**Лабораторная работа №3.**

**Тема:** Организация вычислений в Лиспе.

**Цель:** Изучить основные функции и их особенности для организации вычислений в Лиспе.

1. Предложения LET и LET\*.

Предложение LET создает локальную связь внутри формы:

(LET ((m1 знач1) (m2 знач2)...)

форма1 форма2 ...)

Вначале статические переменные m1, m2, ... связываются (одновременно) с соответствующими значениями знач1, знач2, ... . Затем слева на право вычисляются значения формы1, формы2, ... . Значение последней формы возвращается в качестве значения всей формы. После вычисления связи статических переменных ликвидируются.

Предложения LET можно делать вложенными одно в другое.

\_(LET ((x ‘a) (y ‘b))

(LET ((z ‘c)) (LIST x y z))) ⇨ (a b c)

\_(LET ((x (LET ((z ‘a)) z)) (y ‘b))

(LIST x y)) ⇨ (a b)

\_(LET ((x 1) (y (+ x 1)))

(LIST x y)) ⇨ ERROR

При вычислении у У и Х еще нет связи. Значения переменным присваиваются одновременно. Это означает, что значения всех переменных mi вычисляются до того, как осуществляется связывание с формальными параметрами.

Подобной ошибки можно избежать с помощью формы LET\*:

\_(LET\* ((x 1) (y (+ x 1)))

(LIST x y)) ⇨ (1 2)

2. Последовательные вычисления.

Предложения PROG1 и PROGN позволяют работать с несколькими вычисляемыми формами:

(PROG1 форма1 ... формаN)

(PROGN форма1 ... формаN)

Эти специальные формы последовательно вычисляют свои аргументы и в качестве значения возвращают значение первого (PROG1) или последнего (PROGN) аргумента.

\_(PROG1 (SETQ x 1) (SETQ y 5)) ⇨ 1

\_(PROGN (SETQ j 8) (SETQ z (+x j))) ⇨ 9

3. Разветвление вычислений.

Условное предложение COND:

(COND (p1 a1)

...

(pn an))

Предикатами pi и результирующими выражениями ai могут быть произвольные формы. Выражения pi вычисляются последовательно до тех пор, пока не встретится выражение, значением которого является T. Вычисляется результирующее выражение, соответствующее этому предикату, и полученное значение возвращается в качестве значения всего предложения COND. Если истинного предиката нет, то значением COND будет NIL.

Рекомендуется в качестве последнего предиката использовать символ T. Тогда соответствующее ему an будет вычисляться в том случае, если другие условия не выполняются.

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

Предложения COND можно комбинировать.

(COND ((> x 0) (SETQ рез x))

((< x 0) (SETQ x -x) (SETQ рез х))

((= х 0))

(Т ‘ошибка))

Предложение IF.

(IF условие то-форма иначе-форма)

(IF (> x 0) (SETQ y (+ y x)) (SETQ y (- y x)))

Если выполняется условие (т. е. х>0), то к значению y прибавляется значение х, иначе (x<0) от y отнимается отрицательное значение х, т. е. прибавляется абсолютное его значение.

Можно использовать форму WHEN.

(WHEN условие форма1 форма2 ... )

Выбирающее предложение CASE^

(CASE ключ

(список-ключей1 m11 m12 ... )

(список-ключей2 m21 m22 ... )

....)

Сначала вычисляется значение ключевой формы - ключ. Затем его сравнивают с элементами списка-ключейi. Когда в списке найдено значение ключевой формы, начинают вычисляться соответствующие формы mi1, mi2, ... . Значение последней возвращается в качестве значения всего предложения CASE.

\_(SETQ ключ 3) ⇨ 3

\_(CASE ключ

(1 ‘one)

(2 ‘(one + one) ‘two)

(3 ‘(two + one) ‘three) ⇨ three

4. Циклические вычисления.

Предложение DO.

(DO ((var1 знач1 шаг1) (var2 знач2 шаг2) ...)

(условие-окончания форма11 форма12 ...)

форма21 форма22 ...)

Первый аргумент описывает внутренние переменные var1, var2, ..., их начальные значения - знач1, знач2, ... и формы обновления - шаг1, шаг2, ....

Вначале вычисления предложения DOI внутренним переменным присваиваются начальные значения, если значения не присваиваются, то по умолчанию переменным присваивается NIL. Затем проверяется условие-окончания. Если оно действительно, то последовательно выполняются формы1i и значение последней возвращается в качестве значения всего предложения DO, иначе последовательно вычисляются формы2i.

На следующем цикле переменным vari одновременно присваиваются значения форм - шагi, вычисляемых в текущем контексте, проверяется условие-окончания и т. д.

\_(DO ((x 5 (+ x 1)) (y 8 (+ y 2)) (рез 0))

((< x 10) рез)

(SETQ рез (+ рез x y))

5. Передача управления.

На Лиспе можно писать программы и в обычном операторном стиле с использованием передачи управления. Однако во многих системах не рекомендуется использовать эти предложения, так как их можно заменить другими предложениями (например DO) и, как правило, в более понятной форме. Но мы рассмотрим предложения передачи управления, хотя использовать их не следует.

(PROG (m1 m2 ... mn)

оператор1

оператор2

...

операторm)

Перечисленные в начале формы переменные mi являются локальными статическими переменными формы, которые можно использовать для хранения промежуточных результатов. Если переменных нет, то на месте списка переменных нужно ставить NIL. Если какая-нибудь форма операторi является символом или целым числом, то это метка перехода. На такую метку можно передать управление оператором GO:

(GO метка)

GO не вычисляет значение своего «аргумента».

Кроме этого, в PROG-механизм входит оператор окончания вычисления и возврата значения:

(RETURN результат)

Операторы предложения PROG вычисляются слева направо (сверху вниз), пропуская метки перехода. Оператор RETURN прекращает выполнение предложения PROG; в качестве значения всего предложения возвращается значение аргумента оператора PROG. Если во время вычисления оператор RETURN не встретился, то значением PROG после вычисления его последнего оператора станет NIL .

После вычисления значения формы связи программных переменных исчезают.

**Задания к лабораторной работе.**

1. Запишите следующие лямбда-вызовы с использованием формы LET и вычислите их:

a) ((LAMBDA (x y) (LIST x y)

‘(+ 1 2) ‘c);

b) ((LAMBDA (x y) ((LAMBDA (z) (LIST x y z)) ‘c)

‘a ‘b);

c) ((LAMBDA (x y) (LIST x y))

((LAMBDA (z) z) ‘a)

‘b).

2. Напишите функцию в соответствии с вариантом задания

|  |  |
| --- | --- |
| №  варианта | Задание |
| 1 | Функция должна принимать значение T, если два ее аргумента являются списками и они эквивалентны |
| 2 | Функция должна возвращать список с элементами, перечисленными в обратном порядке. |
| 3 | Функция из исходного списка формирует список, первый элемент которого является суммой всех элементов, второй - сумма элементов без первого элемента, третий – сумма элементов без первого и второго элементов и так далее. |
| 4 | Функция должна вычислить сумму произведений элементов, являющихся числами, двух списков |
| 5 | Функция должна добавлять s-выражение на заданное место в списке |
| 6 | Функция, аргументами которой являются два списка, должна возвращать список из элементов, которые встречаются в каждом из двух списков-аргументов. |
| 7 | Функция трех аргументов (список, старый элемент, новый элемент) должна возвращать список, в котором все вхождения старого элемента заменены на новый элемент на всех уровнях списка. |
| 8 | Функция двух аргументов (список, число) должна возвращать список, в котором все элементы старого элемента, являющиеся числами уменьшены на это число на всех уровнях списка |
| 9 | Функция одного аргумента (список) должна возвращать список, элементами которого являются порядковые номера атомов, входящих в список – аргумент. |
| 10 | Функция должна находить сумму всех числовых элементов в списке. Список может содержать подсписки произвольной глубины |
| 11 | Функция должна удалять из списка N-ый элемент |
| 12 | Функция должна определять, входит ли каждый элемент списка в него лишь один раз |
| 13 | Функция должна определять, является ли первый список подмножеством второго |
| 14 | Функция должна вставлять на заданное место заданный элемент |
| 15 | Функция, аргументами которой являются два списка, должна возвращать список, элементы которого являют только элементы входящие в один из списков |
| 16 | Функция, аргументами которой являются два списка, должна возвращать T, если одинаковые атомы расположены в списках в одном и том же порядке |
| 17 | Функция, аргументами которой являются список и число, должна возвращать элемент списка по заданному номеру, считая от конца |
| 18 | Функция, аргументом которой является список, должна возвращать список, состоящий только из числовых элементов исходного списка. Исходный список может содержать подсписки произвольной глубины вложенности. |
| 19 | Функция, аргументом которой является список и s-выражение, должна возвращать число, показывающее сколько раз s-выражение входит в исходный список. Исходный список может содержать подсписки произвольной глубины вложенности. |
| 20 | Функция, аргументом которой является список, должна возвращать список, содержащий только атомы из исходного списка. Исходный список может содержать подсписки произвольной глубины вложенности. |
| 21 | Функция, аргументами которой являются два списка, должна возвращать список, содержащий только атомы из первого списка, повторяющиеся число раз, заданные элементами второго списка. Например, из списков ( a b c) и (1 2 3) строить список (a b b c c c) |
| 22 | Функция, аргументами которой являются два списка, должна возвращать список, содержащий элементы первого списка, не принадлежащие второму списку |
| 23 | Функция, аргументом которой является список, должна возвращать список, содержащий только атомы из исходного списка, стоящие в исходном списке на четных позициях. |

3. Напишите с помощью композиции условных выражений функции от четырех аргументов

**AND4**(x1 x2 x3 x4) и **OR4**(x1 x2 x3 x4), совпадающие с функциями AND и OR от четырех аргументов.

4. С помощью предложений **COND** или **CASE** определите функцию, которая возвращает в качестве значения столицу заданного аргументом государства.

5. Запрограммируйте с помощью предложения **DO** функцию факториал.

6. Запишите с помощью предложения **PROG** функцию - аналог встроенной функции **LENGTH**, которая возвращает в качестве значения длину списка (количество элементов на верхнем уровне).

**Отчет по лабораторной работе**

Отчет по лабораторной работе должен содержать:

- задание на лабораторную работу;

- протоколы работы в LispWorks