МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)
Кафедра информационных систем и технологий

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«_____» _____ 2017 г.

Условные конструкции и циклы. COND, LOOP

Методические указания к практической работе №3 по дисциплине «Рекурсивно-логическое и функциональное программирование» для студентов, обучающихся по направлению 09.03.02 «Информационные системы и технологии»

Курск 2017

УДК 004
Составитель И.В. Зотов

Рецензент
Кандидат технических наук Ю.А. Халин

Условные конструкции и циклы. COND, LOOP: методические указания к практической работе №3 / Юго-Зап. гос. ун-т; сост. И.В. Зотов. Курск, 2017. 12 с. Библиогр.: с. 12.

Приводится описание базовых функций для поддержки императивного стиля программирования в языке Лисп – CONS, LOOP. Даются практические примеры использования этих функций с иллюстрациями и варианты заданий.

Методические указания предназначены для студентов, обучающихся по направлению 09.03.02 «Информационные системы и технологии». Могут использоваться также студентами, обучающимися по направлениям, связанным с вычислительной техникой и интеллектуальными информационными системами.

Текст печатается в авторской редакции.

Подписано в печать 10.11.2017 . Формат 60x84 1/16.
Усл.печ. л. 0,5. Уч.-изд. л. 0,4. Тираж 100 экз. Заказ 1753. Бесплатно.
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

СОДЕРЖАНИЕ

1. Цель работы

Целью данной работы является приобретение умений в составлении и отладке Lisp-программ с разветвлениями и циклами.

2. Краткие теоретические сведения

Программа на языке Лисп представляет собой набор определений функций в любом порядке. Программа запускается вызовом одной из определенных функций. Вызов должен содержать фактические параметры. При вызове функций значения формальных параметров запоминаются в стеке. Им передается значение фактических параметров (значение формальных параметров восстанавливается после выполнения вызываемой функции). Далее происходит вызов функции с фактическими аргументами. Возникает последовательность вызовов, заканчивающаяся вызовом базовой функции. Вычисленное значение функции возвращается формальному параметру, на месте которого располагается вызов функции.

В языке Лисп функции могут вызываться различными способами, в том числе и последовательно, как в императивных языках программирования, например, в Паскале или С. То есть Лисп является не строго функциональным языком и позволяет программировать порядок вычислений. При этом ход вычислений может зависеть от определенных условий, вызовы функций могут повторяться заданное число раз.

Для программирования последовательности вычислений в Лиспе имеются функции PROG1, PROG2, PROGN. Конструкция PROGi может задаваться как явно, так и неявно. Формат явного PROGi имеет следующий вид: (PROGN F1 F2 … FN), где Fi – некоторая форма. PROGN заставляет последовательно сработать всем формам и значение последней формы FN считает своим результатом. Неявный PROGN действует при вызове определения функции, внутри циклов и разветвлений. PROG1 и PROG2 действуют аналогично, но своим значением считают результаты вычисления первой (F1) и второй (F2) формы соответственно. У функций PROG1, PROG2, PROGN переменное число аргументов.

В среде HomeLisp имеется также конструкция PROG. Она аналогична PROGN, но имеет ряд отличий. Во-первых, в ней можно создавать локальные переменные. Во-вторых, в ней может ис-

пользоваться функция безусловного перехода на метку. В-третьих, завершить выполнение этой конструкции можно в любом месте функцией RETURN.

Примеры (здесь функция LET служит для управления лексическим контекстом, т.е. создания локальных переменных):

```
(let ((a 0) (b 0) (c 0))
(prog1 (setq a 111) (setq b 222) (setq c 333)))

==> 111

(let ((a 0) (b 0) (c 0))
(prog2 (setq a 111) (setq b 222) (setq c 333)))

==> 222

(let ((a 0) (b 0) (c 0))
(progn (setq a 111) (setq b 222) (setq c 333)))

==> 333
```

Для организации разветвлений в ходе вычислительного процесса в Лиспе предусмотрена конструкция COND. Она может быть задана неявно или явно.

Неявный COND включает запись некоторого предиката P и тело F, которое содержит последовательность форм F1 F2 …. Если значение предиката P – истина, то вычисляется тело неявного COND по правилу PROGN и это значение является результатом неявного COND. Если значение предиката NIL, то вычисляется значение функции R, записанной следом за неявным COND, а значение самого неявного COND полагается равным NIL. Отметим, что если значение предиката – T, то после неявного COND вычисление функции R не производится. Если тело пустое, то возвращается значение предиката.

Для записи явного COND используется одноименная функция Лиспа. Формат вызова этой функции:

```
(COND
     ((Условие1) Результат1)
     ((Условие2) Результат2)
     ...
     ((Условиеn) Результатn)
     (T           Результат)
)
```

Вычисление функции COND происходит следующим образом. Сперва вычисляется Условие1. Если получается результат, отлич-

ный от NIL, то в качестве результата выдается значение выражения Результат1. Если Условие1 ложно (равно NIL), то вычисляется Условие2. Если снова получается результат, отличный от NIL, то в качестве результата выдается значение выражения Результат2 и т.д. Если в конструкции COND присутствует часть (T Результат) и ни одно из условий не оказалось истинным, то в качестве результата выдается значение выражения Результат. В случае отсутствия части (T Результат) при ложности всех условий, результатом COND будет NIL. Функция COND имеет аналоги в императивных языках программирования, таких как Бейсик, Паскаль, С.

Примеры:

```
(COND
      ((EQ x y) "Равно")
      (T        "Не равно")
)

==> "Не равно"

(COND
      ((EQ (+ x z) y) "Равно")
      (T              "Не равно")
)

==> "Равно"
```

Еще одной функцией организации разветвлений в HomeLisp является функция IF. Функция IF принимает два обязательных параметра и один необязательный. Все параметры должны быть формами. Работа функции заключается в том, что сначала вычисляется значение первого параметра. Если результат отличен от NIL, то вычисляется второй параметр и функция возвращает результат вычисления. Если же значение первого параметра есть NIL, то вычисляется значение третьего параметра (при его наличии) и результат возвращается в качестве результата. Если же третий параметр не задан, а значение первого параметра есть NIL, то функция IF вернет NIL.

Пример:

```
(let ((x 111))
(if (= 0 (% x 2)) 'Четное 'Нечетное))

==> Нечетное
```

Отметим, что здесь символом % обозначена функция взятия остатка от деления.

Для программирования повторения некоторых действий в языке Лисп имеются несколько функций организации циклов. Основная из этих функций – LOOP. Функция LOOP организует бесконечный цикл (тело которого может составлять из произвольного числа форм). Цикл прерывается вызовом функции RETURN. Контроль числа повторений и выход из цикла обеспечиваются в теле цикла дополнительными функциями. Их добавление – обязанность программиста.

Пример:

```
(SETQ FLAG 0)
(LOOP
      ( (NULL LST) FLAG )
      ( (EQ (CAR LST) 2) (SETQ FLAG 1) )
      (  SETQ LST (CDR LST) )
)
```

Еще одна функция организации циклов в HomeLisp – функция DO. Ее формат следующий:

```
(DO (
          (переменная1 значение1 шаг1)
          (переменная2 значение2 шаг2)
          …
          (переменнаяn значениеn шагn)
    )
         (условие_окончания форма_ок1 форма_ок2 … форма_окm)
         (форма1) (форма2) … (формаk)
)
```

Функция выполняется следующим образом:

1. В текущий лексический контекст добавляются переменные цикла (переменная1, переменная2 ... переменнаяn). Им присваиваются соответствующие значения.

2. Вычисляется условие_окончания. Если оно истинно (т.е. не есть NIL), то вычисляются формы окончания – форма_ок1 , форма_ок2 … форма_окm Результат вычисления последней формы возвращается как результат DO.

3. Если условие окончания ложно, вычисляется тело цикла – формы: (форма1) (форма2) … (формаk).

4. Всем переменным (переменная1 , переменная2 … переменнаяn) присваиваются новые значения путем вычисления соответствующих форм (шаг1 , шаг2 … шагn) и происходит переход к п.2.

Пример:

```
(DO (
      (i 1 (+ i 1))
      (j 1 (+ j 1)))
      ((> j 5) (printline 'ok) (printline 'ok) (printline 'ok))
      (print 'i=) (print i) (prints " ") (print 'j=) (printline j)
)

i=1 j=1
i=2 j=2
i=3 j=3
i=4 j=4
i=5 j=5
ok
ok
ok

==> ok
```

Кроме двух названных функций, в HomeLisp поддерживается функция работы с циклами DOTIMES. Эта функция обеспечивает выполнение блока кода заданное число раз. Синтаксис вызова функции следующий:

```
(dotimes (переменная_цикла
          число_повторений
          результат)
         (форма1)
         (форма2)
         …
         (формаn)
)
```

Вычисление происходит следующим образом. Переменная цикла последовательно получает значения от 0 до значения число_повторений-1. При каждом значении переменной цикла последовательно вычисляются формы ($форма_1$) - ($форма_n$). По завершении цикла вычисляется форма-результат и возвращается в качестве результата DOTIMES.

Пример:

```
(dotimes
   (i 5 t)
     (setq i -1)
     (print 'i=)
     (printline i)
)

i=-1
i=-1
i=-1
i=-1
i=-1
```

```
==> T
```

3. Формулировка задания

Написать и отладить программу на Лиспе, реализующую некоторое вычисление с циклами и ветвлениями согласно варианту задания (см. п.4). Работоспособность программы подтвердить снимками экрана. Для организации ветвлений использовать функцию COND (в явной или неявной форме), циклы реализовать с использованием функций LOOP, DO, DOTIMES. При необходимости применять функцию RETURN.

4. Варианты заданий

a. Вывести все числа от 1 до N, которые делятся на 3 и на 5.
b. Вывести все числа от 1 до N, которые делятся на 5 и на 7.
c. Вывести все числа от 1 до N, которые делятся на 7 и на 13.
d. Вывести все числа от 1 до N, которые делятся на 3, но не делятся на 5.
e. Вывести все числа от 1 до N, которые делятся на 3 и на 7, но не делятся на 5.
f. Вывести все числа от 1 до N, которые делятся на 3 или на 7, но не делятся на 5.
g. Вывести все числа от 1 до N, которые не делятся на 3 и на 7, но делятся на 5.

Примечание: значение N задается преподавателем.

5. Пример решения задачи

Вывести все числа от 1 до N, которые делятся на 5 и на 13.

Текст программы:

```
(LET ((X 1) (N 1000) (COUNT 0))
(LOOP
      (IF (GREATERP X N) (RETURN COUNT))
      (IF (AND (ZEROP (% X 5)) (ZEROP (% X 13)))
            (PROGN
            (SETQ COUNT (ADD1 COUNT))
            (PRINTLINE X)
```

```
            )
        )
        (SETQ X (+ X 1))
)
)
```

В приведенной программе использовался предикат GREATERP, соответствующий операции >. Функция AND применялась для построения сложного условия, объединяющего элементарные условия логическим И. Переменная COUNT была введена для подсчета количества чисел, удовлетворяющих условию задачи.

Результат работы программы в HomeLisp дан ниже.

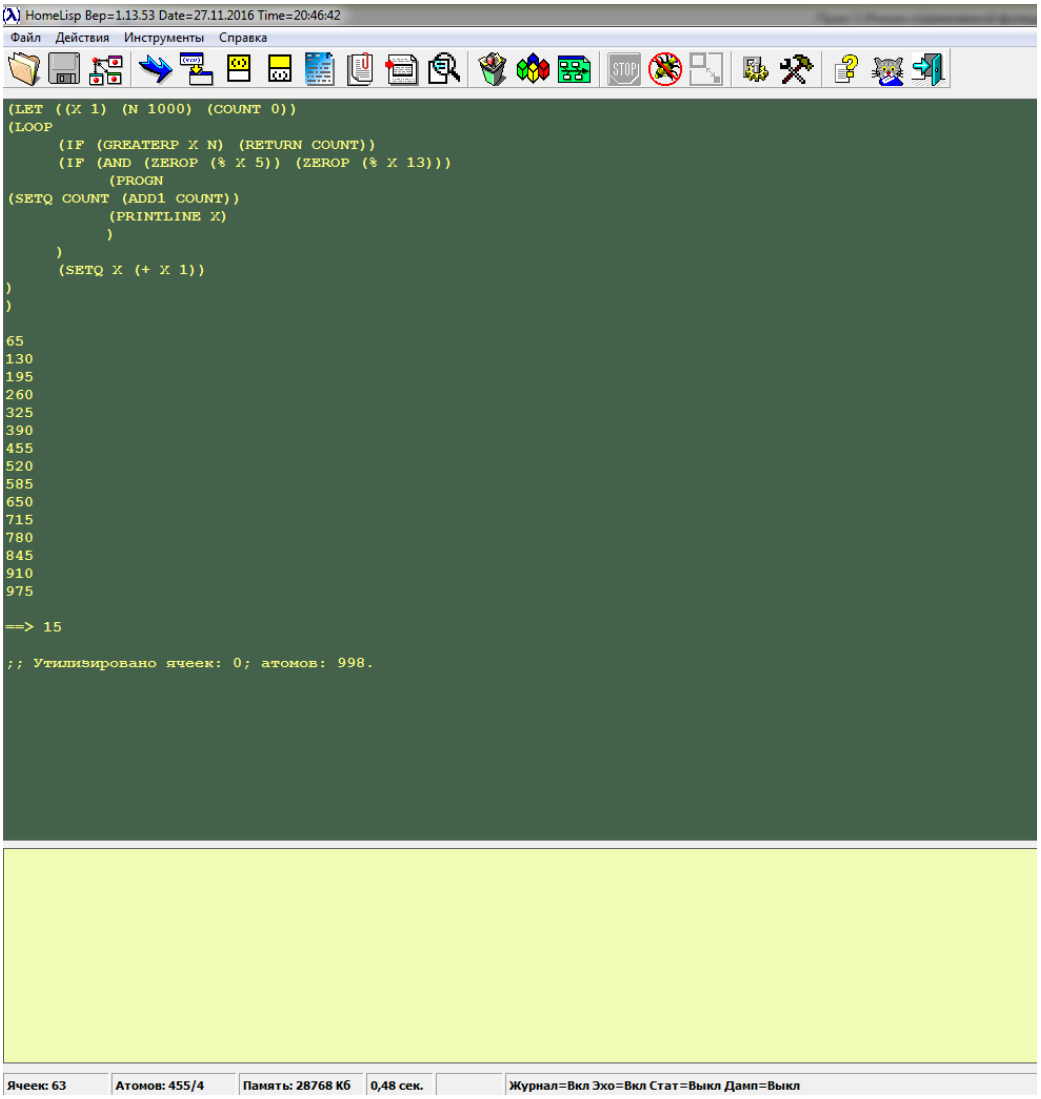

Рисунок 1. Пример выполнения задания

6. Контрольные вопросы

1. Чем отличаются функции PROG1, PROG2 и PROGN?

2. В чем отличие функции PROG от PROGN?
3. Сколько форм можно записывать внутри предложения PROGN?
4. Что такое неявный PROG?
5. Какие функции в Лиспе позволяют организовать разветвление в программе?
6. Сколько альтернативных ветвей можно реализовать с помощью функции COND?
7. Опишите, какие значения возвращает функция IF, если у нее нет третьего аргумента.
8. Что такое циклический процесс?
9. Какие функции в HomeLisp позволяют реализовать циклический процесс?
10. Где в теле функции LOOP должно размещаться условие прерывания цикла?
11. Сколько условий прерывания цикла можно записать в теле функции LOOP?
12. Для чего нужна функция RETURN?

7. Содержание отчета

Результаты выполнения практической работы оформляются в виде отчета, который должен содержать следующие разделы:

- тема работы;
- цель работы;
- индивидуальный вариант задания;
- текст программы;
- снимки экрана, демонстрирующие выполнение задания;
- выводы.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Сергиевский, Георгий Максимович. Функциональное и логическое программирование [Текст] : учебное пособие / Г. М. Сергиевский, Н. Г. Волченков. - М. : Академия, 2010. - 320 с.

2. Шалимов П. Ю. Функциональное программирование [Текст] : учебное пособие. - Издательство БГТУ, 2003. - 160 с.

3. Городняя Л. В. Основы функционального программирования [Электронный ресурс] : учебное пособие. - М. : Интернет-Университет Информационных Технологий, 2004. – 217 с. - Режим доступа: http://biblioclub.ru/index.php?page=book_red&id=233773