Московский Авиационный Институт
(Национальный Исследовательский Университет)

Кафедра 806 «Вычислительная информатика и программирование»
Факультет: «Информационные технологии и прикладная математика»

Дисциплина: «Логическое программирование»

Реферат

«Как научить бабушку логическому программированию»

Группа:	М8О-208Б-18, №2
Студент:	М.А. Алексеева
Преподаватель:	Д.В. Сошников
Оценка:	
Дата:	

Москва, 2019

1.Введение

Быть преподавателем не просто, ведь нужно понятно и очень доступно изложить иногда очень непростой материал, используя разные методики обучения. Так мне предстоит примерить эту роль на себя, нужно определиться в каком направлении двигаться. Обучать (или мучить), будем одного человека, поэтому остановимся на индивидуальном обучении, по возможности заменяя сложные термины и примеры , на более простые и житейские.

Прежде чем приступить к увлекательному образовательному процессу, нужно найти подходящего ученика, иначе кого мы будем учить? Под «подходящим» я имею ввиду то, что все бабушки абсолютно разные, никто в этом и не сомневается. Так вот, бабушки страдающие тяжелыми заболеваниями, связанными тем или иным образом с мозговой деятельностью и памятью, такими как болезнь Альцгеймера и др., нам не подходят, так как обучение превратится в пытку как для «преподавателя», так и для «ученицы».Поэтому возьмем среднестатистическую мудрую женщину лет 60-ти, обогащенную ценным опытом, с хорошим здоровьем и удивительным желанием научиться логическому программированию, например Галину Петровну, и можно начинать первый урок.

2.Основы

Взглянем на программирование и на его языки как на очень широкую сферу, выделяющую в себе несколько направлений или парадигм :

- **структурное**: программа разбивается на блоки — подпрограммы (изолированные друг от друга), а основными элементами управления являются последовательность команд, ветвление и цикл.
- **объектно-ориентированное**: задача моделируется в виде объектов, которые отправляют друг другу сообщения. Объекты обладают свойствами и методами.
- **функциональное**: базовым элементом является функция и сама задача моделируется в виде функции, а, точнее, чаще всего в виде их композиции, когда аргументом одной функции становится другая.
- **логическое**: эту парадигму сформулируем в конце обучения, как его итог и успешный результат.

Для того чтобы Галина Петровна понимала для чего ей нужны знания логического программирования , нужно описать его возможности. В языке логики есть мощные механизмы, позволяющие рассматривать данный язык как средство высокого уровня, имеющие реализацию в следующих сферах:

- Программирование искусственного интеллекта (ИИ).
- Анализ естественного языка
- Представление и обработка знаний
- Компьютерная алгебра
- Символьные вычисления и тд.

В основе логического программирования (лп) лежит логика предикатов первого прядка. Таким образом для решения какой-либо задачи необходимо описать ее на подмножестве предикатов (правил), на которые будет опираться программа или другой

логический интерпретатор, целью которого является автоматический вывод ответа на поставленную задачу.

Любое погружение в новую неизвестную тему, можно представить как погружение в море, сейчас Галина Петровна, наша ученица стоит на берегу, но уже готова сделать первый шаг. Этим первым шагом будут знания об декларативном и императивном стиле программировании, умение их отличать.

3. Декларативное и императивное программирование

Галина Петровна перед уроком принесла мне очень вкусных пирожков, приготовленных по ее фирменному рецепту, им она тоже поделилась, поэтому используем его для наглядного примера.

Императивный стиль

Для этого стиля программирования очень важно описание **как** достичь результата, каждый его шаг. Например:

- Приготовить все ингредиенты
- Насыпать муку в кастрюлю
- Разбить в кастрюлю яйца
- Добавить соль
- Добавить молока

…

Декларативный стиль

Такой стиль, в котором описывается, **какой именно** результат вам нужен. То есть получатель принимает сообщение, например:

- Приготовить пирожки.

И уже сам разбирается, какие шаги для этого надо предпринять.

Но одними житейскими примерами сыт не будешь, особенно когда растишь будущего специалиста в области лп, Галина Петровна оценила метод работы с ней, продолжаем обучение. Разберем отличие двух стилей на несложной задаче.

Необходимо вывести факториал числа F .

Императивный	Декларативный

function(n:integer): integer ;
var i, f : integer ;
begin
  f :=1;
  for i:=2 to n do f := f*i;
  fact := f;
end;
	fact(0,0)
	fact(N,F) :- N1 is N-1,
	  fact(N1,F1) , F is F1*N
	fact(N) = F:-
	  (N=0 → F=1; F is fact(N-1)*N)

Пример декларативного стиля иллюстрирует почти само определение факториала, а также не требует описания каких либо алгоритмических особенностей вычисления (выделение переменной для накопления результата умножения на очередное целое число).

Стоит отметить что этот стиль даже не формулирует факториал как произведение последовательных целых чисел.

Например нам необходимо узнать чему равен пять факториал (5!) с помощью программы, описанной выше в декларативном стиле, необходимо найти значение X, для которого верно утверждение fact(5,X). Запрос на выполнение задачи на языке Пролог выглядит следующим образом:

```
?-fact(5,X)
X=120
```

Галина Петровна, оказалась на удивление очень сообразительной и внимательной и предложила свой метод решения, который является полу-декларативным в императивном программировании стилем. Решение с использованием рекурсивной процедуры :

```
Function fact(n: integer) : integer ;
begin
    if n = 1 then fact := 1;
    else fact := fact(n-1)*n;
end;
```

Рассмотрев этот пример решения у Галины Петровны возник вполне логичный вопрос, где же проходит грань между методами ? Все языки программирования высокого уровня позволяют писать в смешанном стиле и, на самом деле, четкого разделения даже эти два примера не показывают. На мой взгляд, правильнее будет называть один стиль «более декларативным», а другой — «более императивным».

Мы немного узнали про стили программирования , настало время узнать про языки и их историю, которые эти стили используют.

4. Языки программирования и их история

Совершенно очевидно, что программирование и его принципы, какими их мы сейчас представляем, не могли возникнуть раньше, чем появление в мире ЭВМ. Поэтому программирование не смотря на то что опирается на такие древние предметы как математика и логика, относительно молодое явление. Началом зарождения программирования считается середина XX века , но основы были заложены ещё около века назад.

- Середина XIX века Чарльз Беббидж и Ада Лавлейс разработали принцип императивного программирования,представляющий набор команд, который послужил фундаментом для современных ЭВМ и определил доминирование одного из стилей.
- Первые ЭВМ появились в 1940-х годах и программировались с помощью машинных языков. Машинный код состоял из последовательностей нулей и единиц. Каждая элементарная операция имела свой код, необходимо было явно указывать адреса ячеек памяти, в которых хранились данные, или куда их необходимо было сохранять.
- В начале 1950-х годов была осуществлена идея использования символьных имен вместо адресов данных и замены цифровых кодов операций на мнемонические (словесные) обозначения. Язык программирования, реализующий данный подход,

получил название Ассемблер. Программа, записанная на Ассемблере, не может обрабатываться непосредственно процессором. Возникла необходимость преобразования текста программы, записанной на данном языке, в машинный код. Для решения этой задачи были созданы трансляторы.

- Чуть позже, после ассемблера появились три важнейших языков программирования высокого уровня, которые повлияли на дальнейшее развитие всех сфер, хоть как-то связанных с компьютерными технологиями. Этими языками были fortran, COBOL и Lisp.
- Появление императивных языков C, Pascal, в которых было введено понятие типов данных, распределения памяти для переменных и оператора присваивания, также в 1970-х годах появился декларативный язык программирования Prolog.
- Всем известный объектно-ориентированный язык программирования C++ появился в 1980-х годах и до сих пор пользуется большой популярностью.
- 1990-ые, появление таких языков как: Java, Haskell, C#, Mercury. Язык Mercury- более усовершенствованный потомок Prologa и на сегодня считается наиболее декларативным из всех существующих языков.

На этом месте, можно сказать что Галина Петровна сделала первый шаг в море под названием «Логическое программирование», но останавливаться не стоит, поэтому смело идем дальше.

5. Системы логического программирования

Как мы помним логическое программирование основана на языку предикатов и включает в себя описание условий задачи и свойств предметной области на этом языке. Поэтому выполнение программы сводится к определению «выводимости» того или иного заключения из логического описания задачи.

Итак разберем стандартную задачу на языке Prolog, а потом прокомментируем ее, чтобы у Галины Петровны не осталось вопросов.

```
studied (masha,mathematics).                                                    (1)
studied (masha, compscience).
studied (masha,english).
studied (sasha,german).
studied (sasha,literature).

studied_technical(X) :- studied (X,mathematics).                                (2)
studied_technical(X) :- studied (X,compscience).

studied_languages(X) :- studied (X, english).                                   (3)
studied_languages(X) :- studied (X, german).

speciality(X, tech_translator) :- studied_technical(X), studied_languages(X) .  (4)
speciality(X, programmer) :- studied(X, mathematics), studied(X, compscience) .
speciality(X, lit_translator) :- studied(X, literature), studied_languages(X) .
```

Данная программ представляет собой некую базу данных студентов. В первом блоке программы, сказано, что Маша учит три предмета : математику, английский и информатику, а Саша — немецкий и литературу. Это набор фактов, описывающие отношения объектов ы предметной области. В данном случае studied— название предиката, а masha, mathematics, compscience, english, german, sasha, literature— некоторые константы или как правильно называть «атомы», соответствующие объектам в предметной области.

Галина Петровна опять проявила свою наблюдательность и спросила, почему имена студенток Маши и Саши написаны в программе с маленькой буквы, ведь это как-то неправильно. К сожалению Галину Петровну пришлось огорчить и сказать что,написание верное, потому что если бы написание имен выглядело по-другому и было бы написано с большой буквы, интерпретатор воспринял бы идентификатор Masha, как переменную , что абсолютно не правильно.

Следующие блоки (второй, третий и четвертый) представляют собой правила, позволяющие на основе фактов логические выводы. В четвертом блоке в первой строке, говорится о том, что некоторый студент X может стать техническим редактором, если он изучал технические и языковые предметы. Знак « :- » можно перевести как «если», и означает он следствие или импликацию с точностью до порядка аргумента, а « , » означает коньюнкцию и переводится как « и ». Совокупность правил (второй и третий блок) описывают дизьюнкцию « или ». Таким образом, студент X изучал технические предметы , если он изучал математику ИЛИ информатику, аналогично с языковыми предметами.

Чтобы программа запустилась необходимо ввести запрос в следующем виде:

```
?- speciality (masha, programmer).
yes
?- speciality (sasha, programmer).
no
?- speciality (katya, lit_translator).
no
```

Курсивом выделены запросы, вводимые пользователем, жирным шрифтом выделены ответы системы. Первый запрос-ответ демонстрирует , что Маша может стать программистом, а второй , о невозможности вывести то же утверждение о Саше.

Более интересный третий запрос о несуществующей студентке в базе данных, система выводит ответ no, но это вовсе не значит, что утверждение ложно, а всего лишь говорит о том что утверждение невыводимо из содержащихся в логической программе операций.

6. Заключение

К сожалению, невозможно уместить весь курс логического программирования в реферат, ограниченный страницами. А так хотелось вырастить из Галины Петровны хорошего специалиста, но начало положено. Сейчас как никогда кстати уместно наше сравнение обучения с погружением в море, пора отправить нашу ученицу в «свободное плавание». Мы заложили фундамент и как я надеюсь еще больше подогрели интерес к логическому программированию, теперь Галина Петровна не будет чувствовать себя неуверенно посреди океана, а смело поплывет дальше, впрочем как и все мы.