И. А. Кудрявцева,

Российский государственный педагогический университет им. А. И. Герцена, Санкт-Петербург,

А. Г. Ракитин,

Российский морской регистр судоходства, Санкт-Петербург

ЭЗОТЕРИЧЕСКИЕ ЯЗЫКИ ПРОГРАММИРОВАНИЯ И ИХ КЛАССИФИКАЦИЯ

Аннотация

Авторами приведена классификация эзотерических языков программирования и описаны границы эзотерической парадигмы теоретического программирования. Выделены языки, которые можно отнести к «математически обоснованным» эзотерическим языкам программирования и использовать в процессе обучения теоретическому программированию в вузе для исследования способов описания вычислимости.

Ключевые слова: эзотерические языки программирования, классификация эзотерических языков программирования, эзотерическая парадигма теоретического программирования.

Еще Я. А. Коменским было сформулировано «золотое для учащихся правило: все, насколько можно, предлагать чувствам; именно: видимое — зрению, слышимое — слуху, обоняемое — обонянию, вкушаемое — вкусу, осязаемое — осязанию; а если что может быть одновременно воспринимаемо несколькими чувствами, то должно одновременно преподнести нескольким чувствам» [1, с. 349].

В этом «золотом» правиле дидактики обращается внимание на сформулированный призыв к восприятию изучаемого учебного предмета всеми органами чувств: что можешь показать — покажи, что можешь дать послушать — дай послушать, дай потрогать и т. д. (разумеется, в определенной последовательности). И те, кто шли за Коменским в дидактике как педагогике, обращенной вовне, в объективную реальность, пытались найти способ как можно более точного предъявления учащимся конкретных предметов, явлений, процессов.

Воспользуемся «золотым» правилом дидактики как теоретическим основанием для поиска некоторой совокупности языков для обучения программирова-

нию (в школе и вузе) в качестве предмета и средства обучения.

Вначале отметим, что классические языки, повсеместно используемые в процессе обучения, иногда называют *экзотерическими языками* (от гр. *exsoterikos* — внешний, общепринятый, открытый).

Мы же рассмотрим так называемые *эзотерические языки программирования*. Прилагательное в названии происходит от гр. *esoterikos* — внутренний, тайный, скрытый, предназначенный исключительно для посвященных [1].

Текст программы на любом эзотерическом языке понятен лишь «посвященному» либо непонятен вообще (иногда для конструирования программы на эзотерическом языке необходимо написать программу на экзотерическом языке, генерирующую требуемую программу).

В то время как разработчики экзотерических языков программирования стараются сделать синтаксис максимально понятным, а программирование — удобным, создатели эзотерических языков обычно ставят перед собой противоположные задачи.

Контактная информация

Кудрявцева Ирина Андреевна, канд. пед. наук, доцент, доцент кафедры компьютерной инженерии и программной инженерии Института компьютерных наук и технологического образования Российского государственного педагогического университета им. А. И. Герцена, Санкт-Петербург; *адрес:* 191186, г. Санкт-Петербург, наб. реки Мойки, д. 48; *телефон:* (812) 571-44-61; *e-mail:* yarina22@gmail.com

Ракитин Александр Георгиевич, канд. пед. наук, гл. специалист Российского морского регистра судоходства, Санкт-Петербург; *адрес:* 191186, г. Санкт-Петербург, Дворцовая наб., д. 8; *телефон:* (921) 931-84-48; *e-mail:* funcall@rambler.ru

I. A. Kudryavtseva,

The Herzen State Pedagogical University of Russia, Saint Petersburg,

A. G. Rakitin,

Russian Maritime Register of Shipping, Saint Petersburg

ESOTERIC PROGRAMMING LANGUAGES AND THEIR CLASSIFICATION

Abstract

The classification of esoteric programming languages is shown. The borders of esoteric theoretical programming paradigm are described. The languages, that can be classified as "mathematically justified" esoteric programming languages are in focus. The discussion of using it in theoretical programming learning at the university level to study ways of computability description is performed.

Keywords: esoteric programming languages, classification of esoteric programming languages, esoteric paradigm of theoretical programming.

Наконец, некоторые эзотерические языки умышленно ограничены по вычислительной мощности (например, HQ9+), другие являются полными по Тьюрингу, т. е. универсальными языками (напомним, что язык программирования называется *полным по Тьюрингу*, если его синтаксис позволяет имитировать машину Тьюринга или другую эквивалентную вычислительную модель).

Распространено мнение, что эзотерические языки:

- разрабатываются в качестве шутки, забавы или средства для гимнастики ума (или проверки программистского интеллекта); например, М. Сондерс пишет, что они «не созданы для величия, но все равно волнуют умы. Забавно попробовать Spaghetti хотя бы для того, чтобы взять да и усложнить простые вещи — это тренирует мозг, а после головоломного Malbolge вы вернетесь к программированию на старом добром C с новым энтузиазмом, как скороход, сбросивший с ног гирию» [5, с. 45];
- разрабатываются в качестве интеллектуального вызова для программиста;
- не предназначены для «серьезного» использования;
- могут быть использованы в процессе обучения экзотерическому программированию (к сожалению, нам не известны подобные методические исследования).

Покажем, что языки из некоторого подмножества множества эзотерических языков могут быть органично вписаны в содержание обучения информатике и программированию в школе и вузе.

Для достижения поставленной цели сначала остановимся на построенной нами классификации эзотерических языков программирования (см. также [6, 7]), для описания которой приведем основания классификации и выполним классифицирование известных эзотерических языков:

- 1) *INTERCAL-подобные языки*, основной идеей которых является максимальное отличие от уже существующих языков программирования: INTERCAL, FALSE;
- 2) *Brainfuck-подобные языки*, которые при сохранении полноты по Тьюрингу ориентированы на минимальный синтаксис и минимальный размер компилятора: Brainfuck, Brainfork (многозадачный BF), f*ckf*ck, DoubleFuck, Whitespace, Spoon, CaneCode, LOLCODE, Petooh;
- 3) *языки — «черные ящики»*, созданные с целью максимального затруднения процесса создания программного кода: Malbolge;
- 4) *языки, основанные на копировании собственного кода*: Smetana, SMITH, Muriel;
- 5) *языки для недетерминированного* («случайного», *англ.* incidental) *программирования*: Entropy, Whenever, Java2k, FiPl;
- 6) *Befunge-подобные* (или *фунгеоидные*) *языки*, использующие многомерные представления компьютерных программ и технологию программирования «с использованием стрелок»:
 - Unefunge — одномерный;

- Befunge-93 — двухмерный, не являющийся полным по Тьюрингу;
- Befunge — двухмерный;
- Byter — двухмерный;
- Piet — двухмерный, цветовое кодирование;
- Trefunge — трехмерный;
- 4DL — четырехмерный;

7) *языки, пародирующие парадигмы и технологии программирования:*

- Spaghetti, GOTO++ — пародия структурного программирования;
- HQ9+, HQ9++ — пародия процесса тестирования;
- Lithp (J. Zussman, 1982) — пародия на язык Lisp;
- paranoid;

8) *языки, созданные для проверки математических концепций вычислимости:*

- ALPACA — язык, основанный на клеточных автоматах;
- Thue — язык, основанный на полусистемах Туэ;
- Unlambda, Iota, Jot — языки, основанные на λ -исчислении;
- Lazy K;

9) *языки с синтаксисом, имеющим литературную интерпретацию:*

- Chef — на базе кулинарных книг;
- Shakespeare — пьесы Шекспира;
- Haifu — стихотворения в стиле хокку;
- ArnoldC — на базе фраз А. Шварценеггера;
- FiM++ — письма Принцессе Селестии;
- smilesript — использует смайлики;
- emoticon — использует смайлики;
- reMorse — использует азбуку Морзе;
- Beatnik — использует язык хиппи;

10) *языки с измененной (нечеловеческой) формальной логикой:*

- Var'aq — логика расы клингонов из сериала «Звездный путь»;
- Ook! — на базе языка орангутанов;
- COW — на базе языка коров;

11) *языки, не являющиеся эзотерическими*, но часто включаемые в списки эзотерических языков программирования: APL, SNOBOL, Quake C, MC++;

12) *эзотерические языки с неизвестной нам семантикой*: SIMPLE, SARTRE, C—.

Из перечня оснований предложенной классификации следует, что эзотерические языки программирования — это вид языков программирования, не предназначенных для практического применения. В то же время есть все основания считать, что подобные языки можно использовать в качестве средства обучения программированию в школе (см., например, п. 5, 6, 7, 9). Более того, программирование на некоторых из них является неплохой тренировкой, а эзотерические языки нередко включают в список разрешенных языков на конкурсах (соревнованиях, олимпиадах) по программированию.

Покажем, что в вузе эзотерические языки можно использовать не только в качестве *средства*, но и в качестве *предмета* обучения программированию.

В самом деле, из п. 8 приведенной классификации следует, что **существуют эзотерические языки программирования, которые разработаны в качестве:**

- а) эксперимента по осуществлению попытки довести некоторую математическую концепцию понятия «вычислимость» до абсолюта (другими словами, являются реализацией некоторой математической концепции);
- б) интеллектуального вызова для разработчика этого языка;
- в) средства обучения экзотерическому (!) программированию.

В дальнейшем рассмотрим языки, относящиеся к пункту а), которые назовем **эзотерическими языками теоретического программирования**.

Наличие здесь слов «теоретическое программирование» связано с тем, что в теоретических исследованиях в области теоретического программирования сложился *математический аппарат*, содержащий несколько взаимосвязанных разделов: λ -исчисление, комбинаторная логика, теория типов языков программирования, теория категорий.

Рассмотрим некоторые эзотерические языки теоретического программирования [4].

Язык Unlambda — это программная модель бестипового исчисления комбинаторов в базе $\{S, K\}$ без использования модели операции абстракции; более того, он является своеобразным пересечением двух семейств языков программирования: эзотерических и функциональных языков. В языке существуют встроенные функции k и s , которые соответствуют двум базовым комбинаторам K и S соответственно; кроме того, в этом языке имеется оператор аппликации. Все это делает язык полным по А. Тьюрингу.

Приведем пример программы, которая выполняет процесс применения оператора аппликации (‘) к заданным комбинаторам S и K (пояснения представлены однострочными комментариями, начинающимися с ‘#’):

```
# SF G M FM (GM) KM H M
# ‘ ‘sk s s = ‘ ‘ks ‘ss => ‘ ‘ks ‘ss = s
# -----
# ‘ ‘skss
# Результат
# s
```

В дополнение к базовым функциям и оператору применения добавлены методы для осуществления ввода-вывода и организации «ленивых» вычислений.

Главное назначение языка заключается в демонстрации минимального чистого функционального языка программирования, а не в разработке программного обеспечения.

В языке программирования Iota (Ch. Barker) используется только *1-комбинатор* (*йота-комбинатор*) вида $i = \lambda f.fSK$.

Несмотря на это, язык Iota является *универсальным языком программирования*, так как i позволяет представить комбинаторы I , S и K :

$$I = i i, K = i (i (i i)), S = i (i (i i)).$$

Язык программирования Charity (Cockett & Fukushima, 1992) — экспериментальный язык, показывающий, что использование паттернов рекурсии является более рациональным способом задания рекурсивных функций.

Паттерном рекурсии называется описание взаимодействия функций и функционалов функционального языка программирования, приспособленных для рекурсивного решения задачи в конкретном контексте.

Язык Joy является чистым функциональным языком, основанным на комбинаторной логике. Синтаксис языка похож на синтаксис языка Форт: используется обратная польская запись и стек. Имеет теоретическую основу, из которой вытекает преимущество перед другими языками (линейная запись без использования переменных, что должно облегчить автоматизацию написания, проверки и оптимизации программ).

Упомянем также **язык FP** (Дж. Бэкус), который включает в себя как свойства эзотерических языков, так и свойства языков функционального программирования.

Теперь становится очевидным место **указанных выше эзотерических языков (Unlambda, Iota, Charity, Joy) в содержании обучения теоретическому программированию:**

- языковая иллюстрация исчисления комбинаторов (как типового, так и бестипового) — с помощью языка Unlambda;
- языковая иллюстрация базиса комбинаторов, состоящего из одного комбинатора, — с помощью языка Iota;
- языковая иллюстрация комбинаторной логики — с помощью языка Joy;
- языковая иллюстрация паттернов рекурсии (использование функционала «свертка») — с помощью языка Charity.

Для обоснования того, что выделенные языки образуют «маленькую» парадигму программирования, воспользуемся определением понятия *парадигма программирования* [3], под которой понимается образец мышления, предназначенный для отображения класса задач в алгоритмы их решения средствами языков программирования и представляющий собой систему со следующей структурой:

- 1) математические основания (формальные языки), определяющие особенности синтаксиса и семантики языков данной парадигмы;
- 2) конкретные языки программирования, позволяющие выразить алгоритм решения задачи;
- 3) технологии программирования, упрощающие реализацию алгоритмов с учетом архитектурной платформы, операционной системы и системы программирования.

Выполнение п. 1 и 2 для эзотерических языков теоретического программирования очевидно; выполнимость п. 3 не очевидна.

Более того, эзотерические языки теоретического программирования можно отнести к «математически обоснованным» языкам программирования, которые некоторые авторы (Nelson, 1991; Appel, 1992) выделяют в особую категорию «языки про-

граммирования, наследованные от математики» (*англ.* mathematically-derived languages), а А. Кэй отделяет языки, являющиеся «стилем во плоти» (*англ.* crystalization of style), от прочих языков, являющихся «склеиванием возможностей» (*англ.* agglutination of features).

Математически обоснованные языки программирования — это языки, семантика которых является непосредственным воплощением некоторой математической модели, незначительно адаптированной (без нарушения целостности) для превращения в язык для разработки реальных программ. Лишь некоторые языки попадают в эту категорию, большинство языков проектируются исходя из возможности эффективной трансляции в машину Тьюринга и имеют лишь некое подмножество в своем составе, воплощающее ту или иную математическую модель — от арифметики до средств параллелизма (например, Ocaml-л — это Ocaml, дополненный набором конструкций, реализующих π -исчисление).

Приведем **примеры математически обоснованных языков и воплощаемых ими математических моделей:**

- Agda, Epigram, Idris — интуиционистская теория типов Мартин-Лефа;
- APL и его потомки (J, K) — оригинальная семантика, не имеющая названия, воплощающая нотацию Айверсона для работы с массивами (термин *array languages*);
- Coq — исчисление индуктивных конструкций;
- Erlang — исчисление процессов (акторы, π -исчисление);
- Forth — стековая машина и конкатенативный язык программирования;
- Haskell — теория категорий (включая декартово замкнутую категорию, воплощающую λ -исчисление; категорию монад для моделирования побочных эффектов; расширение системы типов Хиндли-Милнера; систему кайндов);
- Joy — композиция функций и гомоморфизм (другими словами, чистый конкатенативный язык и как следствие чистый функциональный язык);
- Lisp — лямбда-исчисление Черча;
- Scheme — «облагороженный» диалект языка Lisp (сильнее типизированный);
- ML — типизированное λ -исчисление, т. е. λ -исчисление, дополненное системой типов Хиндли-Милнера;
- Prolog — исчисление предикатов;
- Mercury — исчисление предикатов, дополненное системой типов Хиндли-Милнера;
- Smalltalk — теория множеств;
- Unlambda — комбинаторная логика;
- Рефал — оригинальная семантика Турчина, носящая название «Рефал-машины» или «Рефал-автомата», создана на основе нормальных алгорифмов Маркова и воплощает композицию теории автоматов, сопоставления с образцом и переписывания термов;
- SQL — исчисление кортежей (вариант реляционного исчисления, основанного на исчислении предикатов первого порядка).

Наличие математического обоснования для языка программирования может гарантировать (или как минимум обещать с очень высокой вероятностью) некоторые или все из следующих положительных свойств:

1. Существенное повышение стабильности программ; в одних случаях — за счет построения доказательства надежности для самого языка (безопасность типов), существенного упрощения формальной верификации программ и даже получения языка, который сам является системой автоматического доказательства (Coq, Agda), в других случаях — за счет раннего обнаружения ошибок (Forth и регулярные выражения).

2. Обеспечение потенциально более высокой эффективности программ. Даже если семантика языка далека от архитектуры целевой платформы компиляции, к нему могут быть применимы формальные методики глобального анализа программ (хотя трудоемкость написания даже тривиального транслятора может оказаться выше).

Например, для языков Scheme и Standard ML существуют развитые оптимизирующие компиляторы и суперкомпиляторы, результат работы которых может уверенно конкурировать по скорости с языком низкого уровня C и даже опережать последний (хотя ресурсоемкость работы самих компиляторов оказывается значительно выше).

Язык Scala (унаследовавший математику от ML) обеспечивает на платформе JVM более высокую скорость, чем «родной» для нее язык Java. С другой стороны, Forth имеет репутацию одного из самых нетребовательных к ресурсам языков (менее требователен, чем C) и используется для разработки приложений реального времени для самых маломощных компьютеров; кроме того, транслятор языка Forth является одним из наименее трудоемких в реализации на ассемблере.

3. Заранее известный (неограниченный или, наоборот, четко очерченный) предел роста сложности программных компонентов, систем и комплексов, которые можно выразить средствами языка с сохранением показателей качества. Языки, не имеющие математического обоснования (а именно такие наиболее часто применяются в мейнстриме: C++, Java, C#, Delphi и др.), на практике ограничивают реализуемую функциональность и/или снижают качество по мере усложнения системы, так как им присущи экспоненциальные кривые роста сложности как касательно работы одного отдельно взятого человека, так и касательно сложности управления проектом в целом.

Прогнозируемая сложность системы приводит либо к поэтапной декомпозиции проекта на множество более мелких задач, каждая из которых решается соответствующим языком, либо к языково-ориентированному программированию для случая, когда адресуемой языком задачей является как раз описание семантик и/или символьные вычисления (Lisp, ML, Haskell, Рефал).

4. Удобство для человека при решении задач, на которые язык ориентирован по своей природе, что в некоторой степени также способно (косвенно) повлиять на повышение стабильности результирующих программ за счет повышения вероятности

обнаружения ошибок в исходном коде и снижения дублирования кода.

Отметим, что языки, наследованные от «наследованных от математики», уже не обязательно будут обладать этими свойствами. Так, язык Python соединяет несколько упомянутых моделей, но для их совмещения не существует математического обоснования, поэтому он не может считаться «наследованным от математики», и как следствие ему присущи лишь последнее из указанных свойств.

Выводы.

Построенная нами классификация эзотерических языков программирования позволила описать границы парадигмы программирования, которая названа эзотерической парадигмой теоретического программирования. Установлено, что некоторые эзотерические языки программирования (Unlambda, Lazy K, Iota, Jot) фактически являются экспериментальными функциональными языками для исследования способов описания *вычислимости* и могут быть использованы в процессе обучения

теоретическому программированию в вузе в качестве предмета обучения.

Литература

1. Коменский Я. А. Великая дидактика. М.: Изд. кн. магазина К. И. Тихомирова, 1896.
2. Коллеж Н. Г. Словарь иностранных слов. М.: Эксмо, 2006.
3. Кудрявцева И. А. Методика обучения бакалавров физико-математического образования математическим основам парадигм программирования: автореф. дис. ... канд. пед. наук. СПб.: РГПУ им. А. И. Герцена, 2005.
4. Кудрявцева И. А. Эзотерические языки в обучении теоретическому программированию // Научное мнение. Психолого-педагогические и юридические науки: научный журнал / Санкт-Петербургский университетский консорциум. 2014. № 11.
5. Сондерс М. Лингва эзотерика // LINUX Format. 2007. № 3 (90).
6. Тюгашев А. А. Языки программирования. СПб.: Питер, 2014.
7. Харьковский З. Путеводитель автостопщика по таинственным знаниям // Компьютерра. 2005. № 12 (584).

НОВОСТИ

Российские ученые предлагают «разогнать» память компьютеров в тысячу раз

Ученые из МФТИ и их коллеги из Германии и Нидерландов нашли способ значительно увеличить скорость работы компьютеров: они предложили использовать так называемые Т-лучи, терагерцовое излучение, которое может переключать ячейки компьютерной памяти в тысячи раз быстрее, чем воздействие магнитного поля, говорится в статье, опубликованной в журнале Nature Photonics.

«Мы сделали важный шаг на пути к терагерцовой электронике: показали качественно новый подход к контролю намагниченности с помощью коротких импульсов терагерцового излучения. Насколько нам известно, наша работа — первое применение подобного возбуждения колебаний магнитных подсистем», — говорит соавтор исследования Анатолий Звездин, руководитель лаборатории физики магнитных гетероструктур и спинтроники для энергосберегающих информационных технологий МФТИ и сотрудник Института общей физики имени А. М. Прохорова.

Быстрый рост объема цифровых данных и сложности вычислительных задач вынуждает создателей компьютеров непрерывно увеличивать скорость вычислений. Многие эксперты считают, что классические компьютеры уже подходят к пределам роста быстродействия, поэтому ученые по всему миру ищут принципиально новые вычислительные технологии. Одно из «узких мест» современных вычислительных машин — память — на каждую операцию записи и перезаписи одной магнитной ячейки нужно время. Сократить время этого цикла очень сложно.

Себастиан Байерл (Sebastian Baierl) из Университета Регенсбурга, Анатолий Звездин, Алексей Кимель из Университета Неймегена (Нидерланды) и МИРЭА, а также их коллеги предложили использовать для переключения

памяти не магнитные поля, а импульсы терагерцового излучения — так называют электромагнитное излучение в диапазоне между инфракрасным и СВЧ-диапазоном с длиной волны около 0,1 миллиметра. Его используют, например, в установках для сканирования в аэропортах: Т-лучи не вредны для организма, но позволяют легко «увидеть» под одеждой оружие и взрывчатку.

Ученые решили провести эксперимент со слабым ферромагнетиком — ортоферритом тулия ($TmFeO_3$), чтобы проверить насколько подходят Т-лучи на роль переключателя состояний «магнитных битов» в этом материале. Магнитное поле в ферромагнетиках создается за счет определенной ориентации спинов — магнитных моментов атомов в микрокристаллах (доменах). Чтобы поменять эту ориентацию, нужно внешнее магнитное поле.

Однако эксперимент показал, что терагерцовое излучение переводит ионы тулия в возбужденное состояние и меняет магнитные свойства и ионов железа, и ионов тулия, причем его воздействие почти в 10 раз сильнее внешнего магнитного поля. Таким образом, исследователи получили очень быстрый и действенный метод «перемагничивания» — хорошую «базу» для создания сверхбыстрой памяти.

По словам ученых, ортоферрит тулия — не единственный материал, который можно «переключать» Т-лучами. Он достаточно удобен для демонстрации, но сам предложенный способ контроля намагниченности можно использовать во многих других магнитных материалах.

«В СССР ортоферриты исследовала группа в МГУ, у нас был приоритет в этой области, в каком-то смысле эта работа — продолжение тех исследований», — отметил Анатолий Звездин.

(По материалам сайта S&T RF «Наука и технологии России»)