ДНЕПРОПЕТРОВСКАЯ ОБЛАСТНАЯ АДМИНИСТРАЦИЯ

ИНСТИТУТ ПРЕДПРИНИМАТЕЛЬСТВА «СТРАТЕГИЯ»

КАФЕДРА ЕКНОМИЧЕСКОЙ КИБЕРНЕТИКИ И

ПРОГРАМНОГО ОБЕСПЕЧЕНИЯ

**ДОКДАД**

**ПО ДИСЦИПЛИНЕ:**

**«ФУНКЦИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ»**

**НА ТЕМУ:**

**«ФОРМАТИРОВАННЫЙ ВВОД »**

Выполнил:

студент группы П-01-51

Руденко Владимир

2003г.

г. Желтые Воды

Если не использовать специальную команду ввода, то данные можно передавать лисповской функции только через параметры и свободные переменные. Соответственно, без использования выводы, результат можно получить лишь через конечное значение выражения. Часто все же возникает необходимость вводить исходные данные и выдавать сообщения т тем и самым управлять и получать промежуточные результаты во время вычислений, как это делается и в других языках программирования.

Лисповская функция чтения READ отличается от ввода в других языках программирования тем, что она обрабатывает выражение целиком, а не отдельные элементы данных. Вызов этой функции осуществляется пользователем (немного упрощенно) в виде:

(READ)

Как только интерпретатор встречает предложение READ, вычисления приостанавливаются до тех пор, пока пользователь не введет какой-нибудь символ или целиком выражение.

READ никак не показывает, что оно ждет ввода выражения. Программист должен сам сообщить об этом при помощи функций ввода. READ лишь читает выражение и возвращает в качестве значения само это выражение, после чего вычисления продолжаются.

По своему действию READ представляет собой функцию, но у нее есть побочный эффект, состоящий именно во вводе выражения. Учитывая это, READ является не чистой функцией, а псевдо функцией.

Если прочитанное значение необходимо сохранить для дальнейшего использования, то вызов READ должен быть аргументом какой-нибудь формы, например присваивания (SETQ), которая свяжет полученное выражение:

\_(setq input (read))

(+ 2 3); введенное выражение

(+ 2 3); значение

\_input

(+ 2 3)

Форма, вызывающая интерпретатор, и функция READ совместно с другими функциями позволяет читать выражения внешние по отношению к программе. Из них можно строить новые лисповские выражения или целые программы. Построенные структуры можно вычислить, передав их непосредственно интерпретатору:

\_(eval input)

5

\_(eval (list (read) (read) (read)))

+ 2 3

5

Функция READ основана на работающей на системном уровне процедуре чтения (Lisp reader). Она читает выражение, образуемое последовательностью знаков, поступающих из файла или иного источника. Внешние устройства становятся доступными из Лисп – системы через объекты, называемые потоками. На логическом уровне потоки независимо от характера внешнего устройства являются последовательностью читаемых или записываемых знаков или битов. Для ввода и вывода, как и для двустороннего обмена, существуют свои типы потоков и специальные операции.

Процедура чтения содержит анализатор, проверяющий знаки в читаемой им последовательности. Чтение обычного алфавитно-цифрового знака никаких особых действий не требует, в то время как чтение специального знака, такого как открывающая или закрывающая скобка, пробел, разделяющий элементы. Или точка, приводит к специальным действиям. Соответствие между различными знаками и действиями определяется так называемой таблицей чтения, которая задает лисповские функции для знаков. Знаки, вызывающие специальные действия, называют макрознаками или макросами чтения, поскольку их чтение требует более сложных действий. Таблица чтения доступна программисту, и он может сам определять новые интерпретации знаков и, таким образом, расширять или изменять синтаксис Лиспа.

Действие макроса – чтения определяется в Лиспе при помощи обыкновенной функции. Она читает и возвращает в качестве значения форму, для построения которой она в свою очередь может предварительно использовать макросы. Определим для примера макрос чтения %, действующий так же, как апостроф. Действие блокировки вычисления пользователь может определить в виде функции, которая рекурсивно читает очередное выражение и возвращает его в составе формы QUOTE:

\_(defun f () (stream char)

(list ‘quote (read)))

Функция, определяющая макрос чтения, имеет в Лиспе два аргумента, первый из которых описывает отток чтения, а значением второго будет сам макрознак. В данном примере мы не использовали параметры.

Запись символов и определенных для них макроинтерпретаций в таблицу чтения осуществляется командой:

(SET-MACRO-CHARACTER знак функция)

\_(set-macro-character

#\% ‘f)

Здесь запись #\% обозначает знак процента (%) как объект с типом данных знак.

После этих определений можно знак процента использовать так же, как апостроф:

\_%(a %b c)

(A (QUOTE B) C)

Таблиц чтения может быть несколько, но процедура чтения использует в каждый момент времени лишь одну таблицу. Текущая таблица сохраняется как значение системной переменной \*READTABLE\*.

Встроенными макросами чтения в Коммон Лиспе являются:

( ; начинает ввод списка или точечной пары

) ; заканчивает ввод списка или точечной пары

' ; возвращает очередное выражение в виде вызова QUOTE

; ; символы до конца строки считаются комментарием

\ ; выделение одиночного специального знака

| ; выделение нескольких специальных знаков

; |…|

“ ;строка: “…”

Макрознаки нельзя использовать в составе символов наподобие обычных знаков, поскольку процедура чтения проинтерпретирует их в соответствии с таблицей как макросы чтения. Для включения таких знаков в состав имен нужно использовать специальные выделяющие знаки: |, \, которые блокируют макрообработку знаков.

**ПРИМЕРЫ**

Чтение строки

Вводим предложение, заканчивающееся вопросительным или восклицательным знаком, и преобразуем его в список:

(defun readf\_ ()

(let ((word (read)))

(cond ((eq word ‘?) ‘(?))

(t (cons word (readf\_))))))

**Ввод в режиме EVALQUOTE**

Функция READ, которая вводит вызов функции в виде: fn(a1 a2 … an) и возвращает значение вызова

(defun readq\_ ()

(eval (cons (read) (read))))