом случае будет нарисован один отрезок. Ответ ДА.

Во втором случае отобразится множество отрезков, складывающихся в закрашенный треугольник. Ответ ДА.

В третьем случае Пролог-Д сначала вычислит значение переменной А, а затем предикат ОТР проведёт один отрезок из точки с координатами 255,120. Ответ после очистки экрана:

```
A=00255
ДРУГИХ РЕШЕНИЙ НЕТ
```

Если в третьем вопросе поменять местами предикаты ВЫЧ и ОТР, то к моменту выполнения ОТР значение переменной А ещё не будет вычислено.

```
? ОТР(A,120,511,0,1),ВЫЧ(A,2,1,511);
```

По умолчанию значение переменной считается равным нулю, поэтому отрезок будет нарисован из точки с координатами 0,120 и только после этого Пролог-Д посчитает значение переменной А.

Одновременное использование всех трёх типов объектов:

```
? ВЫЧ(A,2,1,511),ОТР(A,120,511,Y,1);
```

Из точки 255,120 будут проведены отрезки в множество точек с координатами 512,Y (где Y последовательно примет значения от 0 до 239). После появления текстового курсора, нажатия клавиши и очистки экрана будет напечатан ответ:

```
A=00255
Y=_1
ДРУГИХ РЕШЕНИЙ НЕТ
```

Для Y выводится, по-видимому, индекс этой неопределённой переменной.

Теперь создадим предикат DOT и наполним его фактами о координатах точек:

```
DOT(256,40);
DOT(142,100);
DOT(370,100);
DOT(184,190);
DOT(328,190);
```

Зададим три вопроса. Первый выдаст ответ ДА, второй - НЕТ:

```
? DOT(142,100);
? DOT(14,10);
? DOT(A,B),DOT(C,D),ОТР(A,B,C,D,1);
```

Третий вопрос приведёт к такому поведению Пролога-Д: из базы знаний будет выбрана первая пара точек (они окажутся одинаковыми) и нарисован отрезок между ними (для одинаковых точек это отрезок нулевой длины). Затем после нажатия клавиши экран очистится и появится ответ:

```
A=00256
B=00040
C=00256
```


D=00040

Нажав клавишу ещё раз, мы позволим Прологу-Д продолжить поиск решений. Из базы знаний будет выбрана следующая пара точек и нарисован отрезок между ними. Однако, на экране всё ещё останется предыдущий ответ. Раз за разом нажимая клавишу, мы будем видеть следующий отрезок, но предыдущие значения переменных A, B, C, D. Когда все возможные сочетания координат из базы знаний будут перебраны, появится ответ:

ДРУГИХ РЕШЕНИЙ НЕТ

Для того, чтобы рисовать отрезки не по одному, а все сразу, создадим правило STAR и запустим его на выполнение, задав вопрос:

```
STAR(<-DOT(A,B),DOT(C,D),OTP(A,B,C,D,1),NO());
? STAR();
```

Последний предикат NO нигде не описан, поэтому он всегда будет выдавать НЕТ, а значит и всё правило STAR тоже примет значение НЕТ. Даже найдя в базе знаний факты и нарисовав отрезок, Пролог-Д будет считать, что ответ на вопрос ?STAR не получен, поэтому не напечатает на экран значения переменных A, B, C, D и продолжит искать ответы. Когда база знаний будет исчерпана, Пролог-Д покажет текстовый курсор, а после нажатия клавиши - ответ НЕТ. Ведь предикат STAR так ни разу не принял значение ДА.

Как уже было сказано выше, правило может содержать несколько предикатов OTP, а также другие правила, которые включают предикат OTP. Это позволяет рисовать довольно сложные картины. Стирая отрезки, можно добиться даже некоторого подобия анимации.

Упражнения

1. Объясните работу программы нахождения N-го числа Фибоначчи:

```
ФИ(1,1)<-!;
ФИ(2,1)<-!;
N1(N,A)<-ВЫЧ(1,A,1,N);
N2(N,B)<-ВЫЧ(1,B,2,N);
ФИ(N,F)<-N1(N,A),N2(N,B),ФИ(A,X),ФИ(B,Y),ВЫЧ(1,X,Y,F);
? ФИ(7,F);
```

2. Напишите программу, вычисляющую факториал числа N.