Приднестровский государственный университет им. Т. Г. Шевченко

Физико-математический факультет

Кафедра прикладной математики и информатики

Реферат

по дисциплине «Спецсеминар»

ЯЗЫК SCHEME

Выполнил:

студент гр. 403 д/о

физ.-мат. факультета

Стефанишин Никита Алексеевич

Руководитель:

Ст. преподаватель

Кафедры информатики и ВТ

Великодный Вадим Игоревич

Тирасполь, 2017.

**Содержание**

Аннотация . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . . . . . . . . . . . . . . . . . 3

Введение . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .. . . . . . . . . . . . . . . . . . . 4

1. Что такое Scheme ? . . . . . . . . . . . . . . . . . . .

1.1 Семантики

1. Scheme Синтаксис
2. Практические примеры.. . . . . . . . . . . . . . . . . . . . . . . . . .

Заключение . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

Список литературы

Ссылки

**Аннотация**

**Введение**

Scheme – это функциональный язык программирования. Авторы языка Гай Стил и Джеральд Сассмен из Массачусетского технологического института – создали его в середине 1970-х годов. При разработке Scheme упор был сделан на элегантность и простоту языка. Философия языка подчёркнуто минималистская. В результате, Scheme содержит минимум примитивных конструкций и позволяет выразить все, что угодно путём надстройки над ними. В качестве примера можно указать, что язык использует 2 механизма организации циклов:

* хвостовая рекурсия,
* итеративный подход (в котором используется временные переменные для сохранения промежуточного результата).

Scheme начинался с попытки понять модель акторов  Карла Хьюитта, для чего Стил и Сассман написали «крошечный интерпретатор Лиспа», а затем «добавили механизм создания акторов и посылки сообщений». Scheme был первым диалектом Лиспа, применяющим исключительно статические (а не динамические) области видимости переменных, гарантирующим оптимизацию хвостовой рекурсии и поддерживающим данные булевского типа (#t и #f вместо традиционных T и NIL). Он также был одним из первых языков, непосредственно поддерживающих продолжения. Начиная со спецификации , язык приобрел исключительно мощное и удобное средство для записи макросов на основе шаблонов синтаксического преобразования с «соблюдением гигиены» (анг. *hygienic macro*). В Scheme также реализована «сборка мусора», то есть автоматическое освобождение памяти от неиспользуемых более объектов.

В качестве базовых структур данных язык использует списки и одномерные массивы («векторы»). В соответствии с декларируемым минимализмом, (пока) нет стандартного синтаксиса для поддержки структур с именованными полями, а также средств ООП — все это может быть реализовано программистом по его предпочтению, хотя большинство реализаций языка предлагают готовые механизмы.

Как курьёз, можно отметить, что первоначальное название языка Schemer было изменено на настоящее из-за ограничения на длину имён файлов в.

Хвостовая рекурсия — частный случай рекурсии, при котором любой рекурсивный вызов является последней операцией перед возвратом из функции. Подобный вид рекурсии примечателен тем, что может быть легко заменён на итерацию путём формальной и гарантированно корректной перестройки кода функции.

Что такое Scheme?

Scheme (схема) – маленький, исключительно «чистый» язык, который (что очень важно!) приятно использовать. Схема разрабатывалась таким образом, чтобы малое число универсальных конструкций можно было легко использовать в разных стилях программирования: функциональном, объектно-ориентированном и императивном. Стандарт языка занимает всего около 50(!) страниц, включая формальное определение семантики. Схема основывается на формальной модели лямбда – вычислений, так что здесь полно особенностей, удобных для теоретиков; это позволяет достаточно легко построить умные средства разработки программ.

Схема имеет механизм лексических областей видимости, однородные правила вычислений, и однородное понимание типов данных. Схема не имеет понятий указателя, неинициализированных переменных специальных циклических конструкций или детального управления хранением данных.

1.1. Семантика.

В этом разделе дается обзор семантики Scheme. Детальная неформальная семантика является предметом глав с 3 по 6. Для справочных целей в разделе 7.2 представлена ​​формальная семантика Scheme. Следуя Алголу, Scheme - это язык программирования с программным управлением. Каждое использование переменной связано с лексически очевидным связыванием этой переменной. В схеме есть скрытые, а не манифестные типы. Типы связаны со значениями (также называемыми объектами), а не с переменными. (Некоторые авторы ссылаются на языки со скрытыми типами как слабо типизированные или динамически типизированные языки.) Другими языками со скрытыми типами являются APL, Snobol и другие диалекты Lisp. Языки с манифестными типами (иногда называемые строго типизированными или статически типизированными языками) включают Algol 60, Pascal и C. Все объекты, созданные в ходе вычисления Схемы, включая процедуры и продолжения, имеют неограниченную степень. Объект Scheme никогда не уничтожается. Причина, по которой реализация Scheme (как правило!) Не исчерпывается хранения, заключается в том, что им разрешено вернуть занятое объектом хранилище, если они смогут доказать, что этот объект не может иметь значения для будущих вычислений. Другие языки, в которых большинство объектов имеют неограниченный объем, включают APL и другие диалекты Lisp. Реализации схемы должны быть правильно хвостовыми рекурсивными. Это позволяет выполнять итерационное вычисление в постоянном пространстве, даже если итерационное вычисление описывается синтаксически рекурсивной процедурой. Таким образом, при правильной реализации хвостовой рекурсии итерация может быть выражена с использованием обычной процедуры вызова процедур, так что специальные конструкторы итерации полезны только как синтаксический сахар. См. Раздел 3.5. Схемосхемы - это объекты, которые сами по себе. Процедуры могут быть созданы динамически, сохранены в структурах данных, возвращены как результаты процедур и т. д. Другие языки с такими свойствами включают Common Lisp и ML. Отличительной чертой Scheme является то, что продолжения, которые в большинстве других языков работают только «за кулисами», также имеют «первоклассный» статус. Продолжения полезны для реализации широкого спектра усовершенствованных управляющих конструкций, включая нелокальные выходы, возврат и сопрограммы. См. Раздел 6.4. Аргументы к процедурам схемы всегда передаются по значению, что означает, что фактические выражения аргумента оцениваются до того, как процедура получит контроль над природой, нужна ли процедуре результат оценки или нет. ML, C и APL - это три других языка, которые всегда передают аргументы по значению. Это отличается от семантики lazyevaluation для Haskell или семантики вызова по имени Algol 60, где выражение аргумента не вычисляется, если его значение не требуется в процедуре. Модель арифметики Схемы разработана так, чтобы оставаться максимально независимой от конкретных способов представления чисел в компьютере. В Схеме каждое целое число является рациональным числом, каждое рациональное - вещественным, а каждое реальное - комплексным числом. Таким образом, различие между целочисленной и вещественной арифметикой, столь важное для многих языков программирования, не появляется в схеме. На его месте делается различие между точной арифметикой, которая соответствует математическому идеалу, и неточной арифметикой аппроксимаций. Как и в Common Lisp, точная арифметика не ограничена целыми числами.

В Scheme можно использовать символы, знаки, строки списки, числа, логические значения, векторы, порты и функции.

Слово знак будет использоваться в смысле «печатный символ» (character).

**Синтаксис**

Язык программирования Scheme синтаксически близко к лямбда – исчисления.

**Е** в выражениях

**I** в идентификаторах (переменные)

**C** в Constants

Синтаксис Scheme

предельно проста: она состоит из круглых скобок (их много) и префиксной нотации. Это означает, что оператор приходит крайний левый, а затем операнды, например:

(+ 1 2 3 4)

Скобки используются для того, чтобы определить, что должно быть вычислено. Во многих языках, можно добавить дополнительные скобки, не оказывая влияния на программу, но это не так, на схеме. Добавление дополнительных круглых скобок, скорее всего, приведет к ошибкам или другого несчастного поведения.

Комментарии начинаются точкой с запятой и продолжаются до конца строки.

(+ 1 2 3 4); это комментарий

Синтаксис описывается формально

Практические примеры

*;The first program*

*(****begin***

*(display "Hello, World!")*

*(newline))*

Первая строка является комментарием. Когда схема видит точку с запятой, она игнорирует его и весь последующий текст на линии.

***begin***-Формаявляется способ схемы ее введения последовательностиподформ.В этом случае есть два субформы.Первым из нихявляется вызов*display* процедуры ,который выводит ее аргумент (строка*"Hello, World!"*) на консоль (или «стандартный вывод»).Это сопровождается*newline* вызовом процедуры, который выводит возврат каретки.

>(print “Hello”)

Hello

1

#t

#f

"foo"

'foo

read

;В слушающего строке загрузить файл \*.scm. Это делается путем ввода

(load "examples.scm")

'(1 2 3)

(quote (1 2 3))

(+ 3 (\* 4 5))

(cons 1 2)

(cons 1 '(2 3 4))

(cons '(1 2 3) '(4 5 6))

(cdr '(7 6 5))

(cdr '(it rains every day))

(list 'a)

(list 'a 'b 'c 'd 'e 'f)

(length '(1 3 5 9 11))

(reverse '(1 3 5 9 11))

(define n 5)

(set! n 3)

(if (> 3 2) 'foo 'bar)

((lambda (x) (\* x x)) 3)

((lambda (x y) (+ x y)) 3 4)

(define square (lambda (x) (\* x x)))

(define fac (lambda (n) (if (= n 0) 1 (\* n (fac (- n 1))))))

(define dbl (lambda (x) (\* 2 x)))

(map dbl '(1 2 3 4))

Заключение

Функции могут быть использованы для построения программ.

Новые функции можно конструировать на основе ранее определенных.

Программирование с помощью функций обеспечивает гибкость программ.

Список литературы

<http://o-ili-v.ru/wiki/Scheme>

<http://ilammy.github.io/lisp/>

<https://github.com/ilammy/lisp>

<http://5top100.ru/upload/iblock/85f/85f3f644ead9ca3fb0e76057242926ea.pdf>

<https://vk.com/topic-411354_21893515>

<http://ds26gte.github.io/tyscheme/index-Z-H-1.html>