Базовые функции языка Лисп.

Лекция 2.

Специальности : 230105, 010501

Основные определения.

Определение 1. **Символом называется отличное от числа имя, состоящее из букв, цифр и специальных знаков, которое обозначает некоторый объект реального мира.**

Определение 2. **Атомарными объектами или просто атомами в функциональных языках называются простейшие объекты, из которых строятся остальные структуры.**

Определение 3. **К константам в функциональном программировании относят логические значения T и NIL. Иногда к множеству констант относят также числа. В этом случае говорят о способности символа обозначать объекты. Если символ может обозначать более одного объекта, то его следует рассматривать как переменную.**

Определение 4. **S-выражениями называют символьные выражения, которые образуют области определений и области значений функций Лиспа.**

Виды S-выражений.

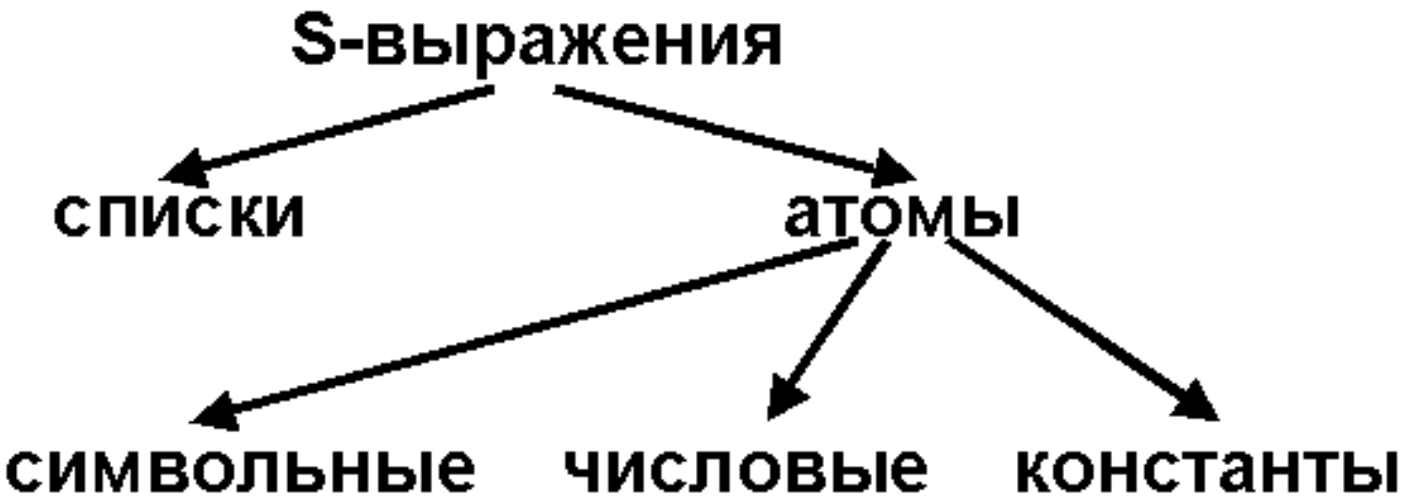

Рис.1.

Списки как средство представления знаний.

Атомы **и** ***списки*** **являются** ***основными*** **типами данных в Лиспе.**

Определение. **Списком в Лиспе называется упорядоченная последовательность S-выражений, заключенная в круглые скобки. Элементы списка отделяются друг от друга пробелами.**

Пример :

(a b c) – список из трех элементов,

((a b c)) – список из одного элемента,

((a b) c) – список из двух элементов.

Одной из отличительных особенностей Лиспа является единая-форма представления данных и программного кода (функций). Для обозначения списка, используемого как данные, и блокировки вычислений в Лиспе определена функция QUOTE (в newLISP-tk quote пишется строчными буквами) :

(QUOTE (a b c)) ⇔ ‘(a b c)

(a b c) – вызов функции с именем a и фактическими параметрами b,c.

Встроенные функции Лиспа.

- **Арифметические (+, -, *, /)**
- **Логические (AND, OR, NOT)**

Деление производится с точностью до седьмого знака. Пример :

$ (/ 1 3) Результат :

0.3333333

Поскольку Лисп является декларативным языком, то в нем используется предикативная форма записи. Для сравнения :

Паскаль.	**Лисп.**
a + b	**(+ a b)**
(a + b)/(b - c)	**(/ (+ a b)(- b c))**
a and b and c	**(and a b c)**

Базовые функции обработки списков.

В Лиспе для построения, разбора и анализа списков существуют простые базовые функции, к которым сводятся символьные вычисления и которые в нотации данного языка следует рассматриваются как система аксиом (алгебра обработки списков). К базисным функциям обработки символьных выражений в muLISP'е относят :

- **Функции-селекторы CAR и CDR;**
- **Функцию-конструктор CONS;**
- **Предикаты ATOM, EQ, EQUAL.**

Селекторы в muLISP’e

Определение. **Селектором называется функция, осуществляющая выборку элемента объекта данных.**

Функция CAR возвращает в качестве значения голову списка :

Функция CDR возвращает в качестве значения хвост списка.

Примеры :

Список	CAR	CDR
(a b c)	a	(b c)
((a)(b c))	(a)	((b c))
(a)	a	nil
()	nil	nil

Конструктор.

Конструктором называется функция, осуществляющая построение объекта данных. Функция CONS строит новый список из переданных ей в качестве аргументов произвольного s-выражения и списка. При этом первый аргумент включается в список-результат в качестве головы, второй – в качестве хвоста.

Примеры

S-выражение	список-аргумент	результат
(a)	(b c)	((a) b c)
(a)	NIL	((a))
NIL	(b c)	(NIL b c)
a	NIL	(a)
a	b	(a.b)

Связь между конструкторами и селекторами.

Функции CAR и CDR являются обратными для конструктора CONS.

Пример.

(CONS (CAR ‘(a b c))(CDR ‘(a b c))) дает исходный список ‘(a b c)

Композиция селекторов в muLISP’e.

Путем комбинации селекторов CAR и CDR можно выделять произвольный элемент списка. В Лиспе допускается сокращенная запись композиции нескольких селекторов (в muLISPe – не более четырех) в виде одного вызова функции.

Пример.

Дан список списков :

‘((a b)(1 2)(3 4 5 6)(7 8 9) 10)

Выделить второй элемент третьего подсписка.

Решение.

Выделяем третий подсписок :

(CAR (CDR (CDR ‘((a b)(1 2)(3 4 5 6)(7 8 9) 10))))

Выделяем второй элемент третьего подсписка :

(CAR (CDR (CAR (CDR (CDR ‘((a b)(1 2)(3 4 5 6)(7 8 9) 10))))))

Сокращенная запись : (CADAR (CDDR ‘((a b)(1 2)(3 4 5 6)(7 8 9) 10)))

Конструирование списков в Лиспе.

Помимо примитивных функций CAR, CDR и CONS списочного ассемблера (уровня ячеек памяти виртуальной Лисп-машины), для построения списков в Лиспе существуют функции более высокого уровня.

Функция list создает список из произвольных элементов. Вызов list эквивалентен суперпозиции вызовов CONS, причем вторым аргументом последнего вызова является nil, который служит основой для наращивания списка.

Пример :

(list ‘a ‘b ‘c) эквивалентно

(cons ‘a (cons ‘b (cons ‘c nil))) в muLISP’e, либо

(cons 'a (cons 'b (cons 'c '()))) в newLISP-tk

Выделение элемента списка.

Для реализуемой с помощью селекторов и их суперпозиции выборки элементов списков в Коммон Лиспе и muLISP'е существуют функции выделения заданного элемента : FIRST, SECOND, THIRD, FOURTH, … и более общая функция NTH, выделяющая n-й элемент списка : (nth *n список*). Следует помнить, что функция nth отсчет элементов списка производит с нуля. В newLISP-tk реализованы first и nth.

Пример :

(nth 5 '(1 2 3 4 5)) дает nil, (nth 4 '(1 2 3 4 5)) дает 5.

Последний элемент списка можно выделить с помощью функции LAST. При этом вызов (last *список*) в muLISP'е дает последний непустой хвост списка *список*.

Пример :

(last '(1 2 3)) дает (3)

(car (last '(1 2 3))) возвращает последний элемент списка, то есть 3.

Предикатные функции в muLISP'e.

Определение. **Предикатами в функциональном программировании называются функции, которые проверяют выполнение некоторого условия и возвращают логическое значение T или NIL.**

ATOM, EQ и EQUAL являются базовыми предикатами Лиспа. С их помощью и используя другие базовые функции, можно задавать более сложные предикаты, которые будут проверять наличие более сложных свойств.

Предикат ATOM проверяет, является ли аргумент атомом.

(atom *s-выражение*) возвращает T, если *s-выражение* является атомом, NIL – в противном случае.

Предикат EQ проверяет тождественность двух атомарных s-выражений.

(eq 'a 'b) дает NIL, (eq 'a 'a) дает T, (eq a a) дает T, (eq 1 1) дает T,

но (eq '(1 2) '(1 2)) дает NIL.

Предикат EQUAL проверяет тождественность произвольных s-выражений.

Примеры : (equal '(1 2) '(1 2)) дает T, (equal '(1 2) '(2 1)) дает NIL.

Следует отметить, что (EQUAL *список* NIL)⇔(NULL *список*)

Реализация базовых функций в newLISP-tk.

Таблица 1.

Соответствие базовых функций в newLISP-tk базовым функциям muLISP'a.

muLISP	newLISP-tk
car	first
cdr	rest
cons	cons
atom	atom?
equal	=

Таблица 2.

Реализация расширения базовых функций muLISP'a в newLISP-tk.

muLISP	newLISP-tk
nth	nth
last	last
null	null?
list	list