УДК [378.016:796]:004.9 DOI 10.37972/chgpu.2020.69.53.026

И. А. Обломов[2], В. В. Ржавин[2], В. И. Краснов[2], К. Н. Фадеева[1]

ПРОБЛЕМЫ ПРЕПОДАВАНИЯ ДЕКЛАРАТИВНЫХ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ

[1]*Чувашский государственный педагогический университет им. И. Я. Яковлева, г. Чебоксары, Россия*

[2]*Чувашский государственный университет имени И. Н. Ульянова, г. Чебоксары, Россия*

Аннотация. Статья посвящена анализу проблем организации обучения логическому и функциональному языкам программирования и путей их решения. Авторами рассмотрены взгляды исследователей на изучение декларативных языков программирования. Обозначены актуальные проблемы преподавания декларативных языков программирования в высшей школе. Решение этих проблем существенно повышает уровень подготовленности будущего специалиста, расширяет его кругозор и совершенствует его навыки разработки сложных программных проектов в различных областях, в том числе в искусственном интеллекте.

Ключевые слова: *декларативные языки программирования, процедурный смысл и декларативная семантика, функциональный аргумент, функции более высокого порядка, S-выражение.*

I. A. Oblomov[2], V.V. Rzhavin[2], V. I. Krasnov[2], K. N. Fadeeva[1]

TEACHING DECLARATIVE PROGRAMMING LANGUAGES

[1]*I. Yakovlev Chuvash State Pedagogical University, Cheboksary, Russia*

[2]*I. Ulyanov Chuvash State University, Cheboksary, Russia*

Abstract. The article is devoted to the analysis of problems of organization of teaching logical and functional programming languages and the ways for the solution of those problems. The authors considered the views of the researchers on the study of declarative programming languages; identified the current problems of teaching declarative programming languages at universities; stated that the solution of the discussed problems increases the level of readiness of future specialists, expands their outlook and develops the skills of development of complex programming projects in various fields, including artificial intelligence.

Keywords: *declarative programming languages, procedural meaning and declarative semantics, functional argument, higher-order functions, S-expression.*

Актуальность исследуемой проблемы. Современные курсы информатики, преподаваемые практически во всех высших учебных заведениях страны, обладают целым рядом особенностей, как правило, не характерных для других дисциплин. С одной стороны, учитывая относительную молодость информатики как учебной дисциплины, а также постоянное совершенствование информационных и телекоммуникационных технологий, содержание учебных курсов, относимых к информатике, не фиксировано. Большинство педагогов, работающих со студентами, используют для преподавания собственные авторские программы, содержание которых может достаточно сильно отличаться. С другой стороны, все многообразие и разрозненность учебных программ по информатике компен-

сируются практически повсеместным выделением во всех учебных курсах содержательных блоков, связанных с технологиями разработки алгоритмов и принципами их кодирования на нескольких популярных языках программирования [10]. Благодаря массовости обучения программированию большинство выпускников вузов имеют определенные навыки программирования. Однако несмотря на это, современный рынок труда испытывает острую нехватку специалистов, способных решать насущные информационные проблемы наиболее эффективными и экономичными методами, особенно связанные с использованием декларативных или неалгоритмических языков программирования. В связи с этим научные исследования, нацеленные на изучение проблем обучения декларативным языкам программирования и путей их решения, до сих пор остаются актуальными.

Целью статьи является выявление проблем преподавания декларативных языков программирования в вузе, а также описание путей их решения. ФГОС 3++ позволяет убрать расхождение между качеством подготовки выпускника учебными учреждениями и требованиями, предъявляемыми к ним работодателями [5]. Формирование требований к результатам освоения основных образовательных программ профессионального образования в части профессиональной компетенции осуществляется на основе соответствующих профессиональных стандартов [4].

Материал и методика исследований. Теоретико-методологическую основу нашего исследования составили труды о методологических аспектах преподавания декларативных языков программирования, раскрывающие категории «логическое программирование», «функциональное программирование», «операторное программирование». Для освещения основных аспектов по теме были использованы теоретический анализ проблемы и предмета исследования, а также научно-педагогический опыт преподавателей.

Результаты исследований и их обсуждение. При обучении бакалавров третьего курса направления подготовки 09.03.01 Информатика и вычислительная техника возникает ряд проблем, связанных с преподаванием им декларативных или неалгоритмических языков программирования, к которым можно отнести язык логического программирования Prolog, язык функционального программирования Lisp. Их главной особенностью является отсутствие какого-либо алгоритма при написании программ [6], то есть заранее определенной последовательности действий, присущей известным алгоритмическим языкам, таким как Pascal, C и их разновидностям. Это является, пожалуй, первой и основной причиной сложного начального восприятия обучающимися декларативных языков. Им нередко бывает трудно понять, как можно написать программу без использования операторов структурного программирования, таких как операторы ветвления, цикла, присваивания, передачи управления. В декларативных языках программирования понятие оператора встречается крайне редко, например, при вычислении арифметических выражений, что им не свойственно. Здесь предполагается обработка символьной информации [2], [6].

Основа программы на Prolog (Пролог) строится на определении некого конечного множества объектов и отношений между ними в виде последовательности фактов (истинных утверждений) и правил, которые доказуемы при определенных условиях. Классическим примером является определение родственных отношений между группой людей:

```
Parent (‘Ivan’,’Petr’).
Parent (‘Maria’,’Petr’).
Parent (‘Petr’,’Sergey’).
man (‘Ivan’).
man (‘Petr’).
woman (‘Maria’).
```

Эти отношения, описанные в виде простых фактов, обычно воспринимаются без особых проблем, поскольку соответствуют естественному пониманию отношений между людьми. Факты языка Пролог часто ассоциируют с аксиомами в математике. Несколько сложнее понимается попытка описания ориентированного графа без циклов как множества ребер, ведущих из вершины в вершину: edge(a,b). edge(a,s). edge(s,r). edge(b,w).

Несложно заметить аналогию с отношениями parent, только вместо имен людей здесь фигурируют названия абстрактных вершин, и проблема непонимания решается, как правило, путем совместной визуальной демонстрации дерева родственных отношений и дерева графа.

Дальнейшие проблемы могут возникнуть при попытке описания правил, которые требуют логического вывода, доказательства. Правила Пролога подобны теоремам, которые доказываются на основе простых фактов и других правил.

Расширить исходную программу родственных отношений можно введением правил типа mother и father:

mother (A, B): - woman (A), parent (A, B). father (A, B): - man (A), parent (A, B).

Интерпретация этих простых правил очевидна: любой объект А является матерью (отцом) по отношению к объекту В, при условии, что объект А является женщиной (мужчиной) и выполняется цель parent (A, B).

Сложность составления правил состоит в первую очередь в том, что цели, идущие в правой части правил, перечисляются через символ ‘,’ (запятая), который в обычных языках используется обычно для разделения составных компонент какого-либо объявления, например, перечисляемого типа данных, или при описании параметров функции. В Прологе запятая выполняет логическую операцию «И», что может быть большой неожиданностью для только начинающих программировать на этом языке. Большую неожиданность может вызвать операция связывания целей по «ИЛИ», обозначаемая символом ‘;’ (точка с запятой). Для большинства программистов с «алгоритмическим стилем мышления» точка с запятой означает завершение оператора или, как в языках ассемблера, начало комментариев. Приобретенные ранее привычки на когнитивном уровне могут сыграть плохую роль, поскольку ошибка типа «рука не туда пошла» составляет большую часть всех ошибок в первых программах на Прологе.

Следующая проблема языка связана с понятием и использованием переменных. Переменная в Прологе всегда начинается с заглавной буквы, например, A, B, C в вышеприведенных правилах. Это не сложно, к этому надо просто привыкнуть. «Подвох» переменных кроется в другом, в частности, в том, что лексический диапазон их действия распространяется только на то предложение, в котором они участвуют. У многих обучающихся возникает желание «протащить» переменную через всю программу, как это принято в алгоритмических языках, то есть объявить ее как глобальную. Причина подобного желания кроется в том, что большая часть переменных в алгоритмических языках предназначена для хранения числовых величин, областью действия которых является локальный блок или целиком вся программа. В Прологе числа используются крайне редко, здесь происходит обработка символьной информации: списков, деревьев, графов и прочих рекурсивных структур. Хранение глобальных объектов в логических программах не используется, а если в период написания программы возникает потребность в этом, то необходимо сменить стиль мышления. Очень часто этот процесс требует значительных временных затрат, поскольку алгоритмический стиль решения задачи невозможно отменить одним волевым усилием.

Дальнейшие проблемы связаны с рекурсивными правилами, которые можно считать одной из основ логического и функционального программирования. Дело в том,

что рекурсивному стилю программирования уделяется мало внимания при изучении языков программирования. Классические примеры рекурсии в алгоритмических языках – вычисление факториала или чисел Фибоначчи – являются самыми неудачными для изучения рекурсии. В Прологе используются по-настоящему рекурсивные структуры данных – списки, деревья и пр.

Примером рекурсивного отношения может быть правило ancestor (предок), которое позволит выявить всех предков в родственных отношениях:

Ancestor (X, Y): - parent (X, Y).
Ancestor (X, Y): - parent (X, Z), ancestor (Z, Y).

Первое правило определяет самого близкого предка, то есть родителя. Это граничное условие или условие, при котором рекурсия заканчивается. Если это правило опустить, рекурсия зайдет в бесконечный цикл. Второе правило определяет рекурсивное определение отношения, поскольку в определении в правой части правила используется это же отношение ancestor. Важным условием определения рекурсивного отношения являются аргументы рекурсивного вызова. Они должны отличаться от исходных аргументов, иначе произойдет переполнение программного стека.

По аналогии с отношением *предок*, можно определить связность двух вершин в произвольном ориентированном графе:

Connected (Node, Node).
Connected (Node1, Node2): - edge (Node1, Link), connected (Link, Node 2).

В первом предложении говорится о том, что каждая вершина произвольного графа связана сама с собой. Это есть условие выхода из рекурсии.

Второе правило утверждает, что произвольная вершина Node 1 связана с вершиной Node 2 при условии, что из вершины Node 1 ведет ребро в вершину Link, которая в свою очередь связана с вершиной Node 2. Несложно заметить, что это определение практически полностью совпадает с определением отношения *предок*.

Сопоставление этих отношений позволяет лучше понять сущность отношения связности вершин. Если отношение *предок* не вызывает никаких затруднений у обучающихся, то связность вершин в графе не всегда ими понимается, поскольку подобное определение не вписывается в рамки алгоритмической парадигмы программирования, что является естественной преградой в период освоения предмета. Одним из путей решения обозначенной проблемы может быть углубленное изучение курсов теории рекурсивного программирования, математической логики, структур и алгоритмов данных.

Любая программа, в том числе и Пролог, представляет множество предложений, основанных на синтаксисе языка программирования. Перестановка порядка следований предложений в алгоритмических языках меняет семантику программы и приводит к искажению ожидаемого результата. В декларативных языках программирования различают два смысла программы – декларативный и процедурный. Первый касается исключительно отношений между объектами и отвечает на вопрос, что необходимо конкретной программе сделать, не указывая, каким образом этого можно достичь. Перестановка предложений в программе не влияет на конечный результат. Второй смысл в большей степени отвечает на вопрос, каким образом был достигнут конечный результат. Порядок следования предложений и целей в них несет тот же смысл, что и в процедурном программировании. Примером этого является рекурсия, где порядок следования предложений должен быть задан очень жестко. В первую очередь должна быть осуществлена проверка на граничное условие или условие выхода из рекурсии, после чего описываются рекурсивные предложения. Причиной частых ошибок является нарушение порядка целей, что приводит к зацикliва-

нию вычислительного процесса. Эти несложные правила запоминаются быстро, однако их невыполнение, например, по невнимательности, часто приводит к курьезным решениям.

При программировании на Прологе большее внимание следует уделять декларативному смыслу, процедурная семантика возлагается на Пролог-систему.

Другой язык, не относящийся к алгоритмическим, – Lisp (Лисп). Его основная задача – обработка списков, что кроется в его названии. Аббревиатуру Lisp следует расшифровывать как list programming (программирование списков). Здесь, как и в Прологе, математические вычисления используются крайне редко.

Лисп-программы представляют собой практическую реализацию теоретических разделов программирования, в частности, теории рекурсивных программ [3], [7], [10]. Функциональный подход, в котором кроме вызова функции нет никаких других действий, нет операторов присваивания, циклов, передачи управления, приводит в первый момент к невосприятию студентами языка Lisp. Дальнейшие трудности его освоения связаны с понятиями *лямбда-функции* и *функции более высокого порядка*, то есть функционалов, и их практическим применением. Понятие функционального аргумента, отсутствующее в процедурных языках, вносит сложность в программу.

Теория лямбда-исчисления, предложенная А. Черчем, послужила теоретической основой языка Лисп. Лямбда-функции языка – это безымянные функции, действующие в контексте их определения [3], [8]. Их часто используют в качестве аргументов других функций, что позволяет определять функции более высокого порядка. Подобные возможности отсутствуют в алгоритмических языках, в которых в качестве аргументов функций могут выступать только объекты данных, что приводит программистов к некоторому замешательству.

Частой причиной ошибок в программах на языке Лисп является специфическая форма записи программ в виде S-выражений. Эта форма предполагает использование скобок, число которых может быть очень большим. Количество скобок и их местоположение в выражении влияет на его смысл и конечный результат. Типовое S-выражение, вычисляющее объединение двух списков через лямбда-выражение, представлено ниже:

```
((lambda(lst_1 lst_2) (append lst_1 lst_2)) '(a b) '(c (e r) f)).
```

Даже по этому простому выражению несложно понять, что наличие скобок является для программ на Лиспе определяющим моментом. Контроль скобок и их местоположения становится трудной задачей для обучающихся. Большая часть ошибок связана именно с этой причиной. Проблема решается достаточно просто: путем использования внешнего редактора текста, имеющего встроенные средства контроля за количеством и парностью скобок, подсвечивающие парные открывающие и соответствующие им закрывающие скобки. Перенос текста в интерпретатор, как правило, не вызывает сложностей. Как показывает практика, применение внешних программных средств студентами отпадает после первого месяца активного их использования.

Отдельно необходимо отметить малое количество публикаций по данной тематике.

Обозначенные особенности в ряде случаев могут стать серьезной проблемой в изучении и освоении декларативных языков программирования. Решение этих проблем позволит будущему программисту получить в пользование мощный инструмент для создания интеллектуальных систем [9].

Многие понятия и принципы декларативного языка программирования, такие как *отсутствие алгоритма, локальность имен переменных, отсутствие привычных операторов*, формируются у обучаемых по мере составления программ, в течение некоторого времени, пока идет процесс естественного привыкания к новому синтаксису языка. Другие проблемы, среди которых процедурный смысл и декларативная семантика, необхо-

димые для понимания программ, требуют более детальной проработки. Во многих случаях помогает индивидуальный подход, применение простых и наглядных примеров. Хорошую поддержку оказывают Internet-средства [10], дающие возможность следить за развитием и расширением декларативных языков программирования. В частности, в последнее время наблюдается интеграция языков с визуальными средствами, например, в Visual Prolog [1], [7] и Visual Lisp имеются средства, поддерживающие операторное программирование, что позволяет осуществить более «мягкий» переход к декларативным языкам.

Резюме. Эффективное решение существующих проблем преподавания декларативных языков программирования в высшей школе существенно повышает уровень подготовленности будущего специалиста, расширяет его кругозор и совершенствует его навыки разработки сложных программных проектов в различных областях, в том числе в искусственном интеллекте.

ЛИТЕРАТУРА

1. *Адаменко А. Н., Кучуков А. М.* Логическое программирование и Visual Prolog. – СПб. : БХВ-Петербург, 2003. – 992 с.

2. *Душкин Р. В.* Функциональное программирование на языке Haskell : учебное пособие. – М. : ДМК Пресс, 2016. – 608 с.

3. *Ездаков А. Л.* Функциональное и логическое программирование : учебное пособие. – М. : БИНОМ. Лаборатория знаний, 2009. – 119 с.

4. *Копышева Т. Н., Митрофанова Т. В., Фадеева К. Н.* Применение проектного метода при обучении бакалавров прикладной информатики в рамках реализации компетентностного подхода // Вестник Чувашского государственного педагогического университета им. И. Я. Яковлева. – 2018. – № 4(100). – С. 185–192.

5. *Лавина Т. А., Фадеева К. Н.* Содержание базовой подготовки бакалавров сервиса к использованию информационных и коммуникационных технологий // Современные проблемы науки и образования. – 2016. – № 4 [Электронный ресурс]. – Режим доступа : http://www.science-education.ru/ru/article/view?id=24877.

6. *Марков В. Н.* Современное логическое программирование на языке Visual Prolog 7.5 : учебник. – СПб. : БХВ-Петербург, 2016. – 544 с.

7. *Обломов И. А.* Методологические аспекты преподавания декларативных языков программирования // Информатика и вычислительная техника : сборник научных трудов. – Чебоксары, 2016. – С. 138–142.

8. *Практика* функционального программирования [Электронный ресурс]. – Режим доступа : http://fprog.ru.

9. *Рассел С., Норвит П.* Искусственный интеллект: современный подход. – М. : Издательский дом «Вильямс», 2006. – 1408 с.

10. *Функциональное* и логическое программирование : методические указания к лабораторным работам / сост. И. А. Обломов. – Чебоксары : Изд-во Чуваш. ун-та, 2007. – 88 с.

Статья поступила в редакцию 18.11.2019

REFERENCES

1. *Adamenko A. N., Kuchukov A. M.* Logicheskoe programmirovanie i Visual Prolog. – SPb. : BHV-Peterburg, 2003. – 992 s.

2. *Dushkin R. V.* Funkcional'noe programmirovanie na yazyke Haskell : uchebnoe posobie. – M. : DMK Press, 2016. – 608 s.

3. *Ezdakov A. L.* Funkcional'noe i logicheskoe programmirovanie : uchebnoe posobie. – M. : BINOM. Laboratoriya znanij, 2009. – 119 s.

4. *Kopysheva T. N., Mitrofanova T. V., Fadeeva K. N.* Primenenie proektnogo metoda pri obuchenii bakalavrov prikladnoj informatiki v ramkah realizacii kompetentnostnogo podhoda // Vestnik Chuvashskogo gosudarstvennogo pedagogicheskogo universiteta im. I. Ya. Yakovleva. – 2018. – № 4(100). – S. 185–192.

5. *Lavina T. A., Fadeeva K. N.* Soderzhanie bazovoj podgotovki bakalavrov servisa k ispol'zovaniyu informacionnyh i kommunikacionnyh tekhnologij // Sovremennye problemy nauki i obrazovaniya. – 2016. – № 4 [Elektronnyj resurs]. – Rezhim dostupa : http://www.science-education.ru/ru/article/view?id=24877.

6. *Markov V. N.* Sovremennoe logicheskoe programmirovanie na yazyke Visual Prolog 7.5 : uchebnik. – SPb. : BHV-Peterburg, 2016. – 544 s.

7. *Oblomov I. A.* Metodologicheskie aspekty prepodavaniya deklarativnyh yazykov programmirovaniya // Informatika i vychislitel'naya tekhnika : sbornik nauchnyh trudov. – Cheboksary, 2016. – S. 138–142.

8. *Praktika* funkcional'nogo programmirovaniya [Elektronnyj resurs]. – Rezhim dostupa : http://fprog.ru.

9. *Rassel S., Norvit P.* Iskusstvennyj intellekt: sovremennyj podhod. – M. : Izdatel'skij dom «Vil'yams», 2006. – 1408 s.

10. *Funkcional'noe* i logicheskoe programmirovanie : metodicheskie ukazaniya k laboratornym rabotam / sost. I. A. Oblomov. – Cheboksary : Izd-vo Chuvash. un-ta, 2007. – 88 s.

The article was contributed on November 18, 2019

Сведения об авторах

Обломов Игорь Александрович – кандидат технических наук, доцент кафедры вычислительной техники Чувашского государственного университета имени И. Н. Ульянова, г. Чебоксары, Россия; e-mail: ra4yes@rambler.ru

Ржавин Вячеслав Валентинович – кандидат технических наук, доцент кафедры вычислительной техники Чувашского государственного университета имени И. Н. Ульянова, г. Чебоксары, Россия; e-mail: grzhavv@gmail.com

Краснов Виталий Иванович – магистрант кафедры вычислительной техники Чувашского государственного университета имени И. Н. Ульянова, г. Чебоксары, Россия; e-mail: efrKras@gmail.com

Фадеева Клара Николаевна – кандидат педагогических наук, доцент кафедры информатики и информационно-коммуникационных технологий Чувашского государственного педагогического университета им. И. Я. Яковлева, г. Чебоксары, Россия; e-mail: fadeevakn@mail.ru

Author information

Oblomov, Igor Aleksandrovich – Candidate of Technical Sciences, Associate Professor of the Department of Computer Engineering, I. Ulyanov Chuvash State University, Cheboksary, Russia; e-mail: ra4yes@rambler.ru

Rzhavin, Vyacheslav Valentinovich – Candidate of Technical Sciences, Associate Professor of the Department of Computer Engineering, I. Ulyanov Chuvash State University, Cheboksary, Russia; e-mail: grzhavv@gmail.com

Krasnov, Vitaly Ivanovich – Master's Degree Student, Department of Computer Engineering, I. Ulyanov Chuvash State University, Cheboksary, Russia; e-mail: efrKras@gmail.com

Fadeeva, Klara Nikolaevna – Candidate of Pedagogics, Associate Professor of the Department of Informatics and Information and Communication Technologies, I. Yakovlev Chuvash State Pedagogical University, Cheboksary, Russia; e-mail: fadeevakn@mail.ru