Задание №13
Макросы

1. Общая постановка задачи

Реализуйте функционал, описанный в задании с номером вашего варианта. При описании можно использовать средства императивного программирования, но функционал должен быть реализован без побочных эффектов. Все циклические процессы должны быть реализованы с помощью хвостовой рекурсии.

Опишите программу в текстовом файле с именем `task13-NN.lsp`, где `NN` — номер вашего варианта. Полученный файл загрузите на портал в качестве выполненного задания.

2. Предварительные замечания

Все циклические процессы должны быть реализованы с помощью хвостовой рекурсии.

Все реализованные макросы не должны иметь «протечек» (стоит обратиться к книге «Practical Common Lisp», глава 8).

Порядок вычислений аргументов макроса — от первого до последнего в порядке обхода в глубину.

Не следует делать предположений на счет задания, не сформулированных явно в условии. Если возникают сомнения — задайте вопрос на форуме «Язык Lisp».

3. Пример выполнения задания

ЗАДАНИЕ: Определите макрос для цикла `while-less`, реализующий итерационный процесс через вызов функции с хвостовой рекурсией. Формат команды для вызова макроса:

```
(while-less expr-1 body)
```

где `expr-1` — выражение, возвращающее числовое значение; `body` — тело цикла, состоящее из одного или нескольких выражений, в котором последнее выражение возвращает числовое значение.

Выражение `expr-1` должно вычисляться точно один раз. Выражение `body` должно вычисляться как минимум один раз. Вычисление `body` повторяется пока полученное значение последнего в нем выражения меньше значения `expr-1`. Результат выполнения макроса — значение последнего выражения в последнем вычислении `body`.

Например, для подсчета суммы $\sum_{i=2}^{N} \sin i$ можно составить код с использованием макроса

```
(let ((i 2)
      (s 0))
  (if (<= i N)
      (while-less (+ N 1)
        (incf s (sin i))
        (incf i)))
  s)
```

Выполните пример задания 3 с использованием разработанного макроса.

РЕШЕНИЕ:

Содержимое файла `task13-NN.lsp`:

```
(defmacro while-less (e1 &body e2)
  ;; генерируем два вспомогательных идентитфикатора. Один сохраняем
  ;; в переменной func-name, второй - в переменной param-name.
  ;; Первый идентификатор будем использовать как имя нашей рекурсивной
  ;; функции для цикла, а второй - как имя формального параметра этой функции
  (let ((func-name (gensym))
        (param-name (gensym)))
    ;; определяем тело самой макроподстановки с указанными параметрами.
```

```
    ;; Заменяем вызов макроса на определение локальной функции
    ;; (с помощью labels) с дальнейшим вызовом этой функции.
    ;; Дальнейшие комментарии касаются ее функционирования.
    ;; Параметр локальной функции - значение выражения e1
    `(labels ((,func-name (,param-name)
                ;; вычисляем последовательно все выражения, входящие в e2
                ;; Для этого поместим все элементы из e2 в тело выражения
                ;; progn. Вычисление такого progn выдаст нам значение
                ;; последнего выражения из e2. Полученное значение
                ;; сохраняем в переменной new-e2
                ;; после этого вычисления нет обращения к параметрам макроса
                ;; напрямую, поэтому использование new-e2 не приведет к
                ;; появлению протечек
                (let ((new-e2 (progn ,@e2)))
                  ;; сравниваем полученное значение с параметром функции
                  (if (< new-e2 ,param-name)
                      ;; если условие цикла выполняется, то вызываем функцию
                      ;; рекурсивно
                      (,func-name ,param-name)
                      ;; иначе выдаем вычисленный результат
                      new-e2))))
       (,func-name ,e1))))  ; начальный вызов функции от выражения,
                            ; переданного в качестве e1. Выражение
                            ; вычислится только один раз - перед передачей
                            ; его значения в качестве фыактического
                            ; параметра

;; Решение для примера задания 3
(defun y (n)
  (let ((i 1)    ; параметр суммы
        j        ; параметр произведения (декларация)
        (sum 0)  ; аккумулятор для суммы
        prod     ; аккумулятор для произведения (декларация)
        ;; функция для выражения, вычисляемого в цикле
        (func-ij (lambda (i j) (+ (/ i j) (* i i 1/2))))
        (max-n (+ 1 n))) ; для того, чтобы не повторять выражение (+ n 1)
    ;; т.к. while-less вычисляет второе выражение минимум один раз, то нужна
    ;; следующая проверка, на случай n = 0, когда нет ни одного слагаемого
    (if (< i max-n) ; if без ветви else
        (while-less max-n
          ;; не делаем проверку (< j max-n), т.к. начальное значение j
          ;; не больше начального i, а i уже сравнивали с max-n
          (setq j 1)
          (setq prod 1)
          (while-less max-n
            (setq prod (* prod (funcall func-ij i j)))
            (incf j))
          (incf sum prod)
          (incf i)))
    sum)) ; возвращаем значение переменной sum как результат let

;; примеры запуска
(print (y 5))
(print (y 7))
(print (y 15))
```

Файлы с примерами можно загрузить с портала.

4. Варианты заданий

1. Определите макрос для цикла с параметром `for`, реализующий итерационный процесс через вызов функции с хвостовой рекурсией. Формат команды для вызова макроса:

```
(for (parameter start-value end-value step) body)
```

где `parameter` — идентификатор — параметр цикла; `start-value` — начальное значение параметра цикла; `end-value` — конечное значение параметра цикла; `step` — шаг значения параметра цикла — опциональный параметр, равный по умолчанию одному; `body` — тело цикла — одно или несколько выражений. Результат выполнения макроса — значение последнего выражения в теле `body`.

Например, для подсчета суммы $\sum_{i=2}^{N} \sin i$ можно составить код с использованием макроса

```
(let ((s 0))
  (for (i 2 N)
    (incf s (sin i)))
  s)
```

Выполните свой вариант задания 3 с использованием разработанного макроса.

2. Определите макрос для цикла с параметром `for`, реализующий итерационный процесс через вызов функции с хвостовой рекурсией. Формат команды для вызова макроса:

```
(for ((parameter-1 start-value-1)
      ...
      (parameter-k start-value-k))
     (continue-expr cont-body)
   body)
```

где `parameter-1` – `parameter-k` — идентификаторы — параметры цикла;
`start-value-1` – `start-value-k` — начальные значения параметров цикла;
`continue-expr` — условие выполнения цикла; `cont-body` — одно или несколько выражений, которые выполняются после каждой итерации (перед проверкой условия `continue-expr`); `body` — тело цикла — одно или несколько выражений. Результат выполнения макроса — значение последнего выражения в теле `body`.

Например, для подсчета суммы $\sum_{i=2}^{N} \sin i$ можно составить код с использованием макроса

```
(for ((i 2)
      (s 0))
     ((<= i N) (incf i))
  (incf s (sin i)))
```

Выполните свой вариант задания 3 с использованием разработанного макроса.

3. Определите макрос для цикла `while`, реализующий итерационный процесс через вызов функции с хвостовой рекурсией. Формат команды для вызова макроса:

```
(while condition body)
```

где `condition` — условие выполнения цикла; `body` — тело цикла — одно или несколько выражений. Тело цикла должно исполняться пока результат вычисления условия не равен `NIL`. Результат выполнения макроса — значение последнего выражения в теле `body`.

Например, для подсчета суммы $\sum_{i=2}^{N} \sin i$ можно составить код с использованием макроса

```
(let ((s 0)
      (i 2))
  (while (<= i N)
    (incf s (sin i))
    (incf i))
  s)
```

Выполните свой вариант задания 3 с использованием разработанного макроса.

4. Определите макрос для цикла `repeat`, реализующий итерационный процесс через вызов функции с хвостовой рекурсией. Формат команды для вызова макроса:

```
(repeat (body) condition)
```

где body — тело цикла — одно или несколько выражений; condition — условие окончания цикла; Тело цикла должно исполняться до тех пор, пока результат вычисления условия не станет отличным от NIL. Результат выполнения макроса — значение последнего выражения в теле body.

Например, для подсчета суммы $\sum_{i=2}^{N} \sin i$ можно составить код с использованием макроса

```
(let ((s 0)
      (i 2))
  (repeat
    ((setf s (+ s (sin i)))
     (incf i))
    (> i N))
  s)
```

Выполните свой вариант задания 3 с использованием разработанного макроса.

5. Определите макрос для цикла do-list, реализующий итерационный процесс через вызов функции с хвостовой рекурсией. Формат команды для вызова макроса:

```
(do-list (parameter list) body)
```

где parameter — идентификатор — параметр цикла; list — список; body — тело цикла — одно или несколько выражений. Параметр цикла пробегает все значения в заданном списке. Тело цикла выполняется для каждого значения параметра цикла. Результат выполнения макроса — значение последнего выражения в теле body.

Например, для подсчета суммы синусов элементов списка L можно составить код с использованием макроса

```
(let ((s 0))
  (do-list (i L)
    (setf s (+ s (sin i)))))
```

Опишите функцию all-in-range, принимающую два целочисленных аргумента M и N и возвращающую список всех целых чисел от M до N.

Выполните свой вариант задания 3 с использованием разработанного макроса и функции all-in-range.

6. Определите макрос для цикла do-range, реализующий итерационный процесс через вызов функции с хвостовой рекурсией. Формат команды для вызова макроса:

```
(do-range (start-value end-value parameter) body)
```

где start-value — начало диапазона; end-value — конец диапазона; parameter — идентификатор — параметр цикла; body — тело цикла — одно или несколько выражений. Тело цикла выполняется для каждого значения параметра в заданном диапазоне с шагом 1. Результат выполнения макроса — значение последнего выражения в теле body.

Например, для подсчета суммы $\sum_{i=2}^{N} \sin i$ можно составить код с использованием макроса

```
(let ((s 0))
  (do-range (2 N i)
    (setf s (+ s (sin i))))
  s)
```

Выполните свой вариант задания 3 с использованием разработанного макроса.

7. Определите макросы для суммирования sum-range и произведения prod-range, реализующие итерационный процесс через вызов функций с хвостовой рекурсией. Формат команд для вызова макроса:

```
(sum-range (parameter start-value end-value) body)
(prod-range (parameter start-value end-value) body)
```

где parameter — идентификатор — параметр суммирования/произведения; start-value — начальное значение параметра; end-value — конечное значение параметра; body — тело — одно или несколько выражений. Результат выполнения макросов — сумма/произведение значений последних выражений в теле body для каждого из значений параметра с шагом 1.

Например, для подсчета суммы $\sum_{i=2}^{N} \sin i$ можно составить код с использованием макроса

```
(sum-range (i 2 N) (sin i))
```

Выполните свой вариант задания 3 с использованием разработанных макросов.

8. Определите макрос для цикла `while`, реализующий итерационный процесс через вызов функции с хвостовой рекурсией. Формат команды для вызова макроса:

```
(while (condition result) body)
```

где `condition` — условие выполнения цикла; `body` — тело цикла — одно или несколько выражений; `result` — выражение, значение которого выдается по окончании цикла. Тело цикла должно исполняться пока результат вычисления условия не равен `NIL`. Результат выполнения макроса — значение выражения `result` (после выполнения `body` достаточное количество раз).

Например, для подсчета суммы $\sum_{i=2}^{N} \sin i$ можно составить код с использованием макроса

```
(let ((s 0)
      (i 2))
  (while ((<= i N) s)
    (setf s (+ s (sin i)))
    (incf i)))
```

Выполните свой вариант задания 4 с использованием разработанного макроса.

9. Определите макрос для цикла `repeat`, реализующий итерационный процесс через вызов функции с хвостовой рекурсией. Формат команды для вызова макроса:

```
(repeat (body) (condition result))
```

где `body` — тело цикла — одно или несколько выражений; `condition` — условие окончания цикла; `result` — выражение, значение которого выдается по окончании цикла. Тело цикла должно исполняться до тех пор, пока результат вычисления условия не станет отличным от `NIL`. Результат выполнения макроса — значение выражения `result` (после выполнения `body` достаточное количество раз).

Например, для подсчета суммы $\sum_{i=2}^{N} \sin i$ можно составить код с использованием макроса

```
(let ((s 0)
      (i 2))
  (repeat
    ((setf s (+ s (sin i)))
     (incf i))
    ((> i N) s)))
```

Выполните свой вариант задания 4 с использованием разработанного макроса.