ИПМ им.М.В.Келдыша РАН

Абрау-2023 • Труды конференции

Л.В. Городняя

Относительное определение языков программирования

Рекомендуемая форма библиографической ссылки

Городняя Л.В. Относительное определение языков программирования // Научный сервис в сети Интернет: труды XXV Всероссийской научной конференции (18-21 сентября 2023 г., онлайн). — М.: ИПМ им. М.В.Келдыша, 2023. — С. 117-129.
https://doi.org/10.20948/abrau-2023-7
https://keldysh.ru/abrau/2023/theses/7.pdf

Видеозапись выступления

Относительное определение языков программирования

Л.В. Городняя[1, 2]

[1] *Институт систем информатики им. А.П. Ершова СО РАН*
[2] *Новосибирский государственный университет*

Аннотация. В тезисах предложен формализм, поддерживающий относительное определение языков программирования, позволяющий чётко выражать разницу между языками при их сравнительном анализе, а также, при определении новых языков на основе уже известных. Формализм представляет собой расширение БНФ, в котором имена используемых понятий можно сопровождать указанием на близкие понятия из других языков. Поддержана возможность указывать части используемого понятия, такие как варианты определения или неявные понятия, с точностью до синтаксической позиции

Ключевые слова: определение языка программирования, сравнение языков программирования, расширение БНФ, сквозные понятия, инородные понятия, неявные понятия, синтаксическая позиция, функциональное программирование

Relative programming language definition

L.V.Gorodnyaya[1,2]

[1] *A.P. Ershov Institute of Informatics Systems*
[2] *Novosibirsk State University*

Abstract. The article proposes a formalism that supports the relative definition of programming languages, allowing you to clearly express the difference between languages when analyzing and comparing them, as well as when defining new languages based on already known ones. Formalism is an extension of the BNF, in which the names of the concepts used can be accompanied by an indication of foreign concepts from other languages. As a result, it becomes possible to specify parts of the used concept, such as variants of the definition or implicit concepts, up to a syntactic position. The syntactic position can be specified using a terminal string, symbolizing the choice of a variant of the definition of a concept or indicating an implicit concept, the unnamed definition of which is located immediately after the string.

Keywords: programming language definition, comparison of programming languages, BNF extension, general concepts, foreign concepts, implicit concepts, syntactic position, functional programming

В лаборатории информационных систем ИСИ СО РАН традиционно исследуются средства и методы создания и применения языков программирования в процессе обучения программированию [5-10]. При выполнении работ по сравнительному анализу языков программирования (ЯП) нужны средства для точного представления выявленного сходства и различия исследуемых ЯП [2, 4, 15-17]. Исследование и создание новых языков может использовать близкие понятия ряда предшественников. И в том, и в другом случае выделяют конструкции, соответствующие общим понятиям, вариантам синтаксиса и семантики их представления, а также версиям их прагматики, возможно развивающимся в таких процессах [1, 11, 18, 20]. Общими могут быть как полные определения исходных понятий ЯП, так и их части, включая неявные понятия, не имеющие имён в определении, но допускающие указание синтаксической позиции. Синтаксическая позиция может быть задана с помощью терминальной цепочки, символизирующей выбор варианта определения или указывающей на неявное понятие, безымянное определение которого расположено непосредственно вслед за цепочкой. Такие понятия можно выделять в отдельные вспомогательные понятия. При переходе в другой язык определение может расширяться. Известны прецеденты разработки новых ЯП на основе общепризнанных, доказавших свою жизнеспособность. Так, например, языки Val и Sisal созданы на основе языка Pascal, а mPC на основе C. Имеются ряды родственных ЯП, такие как Lisp 1, Lisp 1.5, muLisp, Clisp, Clojure или C, C++, C#, для которых можно точно представлять границы понятий, изменяющихся по мере продвижения по этому ряду. Известны линейки языков или подъязыков, связанных отношением наследования или влияния, допускающие чёткое представления фактических заимствований и уровня новизны или своих решений [13, 25, 27, 28].

Для показа методики относительных определений выполнено самоописание предлагаемого формализма, опирающееся на классические БНФ. Цель методики — создание поддержки для сравнительного исследования имеющихся и вновь создаваемых ЯП. Тезисы содержат пояснения по такой методике применения формализма отдельных определений при сравнении языков программирования, формальные определения которых доступны.

1. Сравнительный анализ языков программирования

Уже в 1960-ых годах, когда число ЯП исчислялось десятками, Б.Хигман обратил внимание на проблему сравнительного изучения ЯП [20]. На результатах анализа языков от ассемблера до CPL, включая Fortran, Cobol, Lisp, Algol, IPL-V и PL/1, Б.Хигман показал историю развития программистских понятий, связывая её с расширением сферы компьютерных приложений и отмечая отличия от лингвистической и математической терминологии. Он счёл нужным особо рассмотреть средства ввода-вывода данных, необходимые для подготовки и отладки программ. Б.Хигман особо подчёркнул, что целью создания ЯП является

будущее. В 1970-е число ЯП исчисляется сотнями, а в 1980-е тысячами. В эти годы появляются книги Т.Пратта и Т.Пратта с М.Зелковиц, в которых рассматривается и упоминается более 30-ти ЯП (C, Java, Perl, Pascal, ML, Fortran, C++, Smalltalk, Prolog, Lisp, ada, Postscript, XML, CGI, LaTeX, html, PL/1, Snobol, APL, Basic, Cobol, CPL, DCPL, assembler, CORC, CUPL, PL/C, Algol-60, Algol W и др.), причём их анализу предпослан обширный обзор эволюции архитектурных возможностей и методов разработки систем программирования [15, 16], включая средства отладки и верификации программ. Материал каждого ЯП привлекается по мере описания прогресса электроники и новых решений при подготовке программ. Получается ряд ЯП, ответивших на такие технологические вызовы — Fortran, Lisp, C, Pascal, Ada, Perl, ML, C++, Prolog, Smalltalk, Postscript, Java. Сравнение учебных ЯП можно рассматривать как упорядочивание по продуктивности программирования [8, 9].

Теперь число ЯП исчисляется десятками тысяч. Появляются новые попытки сравнивать ЯП. Например, Л.Прехельт представил результаты сравнения семи ЯП (C, C++, Java, Perl, Python, Rexx и Tcl) на основе измерения отдельных характеристик производительности программ [17]. В этом плане показательны результаты измерения качество компиляторов, учитывающее около 200 весьма различных параметров [3]. Интересны, а во многом и неожиданны, результаты сравнения языка Lisp с языком C++ и другими ЯП по продуктивности программирования [29-35]. Иногда производительность компиляторов повышают погружением их реализации на аппаратный уровень. [14]. Сравнительный анализ языков Algol и Algol 68 лёг в основу эксперимента по конвертации языка высокого уровня без привлечения промежуточного языка низкого уровня [19].

Таким образом, кроме собственно наследования существуют разные параметры отношений порядка в пространстве ЯП — расширение сферы приложений, эволюция архитектуры, производительность программ, продуктивность программирования, а также различные направления использования результатов сравнительного анализа ЯП. Существуют и другие критерии упорядочения ЯП по востребованности и результативности [23]. Доступны ресурсы с достаточно обширными данными о современном пространстве ЯП с частичной характеристикой их «анкетных данных», включая отношение наследования без указания параметров наследования и его границ [24-26]. Для навигации в таком пространстве просто сообщается о взаимовлиянии отдельных ЯП и поддержанных в них парадигмах. Например, язык Haskell создан под влияние языков Lisp, Common Lisp, Scheme, ISWIM (Landin), APL, FP (Backus), ML, Standard ML, Hope, Hope+, Clean, Id, Gofer, Sisal, Miranda (Turner), язык Kotlin — под влиянием Groovy, C#, Gosu, Java, Ruby, JavaScript, Scala, Python и ML. Язык Java повлиял на Ada 2005, BeanShell, C#, Chapel, Clojure, ECMAScript, Fantom, Gambas, Groovy, Hack, Haxe, J#, Kotlin, PHP, Python, Scala, Seed7, Vala, JavaScript, JS++, ML на Miranda, Haskell, Cyclone, Nemerle, C++. Уточнение таких сведений более детальными данными о результатах сравнительного анализа по производственно значимым параметрам может

служить обоснованием выбора ЯП для программистских проектов, а также для развёртывания работ, использующих разные ЯП.

Для более точной ориентировки в пространстве ЯП нужны средства и методы детального представления сходства и различия конкретных языков, позволяющие понимать их достоинства и недостатки, причины наследования одних механизмов, отказа от других и привлечения новых реализационных решений, что входит в задачу сравнительного исследования ЯП.

2. Требования к представлению результатов анализа

Языки программирования обычно допускают представление их определений в виде форм Бэкуса-Наура (БНФ). Рассмотрим задачу представления результатов сравнительного анализа ЯП, обладающих таким формальным определением или достаточно ясным описанием с примерами программ, позволяющими такое определение восстановить. Часть понятий в ряде языков могут обладать сходством, учёт которого полезен при изучении ЯП и выборе решений при разработке для него системы программирования (СП). Задача сравнительного анализа ЯП заключается в приведении их определений к форме, по которой можно точно представлять наследование понятий при переходе от одного ЯП к другому. Для этого недостаточно одинаковым понятиям дать одинаковые имена. Сходство понятий может быть частичным, на уровне вариантов определения или не выделенных в определении неявных понятий. Для двух языков, один из которых рассматривается как наследник другого, необходимо явное представление наследуемых понятий с указанием предшественника. Для конкретного языка можно задать список других ЯП, с которыми следует выразить его сходство и различие. Таким образом, для решения задач сравнительного исследования ЯП необходимо следующие средства и методы:

1. объявление группы близких языков, связанных отношение наследования;
2. определение формы для представления наследуемых вариантов общих понятий;
3. определение границ частично наследуемых неявных понятий;
4. унификация имён общих понятий;
5. приведение определений ЯП к форме, удобной для представления результатов анализа;
6. инструментарий для проверки корректности представления результатов сравнительного анализа.

3. Расширения БНФ

Классическое определение БНФ обычно использует ряд обозначений, слегка варьирующихся в разных источниках. Такие формулы достаточны для

автоматизации анализа и генерации контекстно-свободных языков на основе нормализованного набора правил, связывающих имена понятий языка с их определениями, обладающими вариантами из цепочек элементов. Элементами могут быть символы, лексемы, ключевые слова или имена понятий. При самоописании БНФ часто выделяют следующие понятия: Правило, Понятие, Определение, Имя, Цепочка, Элемент, Терминал, Ключевое_слово, Знак, Лексема. Вводим обозначения для определения синтаксиса ЯП, использующего понятия или их части их определений других ЯП, начиная с обычных БНФ[1] :

::= — Разделитель левой и правой частей правила.

| — Разделитель вариантов правила.

" — ограничитель терминального символа.

<Пусто> — Пустая строка.

Этих обозначений достаточно для классических БНФ. Расширенные БНФ (РБНФ) для удобства автоматизации конструирования компиляторов структурируют определение языка на участки, обладающие разными методами обработки при создании анализатора текстов программ. Самоопределение РБНФ наследует ряд понятий БНФ (Правило, Понятие, Определение, Цепочка, Терминал, Знак, Лексема, Ключ) и дополняет понятие «Элемент» структурирующими скобками и соответствующими правилами (Группа, Факультатив, Многократно, Перечень, Один_из). Для самоопределения расширенных БНФ часто вводят дополнительные обозначения:

**** — разделитель элемента и знака в перечне, содержащем хотя бы один элемент;

( ) — парные круглые скобки для выделения группы элементов;

[ ] — парные квадратные скобки для выделения факультативной группы элементов;

{ } — парные фигурные скобки для выделения перечня из независимых элементов;

... — повторение произвольного числа предшествующего элемента.

Относительные РБНФ (ОРБНФ) для представления результатов сравнительного исследования ЯП обеспечивают разметку определения языка наименованиями предшественников или близких ЯП. Возможно наследование полного гнезда определения,\. В таком случае все используемые понятия. ОРБНФ наследует от РБНФ понятия Цепочка, Элемент, Терминал, Знак, Лексема, Ключ, Группа, Факультатив, Многократно, Перечень, Один. Возможно уточнение прежнего правила из РБНФ. Такое уточнение в определении ОРБНФ выполнено

---

[1] Для удобочитаемости мета-обозначения выделены шрифтом **Arial Black**

для понятий Правило и Понятие. Кроме того, введены свои, новые правила «ПонятиеЯП», «ИндексПеременная», «Номер». ПонятиеЯП нужно для обозначения общего понятия, имеющего определения в разных ЯП. Варианты определения выбираются по заданным терминальным символом — префиксам для выделения нужного варианта. Возможно использование локального неявного понятия, определение которого начинается с синтаксической позиции вслед за терминалом. ИндексПеременная предназначена для обозначения многократно входящего понятия при необходимости хранить разные значения, полученные при разборе текста.

Для языков, использующих инородные понятия из других ЯП, вводим обозначения, выражающие связь между языками:

: — // принадлежность ЯП

_ — // хоть что,

. — // префикс или постфикс неявного понятия

! — // префиксы Вариантов

КЯП **: (** ЯП **[.**Имя**] ...)** — конкретный ЯП, наследующий понятия или части определений из перечисленных ЯП.

Имя **::= |** Цепочка — дополнительный вариант к более раннему определению.

Имя **:** ЯП **[.**Имя**]** — имя понятия, наследуемого из данного ЯП.

{ Имя **... } :** ЯП **[.**Имя**]** — множество понятий, наследуемых из данного ЯП.

Имя **: ((** ЯП **[.**Имя**]) ...)** — имя понятия, наследуемого из списка ЯП.

Имя **:** ЯП **[.**Имя**] ! (...** "Терминал**" ...)** — вырезка Вариантов из определения данного ЯП. Варианты определения имеют префиксы, заданные терминальным символом, или постфиксы.

Имя **:** ЯП **[.**Имя**] .** "Терминал**"... [**"Терминал**"]** — Выделебие неявного понятия, определение которого начинается с синтаксической позиции вслед за терминальным символом до конца варианта или до следующего заданного терминального символа.

Такие обозначения при сравнительном анализе двух ЯП позволяют выражать определения понятий заданного языка по отношению к близким понятиям другого ЯП:

1. понятие языка можно определить как близкое понятие другого ЯП;
2. понятие языка можно определить как ряд вариантов близкого понятия другого ЯП;
3. понятие языка можно определить как неявное понятие, выделяемое с помощью терминальных символов из одного варианта близкого понятия другого ЯП;
4. понятие языка можно определить, присоединив к определению

понятия другого ЯП дополнительные варианты;

5. понятие языка можно определить как объединение определений близких понятий из ряда других ЯП.

4. Опыт применения ОРБНФ при определении языка СИНХРО

При создании новых ЯП нередко используют термин «диалекты» для описания частей языка [21, 22]. Учебный язык СИНХРО содержит пять диалектов, связанных с этапами обучения [10]. Исходный объём определения составлял примерно пять страниц. Диалект Мульти нацелен на ознакомление с феноменами параллелизма с помощью игр на клетчатой доске. Далее работает диалект Асинхр для представления асинхронных процессов в виде многопоточных программ из независимвых потоков, затем привлекается диалект Синпар, позволяющий приобретать навыки синхронизации многопоточных программ, и диалект АМК (абстрактный многопроцессорный комплекс) для представления многопроцессорных программ с использованием общей памяти. Есть ещё диалект Трансформ для представления, отладки, оптимизации, декомпозиции и преобразования программ на всех этапах обучения.

Диалект АМК вводит понятия Программа, Контекст, Процессор, Регистр, Система_команд, Набор, Поток, Действие, Команда, Константа, Данное, Перечень, Имя. Объём определения меньше половины страницы без описания команд.

Диалект Мульти, нацеленный на ознакомление с параллелизмом, наследует из определения АМК понятия Набор, Поток, Имя, Перечень, Система_команд, Константа и дополняет определения понятий Программа, Контекст, Данное, Действие, Команда. Кроме того, он вводит свои определения понятий Условие, Клетка, Изображение, Робот, Исполнитель, Процессор, Сценарий, используемых при оперировании играми и их конструировании на начальном этапе ознакомления с параллелизмом. Другими диалектам эти понятия не наследуются. Объём определения две страницы.

Диалект Асинхр ориентирован на приобретение навыков многопоточного программирования с наследованием понятий из диалектов АМК и Мульти. Его определение наследует понятия Программа, Имя, Перечень из Мульти и понятие Команда из АМК. Кроме того, уточняются понятия Контекст из АМК и Действие, Условие, Набор, Поток из Мульти. Вводятся и свои понятия — Структура, Определение, Адрес, Фрагмент, Выр, Арг, Формула, Фильтр, Индекс, Свёртка, Функция, Мемофункция, Схема, МакроПараметр, возникающие при решении ряда учебных задач по организации асинхронных процессов. Объём определения одна страница.

Определение диалекта Синпар наиболее ярко показывает выигрыш от ОРБНФ. Этот диалект наследует понятия из диалекта Асинхр, унаследовавшего ряд понятий из Мульти и АМК. Из диалекта Асинхр наследуются понятия Контекст, Определение, Поток, Условие, Адрес, Фрагмент, Выр, Формула,

Структура, Фильтр, Индекс, Свёртка, Арг, МакроПараметр. Кроме того, уточняются определения понятий Набор. Данное, Действие связанные с появлением нового понятия Барьер. В результате собственно определение Синпар становится предельно лаконичным на фоне знакомства с предшественниками. Объём определения меньше трети страницы. Общий объём определения языка СИНХРО сократился примерно вдвое, что особенно заметно на определении диалекта Синпар, в котором после наследования понятий диалекта Асинхр достаточно ввести определения, связанные с барьерами в сети процессов.

Заключение

В данных тезисах описан формализм относительного определения языка программирования — ОРБНФ, удобного для представления результатов анализа и сравнения ЯП, а также конструирования новых ЯП с использованием ранее созданных определений. Опыт студенческих семестровых работ показал, что с заданием на анализ ЯП справляются почти все, но задания на сравнение ЯП оказались неподъёмными. Не исключено, что причина связана с необходимостью изучать обширный новый материал. Описания ЯП нередко занимают более тысячи страниц. Это можно смягчить приведением описаний к понятийным таблицам. Другая причина может быть связана с отсутствием чётких форм представления результатов сравнения, что можно преодолеть с помощью относительных БНФ, особенно для ЯП, описания которых можно сопровождать готовой понятийной таблицей. Именно такой механизм рассмотрен в данных тезисах. Для ЯП, связанных маршрутом наследования, доступным по данным из Википедии или специальным сайтам [25-28], такой механизм позволяет представлять шаги изменения используемых средств и решений. Для независимых ЯП так можно представлять их сходство и различие. Пока приведена лишь небольшая иллюстрация предложенного формализма на материале диалектов учебного языка СИНХРО [10]. Планируется эксперимент по использованию ОРБНФ при сравнении родственных и независимых ЯП.

Литература

1. Андреева Т.А. и др. Компьютерные языки как форма и средство представления, порождения и анализа научных и профессиональных знаний. // Тр. XV Всерос. научно-методической конф. «Телематика-2008». — СПб., 2008. С. 77-78.
2. Баррон Д. Введение в языки программирования / 1980. 192 с.
3. Васючкова Т.С. Определение и построение метрики аттестации транслирующих систем. // Новосибирск. Трансляция и оптимизация программ. 1983. С. 5-14.
4. Вольфенгаген В. Э. Конструкции языков программирования. Приёмы описания. — М.: Центр ЮрИнфоР, 2001. — 276 с. — ISBN 5-89158-079-9

5. Городняя Л.В. Абстрактная машина языка программирования учебного назначения СИНХРО / Вестник НГУ. Серия: Информационные технологии. 2021;19(4):16-35. https://doi.org/10.25205/1818-7900-2021-19-4-16-35
6. Городняя Л.В. Модели работы с памятью в учебном языке программирования СИНХРО // Научный сервис в сети Интернет: труды XXIV Всероссийской научной конференции (19-22 сентября 2022 г., онлайн). — М.: ИПМ им. М.В.Келдыша, 2022. — С. 137-154. https://doi.org/10.20948/abrau-2022-1, https://keldysh.ru/abrau/2022/theses/1.pdf
7. Городняя Л.В. ПОДХОДЫ К ПРЕДСТАВЛЕНИЮ СИНТАКСИСА ЯЗЫКОВ ПРОГРАММИРОВАНИЯ / Новосибирск. Препринт 185 https://www.iis.nsk.su/files/preprints/Preprint_185.pdf
8. Городняя Л.В. Сравнение учебных языков программирования Бейсик, Паскаль и Рапира. // Информатика и компьютерная грамотность. - М.: Наука, 1988.
9. Городняя Л.В. Сравнение учебных языков программирования Бейсик, Паскаль, Рапира. — М.: "Информатика и образование", 2/1989
10. Городняя Л.В. Язык параллельного программирования Синхро, предназначенный для обучения. — Новосибирск, 2016. — 30 с. (Препринт/ИСИ СО РАН; N 180). URL: https://www.iis.nsk.su/files/preprints/gorodnyaya_180.pdf
11. Григорьев С.А., Ишанов С.А. Сравнительный курс языков программирования Fortran и C Учебное пособие. — Калининград: Изд-во КГУ, 1998. — 94 с.
12. Звенигородский Г.А. Первые уроки программирования. — Библиотечка «Квант». — М.: Наука, 1985. — Т. 41. http://cip.iis.nsk.su/files/course/zven-ves.pdf.
13. Лавров С.С. Расширяемость языков. Подходы и практика. В сб.: Прикладная информатика, вып. 2. М.: Финансы и статистика, 1984, с. 17-22.
14. Неменман М.Е. Специализированный процессор для системы программирования? / Языки и системы программирования, 1981. С. 17-20
15. Т.Пратт, "Языки программирования: разработка и реализация / М. Мир. 1979. 576 с.
16. Пратт Т., Зелковиц М. Языки программирования. Разработка и реализация Издательство Питер, 2002, 688 с. ISBN: 5-318-00189-0
17. Лутц Прехельт Эмпирическое сравнение семи языков программирования / Открытые системы. СУБД. 2002, № 12. 15.12.2000 https://www.osp.ru/os/2000/12/178361

18. Роберт У. Себеста. Основные концепции языков программирования / Пер. с англ. — 5-е изд. — М.: Вильямс, 2001. — 672 с. — 5000 экз. — ISBN 5-8459-0192-8 (рус.).
19. Рар А.Ф. Принципы построения конвертора с Алгола-60 на Алгол-68 // Новосибирск. Трансляция и преобразования программ. 1984
20. Хигман Б. Сравнительное изучение языков программирования Мир 1974. 204 с.
21. https://www.osnews.com/story/24798/new-red-programming-language-gets-syllable-backend/ New Red Programming Language Gets Syllable Backend
22. http://www.rebol.com/ REBOL Language. — https://web.archive.org/web/20090221215758/http://rebol.ru/ В ожидании Rebol 3T.
23. https://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%A0%D0%B5%D0%B9%D1%82%D0%B8%D0%BD%D0%B3_%D0%B2%D0%BE%D1%81%D1%82%D1%80%D0%B5%D0%B1%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D0%B8_%D1%8F%D0%B7%D1%8B%D0%BA%D0%BE%D0%B2_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F Рейтинг востребованности ЯП
24. https://skillbox.ru/media/code/kuda-propali-sovetskie-yazyki-programmirovaniya/ Рефал, Эль и Рапира. Куда пропали советские языки программирования?
25. https://hopl.info/findlanguages.prx?id=ru&which=ByCountry
26. http://compiler.su/entuziasty-razrabotchiki-kompilyatorov-i-ikh-proekty.php
27. Прогопедия http://ru.progopedia.com/
28. Левенец https://www.levenez.com/lang/
29. https://psachin.gitlab.io/lisp-java-notes.html — сравнение C++ vs Java vs Lisp
30. https://www.cs.unm.edu/~williams/cs257/lisp-study.html
31. *https://renato.athaydes.com/posts/revisiting-prechelt-paper-comparing-languages.html.*
32. Vivek Shangari http://rus-linux.net/MyLDP/algol/LISP/lisp01.html — Lisp: Слезы радости, ч.1 "Lisp: Tears of Joy, Part 1"
33. https://habr.com/ru/articles/542068/ Почему я остаюсь с Лиспом (и вам тоже стоит)
34. https://deep-econom.livejournal.com/237992.html Lisp
35. https://habr.com/ru/companies/sberbank/articles/655509/ Первый древнейший: в чём уникальность языка программирования LISP

References

1. Andreyeva T.A. i dr. Komp'yuternyye yazyki kak forma i sredstvo predstavleniya, porozhdeniya i analiza nauchnykh i professional'nykh znaniy. // Tr. XV Vseros. nauchno-metodicheskoy konf. «Telematika-2008». — SPb., 2008. S. 77-78.
2. Barron D. Vvedeniye v yazyki programmirovaniya / 1980. 192 s.
3. Vasyuchkova T.S. Opredeleniye i postroyeniye metriki attestatsii transliruyushchikh sistem. // Translyatsiya i optimizatsiya programm. 1983. S. 5-14
4. Vol'fengagen V. E. Konstruktsii yazykov programmirovaniya. Priyomy opisaniya. — M.: Tsentr YurInfoR, 2001. — 276 s. — ISBN 5-89158-079-9.
5. Gorodnyaya L.V. Abstraktnaya mashina yazyka programmirovaniya uchebnogo naznacheniya SINKHRO / Vestnik NGU. Seriya: Informatsionnyye tekhnologii. 2021;19(4):16-35. https://doi.org/10.25205/1818-7900-2021-19-4-16-35
6. Gorodnyaya L.V. Modeli raboty s pamyat'yu v uchebnom yazyke programmirovaniya SINKHRO //Nauchnyy servis v seti Internet: trudy XXIV Vserossiyskoy nauchnoy konferentsii (19-22 sentyabrya 2022 g., onlayn). — M.: IPM im. M.V.Keldysha, 2022. — S. 137-154. https://doi.org/10.20948/abrau-2022-1, https://keldysh.ru/abrau/2022/theses/1.pdf
7. Gorodnyaya L.V. PODKHODY K PREDSTAVLENIYU SINTAKSISA YAZYKOV PROGRAMMIROVANIYA Preprint 185. https://www.iis.nsk.su/files/preprints/Preprint_185.pdf
8. Gorodnyaya L.V. Sravneniye uchebnykh yazykov programmirovaniya Beysik, Paskal' i Rapira. // Informatika i komp'yuternaya gramotnost'. — M.: Nauka, 1988.
9. Gorodnyaya L.V. Sravneniye uchebnykh yazykov programmirovaniya Beysik, Paskal', Rapira. - M.: "Informatika i obrazovaniye", 2/1989
10. Gorodnyaya L.V. YAzyk parallel'nogo programmirovaniya Sinkhro, prednaznachennyy dlya obucheniya. — Novosibirsk, 2016. — 30 s. (Preprint/ISI SO RAN; N 180). URL: https://www.iis.nsk.su/files/preprints/gorodnyaya_180.pdf
11. Grigor'yev S.A., Ishanov S.A. Sravnitel'nyy kurs yazykov programmirovaniya Fortran i S Uchebnoye posobiye. — Kaliningrad: Izd-vo KGU, 1998. — 94 s.
12. Zvenigorodskiy G.A. Pervyye uroki programmirovaniya. — Bibliotechka «Kvanta». — M.: Nauka, 1985. — T. 41. http://cip.iis.nsk.su/files/course/zven-ves.pdf.
13. Lavrov S.S. Rasshiryayemost' yazykov. Podkhody i praktika. V sb.: Prikladnaya informatika, vyp. 2. M.: Finansy i statistika, 1984, s. 17 — 22.

14. Nemenman M.Ye. Spetsializirovannyy protsessor dlya sistemy programmirovaniya? / YAzyki i sistemy programmirovaniya, 1981. S. 17-20
15. T.Pratt, "YAzyki programmirovaniya: razrabotka i realizatsiya /M. Mir. 1979. 576 s.
16. Pratt T., Zelkovits M. YAzyki programmirovaniya. Razrabotka i realizatsiya Izdatel'stvo Piter, 2002, 688 s. ISBN: 5-318-00189-0
17. Lutts Prekhel't Empiricheskoye sravneniye semi yazykov programmirovaniya / Otkrytyye sistemy. SUBD. 2002, ? 12. 15.12.2000 https://www.osp.ru/os/2000/12/178361
18. Robert W. Sebesta. Osnovnyye kontseptsii yazykov programmirovaniya / Per s angl. — 5th ed. — M.: Williams, 2001. — 672 p. — ISBN 5-8459-0192-8 (Rus)
19. Rar A.F. Printsipy postroyeniya konvertora s Algola-60 na Algol-68 // Novosibirsk, Translyatsiya i preobrazovaniya programm. 1984
20. Khigman B. Sravnitel'noye izucheniye yazykov programmirovaniya Mir 1974. 204 s.
21. https://www.osnews.com/story/24798/new-red-programming-language-gets-syllable-backend/ New Red Programming Language Gets Syllable Backend
22. http://www.rebol.com/ REBOL Language. — https://web.archive.org/web/20090221215758/http://rebol.ru/ В ожидании Rebol 3T.
23. https://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:%D0%A0%D0%B5%D0%B9%D1%82%D0%B8%D0%BD%D0%B3_%D0%B2%D0%BE%D1%81%D1%82%D1%80%D0%B5%D0%B1%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D0%B8_%D1%8F%D0%B7%D1%8B%D0%BA%D0%BE%D0%B2_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F Reyting vostrebovannosti YAP
24. https://skillbox.ru/media/code/kuda-propali-sovetskie-yazyki-programmirovaniya/ Refal, El' i Rapira. Kuda propali sovetskiye yazyki programmirovaniya?
25. https://hopl.info/findlanguages.prx?id=ru&which=ByCountry
26. http://compiler.su/entuziasty-razrabotchiki-kompilyatorov-i-ikh-proekty.php
27. Progopediya http://ru.progopedia.com/
28. https://www.levenez.com/lang/
29. https://psachin.gitlab.io/lisp-java-notes.html — sravneniye C++ vs Java vs Lisp
30. https://www.cs.unm.edu/~williams/cs257/lisp-study.html
31. https://renato.athaydes.com/posts/revisiting-prechelt-paper-comparing-languages.html.
32. Vivek Shangari http://rus-linux.net/MyLDP/algol/LISP/lisp01.html — Lisp: Slezy radosti, ch.1 "Lisp: Tears of Joy, Part 1"

33. https://habr.com/ru/articles/542068/ Pochemu ya ostayus' s Lispom (i vam tozhe stoit)
34. https://deep-econom.livejournal.com/237992.html Lisp
35. https://habr.com/ru/companies/sberbank/articles/655509/ Pervyy drevneyshiy: v chom unikal'nost' yazyka programmirovaniya LISP