Федеральное государственное бюджетное образовательное учреждение высшего

образования

«Чувашский государственный университет им. И.Н. Ульянова»

Кафедра вычислительной техники

Лабораторная работа №11

**Функции высших порядков: применяющие и отображающие функционалы**

Вариант 9

Выполнил: Иванов В.С.

студент группы ИВТ-41-22

Проверил: кандидат технических наук

Обломов Игорь Александрович

Чебоксары, 2024

Цель работы: изучить функционалы языка Лисп

Теоретические сведения.

Функции, используемые до сих пор, имели в качестве аргументов выражения, относящиеся по типу к данным. То же можно сказать и относительно результатов, возвращаемых функциями. Однако в Лиспе принята единообразная скобочная форма записи S-выражений, что позволяет сделать вывод, что смысл любого выражения зависит от его контекста. Аргумент функции, значением которого является функция, называется *функциональным аргументом*, а функция, имеющая функциональный аргумент – *функционалом*. Функция, которая возвращает в качестве результата другую функцию, называется функцией с *функциональным значением*.

Одним из основных классов функционалов являются те, которые применяют функциональный аргумент к параметрам. Такие функционалы называются *применяющими.* Они родственны универсальной функции Лиспа – EVAL, которая вычисляет значение произвольной формы, а не только функции. В Лиспе имеются применяющие функционалы APPLY и FANCALL.

Формат функционала APPLY следующий:

**(apply fn list),**

где fn – функция; list – список аргументов, к которому применяется функция.

Следующий применяющий функционал – FUNCALL по своему действию аналогичен APPLY, но аргументы для вызываемой функции получает не списком, а по отдельности. Его формат следующий:

**(funcall fn ar1 ar2 … arn)**.

Можно сказать, что он применяет функцию к последовательности аргументов, например,

Другой важный класс функционалов, которые определенным образом отображают список (последовательность) в новую последовательность или порождают побочный эффект, связанный с исходной последовательностью. Такие функционалы носят название *отображающие функционалы* или MAP-функционалы. Различают два вида MAP-функционалов, первый из которых применяет функциональный аргумент к CAR-последовательностям списка, а второй – к CDR-последовательностям. Наиболее распространенные из MAP-функционалов – MAPCAR и MAPLIST.

Формат MAPCAR следующий:

(mapcar fn ’(ar1 ar2 … arn)).

Функционал работает следующим образом: вначале применяется функция к CAR-элементу, затем к CADR-элементу и так далее, до конца списка. Результатом функции будет являться список результатов от применения функционального аргумента ко второму параметру MAPCAR

Для работы хвостовыми частями списков используют функционал MAPLIST, его формат следующий:

(maplist fn list),

где – fn – функциональный аргумент;

– list – список.

MAPLIST работает подобно MAPCAR, но только не над элементами списка, а над CDR-последовательностями списка, причем исходно полагается, что исходный список является собственным хвостом.

Отображающие функционалы чаще всего используются для программирования циклов специального вида. Разновидностью MAPCAR и MAPLIST являются функционалы MAPCAN и MAPCON, которые строят новые списки из результатов, используя структуроразрушающую функцию NCONC.

Разновидностью функционалов в Лиспе являются предикаты планирования функций.

(some тест список1 ... списокN) – выполняет действия предиката <тест> над car-обьектами <списка1>, ..., <спискаN>, затем – над cad-обьектами каждого списка и до тех пор, пока тест не вернет значение, отличное от NIL, или не встретится конец списка. Если тест возвращает значение, отличное от NIL, SOME возвращает это значение. Если конец списка достигнут, SOME возвращает NIL.

(notany тест список1 ... списокN) – выполняет действия предиката <тест> над car-элементами <списка1>, ..., <спискаN>, затем - над cadr-элементами каждого списка и до тех пор, пока тест не выдаст значение, отличное от NIL, или не встретится конец списка. Если тест вернул значение, отличное от NIL, NOTANY возвращает NIL. Если встретился конец списка, NOTANY возвращает Т.

(every тест список1 ... списокN) выполняет действия предиката <тест> над car-элементами <списка1>, ..., <спискаN>, затем – над cadr-элементами каждого списка, и т.д. до тех пор, пока тест не выдаст NIL или не встретится конец списка. Если тест выдает NIL, EVERY возвращает NIL. Если встретился конец списка, EVERY возвращает Т.

(notevery тест список1 ... списокN) выполняет действия предиката <тест> над car-элементами <списка1>,...,<спискаN>, затем – над cadr-элементами каждого списка, и т.д. до тех пор, пока тест не вернет NIL или не встретится конец списка. Если тест вернет NIL, NOTEVERY возвращает Т. Если встретился конец списка, NOTEVERY возвращает NIL.

Код программы.

(write (apply '\* '(0.5 2 3)))

(terpri)

(write (apply '\* '(0.5 5 10)))

(terpri)

(write (funcall '\* 0.5 2 3))

(terpri)

(write (funcall '\* 0.5 5 10))

(terpri)

;2

(write (apply (lambda (a b) (\* 0.5 a b)) '(2 3)))

(terpri)

(write (funcall (lambda (a b) (\* 0.5 a b)) 2 3))

;3

(terpri)

(write ((lambda (a b fn)(\* 0.5 (funcall fn a b))) 2 3 '\*))

(terpri)

(write ((lambda (lst) ( \* 0.5 (apply '\* lst))) '(2 3)))

(terpri)

;4

(write (mapcar 'cons '(1 2 3) '(a b c)))

(terpri)

(write (maplist 'cons '(1 2 3) '(a b c)))

(terpri)

(write (mapcon 'reverse '(1 2 3)))

(terpri)

;5

(write (mapcar (lambda (lst1 lst2) (cons lst1 lst2)) '(1 2 3) '(a b c)))

(terpri)

;6

(write ((lambda (lst1 lst2)

    (maplist #'cons lst1 lst2)) '(1 2 3) '(a b c))

)

(terpri)

;7

(write (some 'equal '(a b c) '(a b d)))

Вывод: изучил функционалы языка Лисп.