Лекция 3.

Раздел 3. Основы функционального программирования на языках Lisp, Scheme и FP

Тема 7-9. Синтаксис Лиспа. Понятие S-выражения. Значение функции. Базовые функции Лиспа. Рекурсивные функции и их определение.
Тема 10. Язык Ским (SCHEME) как диалект языка Лисп. Реализация аппликативных операторов и комбинаторов на языке Ским.
Тема 11. Основные конструкции языка FP. Примитивные функции и комбинирующие формы.

План лекции.

1. Язык программирования Лисп.
2. Язык программирования Ским (SCHEME) как диалект языка Лисп.
3. Основные конструкции языка FP.

Основная часть.

Язык ЛИСП (LISP) был разработан в 1958 году американским ученым Джоном Маккарти как функциональный язык, предназначенный для обработки списков (LISt Processing).

В основу языка положен серьезный математический аппарат:

- лямбда-исчисление Черча
- алгебра списочных структур
- теория рекурсивных функций

Долгое время язык использовался узким кругом исследователей. Широкое распространение язык получил в конце 70-х - начале 80-х годов с появлением необходимой мощности вычислительных машин и соответствующего круга задач. В настоящем - Лисп одно из главных инструментальных средств систем искусственного интеллекта. Он принят как один из двух основных ЯП для министерства обороны США. система AutoCAD разработана на Лиспе.

Основу ЛИСПа составляют символьные выражения, которые называются **S-выражениями** и образуют область определения для функциональных программ.

S-выражение (Simbolic expresion) - основная структура данных в ЛИСПе.

(ДЖОН СМИТ 33 ГОДА) \

S-ВЫРАЖЕНИЯ

((МАША 21) (ВАСЯ 24) (ПЕТЯ 1)) /

S-выражение - это либо атом, либо список.

Атомы - это простейшие объекты Лиспа, из которых строятся остальные структуры. Атомы бывают двух типов - символьные и числовые. Символьные атомы - последовательность букв и цифр, при этом должен быть по крайней мере один символ отличающий его от числа. Символ как правило обозначает какой-либо предмет, объект вещь, действие. Символьный атом рассматривается как неделимое целое. К символьным атомам применяется только одна операция: сравнение. В MCL в состав символа могут входить: **+ - * / @ $ % ^ _ \ <>**

Числовые атомы - обыкновенные числа

124

-344

4.5 3.055E8

Числа - это константы. Типы чисел зависят от реализации ЛИСПа.

Атом - простейшее S-выражение.

Термин **S-выражение** относится к соглашению о способе записи полуструктурированных данных в доступной для человеческого понимания текстовой форме. Символические выражения создаются, в основном, из символов и списков. S-выражения наиболее известны благодаря их использованию в языках программирования семейства Лисп. В ЛИСПЕ список - это последовательность элементов (list). Элементами являются или атомы, или списки. Списки заключаются в скобки, элементы списка разделяются пробелами. Список, в котором нет ни одного элемента, называется пустым списком и обозначается "()" или символом NIL.

NIL обозначает кроме этого, в логических выражениях логическую константу "ложь" (false). Логическое "да"(true) задается символом "T". Атомы и списки - это символьные выражения или S-выражения.

Scheme (СКИМ) - диалект языка программирования Лисп. Scheme был разработан Гаем Стилом (Guy L. Steele) и Джеральдом Сассменом (Gerald Jay Sussman). Основные особенности Scheme:

- минимализм языка, основанный на лямбда-исчислении, которое используется для получения значительной части синтаксиса языка (11 из 23 синтаксических конструкций) из более примитивных конструкций.
- статическая лексическая область видимости: имя переменной относится к самой локальной области видимости; таким образом, код можно читать и интерпретировать вне зависимости от того, в каком контексте он будет вызываться.
- блоки, выражающиеся тремя конструкциями let, let* и letrec.

- “правильная” хвостовая рекурсия, позволяющая записывать итеративные алгоритмы более идиоматично, через рекурсию, и при этом оптимизирующая их так, чтобы поддерживать неограниченное количество активных вызовов.
- продолжения (абстрактные представления состояний программы) как объекты первого класса (процедура call-with-current-continuation).
- общее пространство имен для переменных и процедур.

В качестве базовых структур данных язык использует списки и одномерные массивы («векторы»). Несколько базовых принципов языка.

1. Круглые скобки. Любое законченное выражение должно быть заключено в них. Это отпугивает тех, кто видит код впервые, но впоследствии на практике не вызывает сложностей.

2. Никаких дополнительных служебных символов. Хватит скобок. Точка с запятой, кстати, отделяет от кода комментарии.

3. Построение конструкций по типу «действие-предмет». В языке программирования это смотрится необычно, но в переводе на естественный язык вполне понятно.

FP («Functional Programming») - алгебраический язык программирования. Изобретен Джоном Бэкусом (John Backus), дальнейшей стандартизации FP не подвергался. Был задуман как модель языка, поддерживающего стиль программирования на уровне функций, больше как математическая модель, чем как действующий язык. Не используется при разработке реальных программных комплексов. Тем не менее, концепции, воплощенные в FP, сильно повлияли на создание других функциональных языков. FP состоит из трех групп сущностей:

1. Атомы. FP использует следующие типы атомов:

- скалярные значения.
- последовательности.
- значение «неопределенность».

2. Функции могут быть элементарными или определенными программистом. Элементарные функции FP включают следующие:

- тождество: id:x возвращает x.
- константа: %x:y возвращает x для любого значения y.
- математические +,-,*,/: +:<x y> возвращает x+y, если x и y - скалярные значения.
- выборочная функция (селектор): i:<x_1 ... x_n> возвращает x_i, если 1 <= i <= n.
- сравнения =,>,<:=:<x y> возвращает T, если x=y, и F в противном случае.

- ряд векторных и матричных функций: конкатенация, присоединение в начало/конец, циклическая перестановка, транспонирование и т.д.

Определение функции программистом имеет следующий синтаксис: { functionName (function code) }. После определения функция вызывается так же, как элементарные функции.

3. Функциональные формы не могут определяться программистом, т.е. используются только формы, встроенные в язык. Список стандартных форм FP включает в себя следующие:

- составление: $[f_1, ..., f_n]:x$ эквивалентно $<f_1:x ... f_n:x>$.
- композиция: f @ g:x эквивалентно f:(g:x).
- условный выбор записывается как (condition -> trueChoice ; falseChoice) и применяется к атому x по следующему правилу: сначала вычисляется функция condition:x; если ее результат - T, возвращается trueChoice:x, иначе - falseChoice:x. Все три аргумента условного выбора - функции.
- применить ко всем: $@f:<x_1 ... x_n>$ эквивалентно $<f:x_1 ... f:x_n>$.
- правая вставка: $!f:<x_1 x_2 ... x_n>$ эквивалентно $f:<x_1 !f:<x_2 ... x_n>>$ (и применяется рекурсивно); !f:<x> эквивалентно x (левая вставка вычисляется аналогично).