

ВЫЧИСЛЕНИЕ В ЛИСПЕ

Функциональное
программирование
Григорьева И.В.

- Программа состоит из форм и функций
- Управляющие структуры Лиспа являются формами
LET создает локальную связь
- Последовательные вычисления: PROG1, PROG2 и PROGN
- Разветвление вычислений: условное предложение
COND
- Другие условные предложения: IF, WHEN, UNLESS и CASE
- Циклические вычисления: предложение DO
- Предложения PROG, GO и RETURN
- Другие циклические структуры
- Повторение через итерацию или рекурсию
- Формы динамического прекращения вычислений:
CATCH и THROW

Программа состоит из форм и функций

Под *формой* (form) понимается такое символьное выражение, значение которого может быть найдено, интерпретатором. Ранее мы уже использовали наиболее простые формы языка: константы, переменные, лямбда-вызовы, вызовы функций и их сочетания. Кроме них были рассмотрены некоторые специальные формы, такие как QUOTE и SETQ, трактующие свои аргументы иначе, чем обычные функции. Лямбда-выражение без фактических параметров не является формой.

Вычислимые выражения можно разделить на три группы:

1. *Самоопределенные* формы. Эти формы, подобно константам, являются лисповскими объектами, представляющими лишь самих себя. Это такие формы, как числа и специальные константы Т и NIL, а также знаки, строки и битовые векторы.
2. Символы, которые используются в качестве переменных.

3. Формы в виде списочной структуры, которыми являются:

1. Вызовы функций и лямбда-вызовы.
2. *Специальные формы*, в число которых входят SETQ, QUOTE и многие описанные в этой главе формы, предназначенные для управления вычислением и контекстом.
3. *Макроподпрограммы* (будут рассмотрены позже).

У каждой формы свой синтаксис и семантика, основанные на едином способе записи и интерпретации.

Управляющие структуры Лиспа являются формами

Управляющие структуры Лиспа (*предложения*) выглядят внешне как вызовы функций. Предложения будут записываться в виде скобочных выражений, первый элемент которых действует как имя управляющей структуры, а остальные элементы - как "аргументы". Результатом вычисления, так же как у функции, является значение, т.е. управляющие структуры представляют собой формы. Однако предложения не являются вызовами функций, и разные предложения используют аргументы по-разному.

Наиболее важные с точки зрения
программирования синтаксические
формы можно на основе их использова-
ния разделить на следующие группы:

- Работа с контекстом:
 - QUOTE или блокировка вычисления;
 - вызов функции и лямбда-вызов;
 - предложения LET и LET*.
- Последовательное выполнение:
 - предложения PROG1, PROG2 и PROGN.

- Разветвление вычислений:
 - условные предложения COND, IF, WHEN, UNLESS;
 - выбирающее предложение CASE.
- Итерации:
 - циклические предложения DO, DO*, LOOP, DOTIMES, DOUNTIL
- Передачи управления:
 - предложения PROG, GO и RETURN.
- Динамическое управление вычислением:
 - THROW и CATCH, а также BLOCK.

LET создает локальную связь

Вычисление вызова функции создает на время вычисления новые связи для формальных параметров функции. Новые связи внутри формы можно создать и с помощью предложения LET. Эта структура выглядит так:

(LET ((*m1* знач1) (*t2* знач2) ...) *форма1*
форма2...)

Предложение LET вычисляется так, что сначала статические переменные m_1, m_2, \dots из первого "аргумента" формы связываются с соответствующими значениями $\text{знач}_1, \text{знач}_2, \dots$. Затем слева направо вычисляются значения форм $\text{форм}_1, \text{форм}_2, \dots$. В качестве значения всей формы возвращается значение последней формы. Как и у функций, после окончания вычисления связи статических переменных m_1, m_2, \dots ликвидируются и любые изменения их значений (SETQ) не будут видны извне.

Например:

_(setq x 2)

2

_(let ((x 0)) (setq x 1))

1

_x

2

Форма LET является на самом деле синтаксическим видоизменением лямбда-вызыва, в которой формальные и фактические параметры помещены совместно в начале формы:

(LET ((*m1 at*) (*m2 a2*) ... (*mn an*)) *форма1 форма2* ...)

((LAMBDA
(*m1 m2 ... mn*) ; формальные параметры
форма1 форма2 ...) ; тело функции
a1 a2 ... an) ; фактические параметры

Значения переменным формы LET присваиваются *одновременно*. Это означает, что значения всех переменных m_i вычисляются до того, как осуществляется связывание с формальными параметрами. Новые связи этих переменных еще не действуют в момент вычисления начальных значений переменных, которые перечислены в форме позднее. Например:

```
_ (let ((x 2) (y (* 3 x)))  
  (list x y)) ; при вычислении Y  
Error: Unbound atom X ; у X нет связи
```

Побочный эффект можно наблюдать при работе с формой LET* подобной LET, но вычисляющей значения переменных *последовательно*:

```
_ (let* ((x 2) (y (* 3 x)))  
  (list x y))  (2 6)
```

Последовательные вычисления: PROG1, PROG2 и PROGN

Предложения PROG1, PROG2 и PROGN позволяют работать с некоторыми вычисляемыми формами:

(PROG1 *форма1* *форма2* ... *формaN*)

(PROG2 *форма1* *форма2* ... *формaN*)

(PROGN *форма1* *форма2* ... *формaN*)

У этих специальных форм переменное число аргументов, которые они последовательно вычисляют и возвращают в качестве значения значение первого (PROG1), второго (PROG2) или последнего (PROGN) аргумента. Эти формы не содержат механизма определения внутренних переменных:

(progn (setq x 2) (setq y (* 3 x))))

6

x

2

Разветвление вычислений: условное предложение COND

Предложение COND является основным средством разветвления вычислений. Это синтаксическая форма, позволяющая управлять вычислениями на основе определяемых предикатами условий.

Структура условного предложения такова:

(COND (p1 a1) (p2 a2) ... (pN aN))

Предикатами r_i и результирующими выражениями a_i могут быть произвольные формы. Значение предложения COND определяется следующим образом:

1. Выражения r_i , выполняющие роль предикатов, вычисляются последовательно слева направо (сверху вниз) до тех пор, пока не встретится выражение, значением которого не является NIL, т.е. логическим значением которого является истина.
2. Вычисляется результирующее выражение, соответствующее этому предикату, и полученное значение возвращается в качестве значения всего предложения COND.
3. Если истинного предиката нет, то значением COND будет NIL.

Рекомендуется в качестве последнего предиката использовать символ Т, и соответствующее ему результирующее выражение будет вычисляться всегда в тех случаях, когда ни одно другое условие не выполняется. В следующем примере с помощью предложения COND определена функция, устанавливающая тип выражения:

```
_ (defun тип(l) (cond ((null l) 'пусто) ((atom l)
  `атом) (t 'список)))
```

ТИП

_ (тип '(a b c))

СПИСОК

_ (тип (atom '(a t o m)))

ПУСТО

В условном предложении может отсутствовать результирующее выражение *ai* или на его месте часто может быть последовательность форм:

(COND (*p1 a1*)

....

(*pi*)

; результирующее

....

(*pk ak1 ak2... akN*)

; выражение отсутствует

...)

; последовательность форм

; в качестве результата

Если условию не ставится в соответствие результирующее выражение, то в качестве результата предложения COND при истинности предиката выдается само значение предиката. Если же условию соответствует несколько форм, то при его истинности формы вычисляются последовательно слева направо и результатом предложения COND будет значение последней формы последовательности (неявный PROGN).

_(defun И (x y) (cond (x y) (t nil)))

И

_(и t nil)

NIL

_(defun или (x y) (cond (x t) (t y)))

ИЛИ

_(или t nil)

T

_(defun не (x) (not x))

НЕ

_(не t)

NIL

_(defun => (x y) (cond (x y) (t t)))

=>

_(=> nil t)

T

$_\!(\text{defun } => (\mathbf{x} \mathbf{y}) (\text{или } \mathbf{x} (\text{не } \mathbf{y})))$

$=>$

$_\!(\text{defun } \Leftrightarrow (\mathbf{x} \mathbf{y}) (\text{и } (=> \mathbf{x} \mathbf{y}) (=> \mathbf{y} \mathbf{x})))$

\Leftrightarrow

Предикаты "и" и "или" входят в состав встроенных функций Лиспа и называются AND и OR. Число их аргументов может быть произвольным.

(and (atom nil) (null nil) (eq nil nil))

Т

Предикат AND в случае истинности возвращает в качестве значения значение своего последнего аргумента. Его иногда используют как упрощение условного предложения по следующему образцу:

(AND условие₁ условие₂ ... условие_N)

↔

**(COND ((AND условие₁ условие₂ ... условие_N)
(T NIL)))**

Предложения COND можно комбинировать таким же образом, как и вызовы функций. Например, предикат "исключающее или" (exclusive or или xor), который ложен, когда оба аргумента одновременно либо истинны, либо нет, можно определить следующим образом:

```
_ (defun xor(x y)
      (cond (x (cond (y nil)
                      (t t)))
            (t y)))
```

XOR

Другие условные предложения: IF, WHEN, UNLESS и CASE

(IF *условие то-форма иначе-форма*)



(COND (*условие то-форма*)
(Т иначе-форма))

(if (atom t) 'атом 'список)

АТОМ

(WHEN *условие форма1 форма2 ...*)

↔

(UNLESS (**NOT** *условие*) *форма1 форма2 ...*)

↔

(COND (*условие форма1 форма2 ...*))

↔

(IF *условие* (**PROGN** *форма1 форма2 ...*)
NIL)

(CASE *ключ*

(*список-ключей1* *m11 m12 ...*)

(*список-ключей2* *m21 m22 ...*))

Сначала в форме CASE вычисляется значение ключевой формы *ключ*. Затем его сравнивают с элементами списков ключей *список-ключей*, с которого начинаются альтернативы. Когда в списке найдено значение ключевой формы, начинают вычисляться соответствующие формы *mi1, mi2, ...,* значение последней из которых и возвращается в качестве значения всего предложения CASE (неявный PROGN).

Повторение через итерацию или рекурсию

В "чистом" функциональном Лиспе нет ни циклических предложений (DO, PROG и другие), ни тем более операторов передачи управления. Для программирования повторяющихся вычислений в нем используются лишь условные предложения и определения *рекурсивных*, или вызывающих самих себя, функций.

```
_ (defun ! (n)
  (if (= n 0) 1
      (* n (! (- n 1)))))

!
(! 5)
120
```

Упражнения

1. Запишите следующие лямбда-вызывы с использованием формы LET и вычислите их значения на машине:

- $((lambda (x y) (list x y)) (+ 2 3) 'c)$
- $((lambda (x y) ((lambda (z) (list x y z)) 'c))$
- $((lambda (x y) (list x y))$
- $((lambda (z) z) 'a) 'b)$

2. С помощью предложений COND или CASE определите функцию, которая возвращает в качестве значения столицу заданного аргументом государства:
(столица 'Финляндия) ХЕЛЬСИНКИ

3. Предикат сравнения ($> x y$) истинен, если x больше, чем y . Опишите с помощью предиката $>$ и условного предложения функцию, которая возвращает из трех числовых аргументов значение среднего по величине числа:

_(среднее 4 7 6) 6

4. Можно ли с помощью предложения COND запрограммировать предложение IF как функцию?

5. Определите функцию (ПРОИЗВЕДЕНИЕ n m), вычисляющую произведение двух целых положительных чисел.
6. Функция (LENGTH x) является встроенной функцией, которая возвращает в качестве значения длину списка (количество элементов на верхнем уровне). Определите функцию LENGTH рекурсивно с помощью предложения COND.

6. В математике числа Фибоначчи образуют ряд 0, 1, 1, 2, 3, 5, 8, Эту последовательность можно определить с помощью следующей функции FIB:

$$fib(n) = 0 \quad , \text{ если } n=0$$

$$fib(n) = 1 \quad , \text{ если } n=1$$

$$fib(n) = fib(n-1)*fib(n-2) \quad , \text{ если } n>1$$

Определите лисповскую функцию fib(n), вычисляющую n-й элемент ряда Фибоначчи.

7. Определите функцию ДОБАВЬ,
прибавляющую к
элементам списка данное число:

_ (добавь '(2 7 3) 3)
(5 10 6)