Концепция виртуальной Лисп-машины.

Лекция 7.

Специальности : 230105, 010501

Структуры данных в концепции СФЯ.

Строго Функциональный Язык (СФЯ) не предполагает наличия специальных функций изменения структур. В СФЯ для ранее созданных структур допускается :

- **Анализ;**
- **Расчленение;**
- **Копирование.**

Созданные структуры никогда не изменяются. Структуры, значения которых уже не нужны, не уничтожаются.

Лисп-ячейки.

Участок оперативной памяти, в котором работает Лисп-система, разбивается в представлении последней на *списочные ячейки* (Лисп-ячейки). Списочная ячейка состоит из двух частей, полей CAR и CDR (Рис. 1).

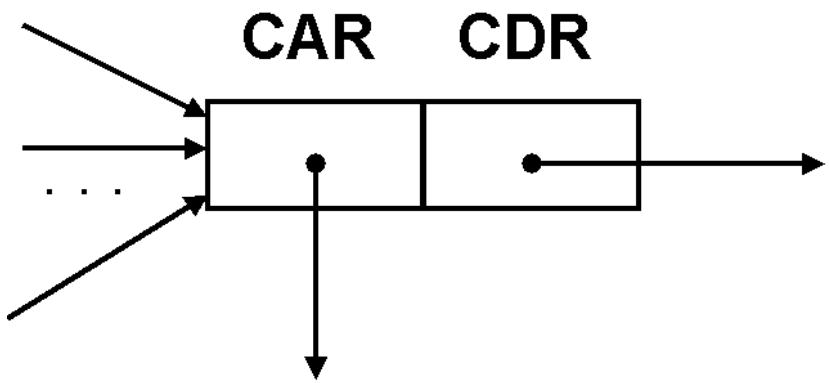

Рис. 1

Указатель может ссылаться на другую списочную ячейку или на некоторый другой списочный объект, например, атом. На каждую списочную ячейку может ссылаться произвольное количество указателей.

Структура одноуровневого списка.

Указателем списка является указатель на первую ячейку списка. На ячейку могут указывать :

- **Поля CAR и CDR других ячеек;**
- **Указатели на значения у символов.**

Графически одноуровневый список представляется последовательностью ячеек, связанных друг с другом через указатели в правой части ячеек (рис. 2).

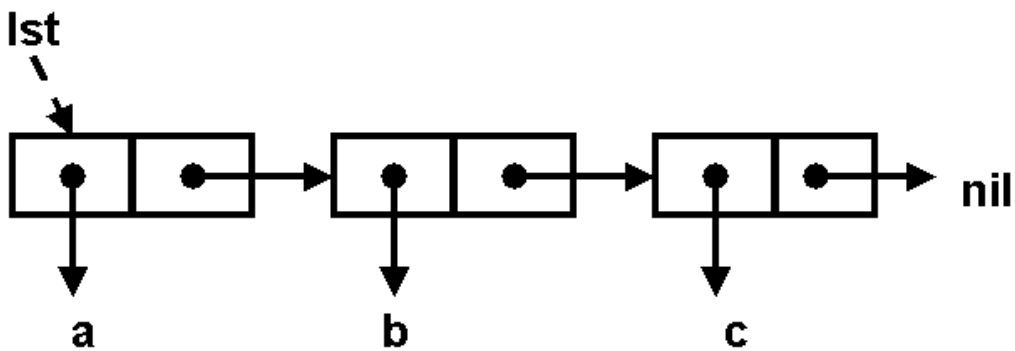

Рис. 2

Указатели и присваивание.

Системные свойства символа включают указатель на значение. Побочным эффектом функции присваивания SETQ является замещение указателя в поле значения символа. Например, вызов (setq lst '(a b c)) создает в качестве побочного эффекта изображенную на рис.2 штриховую стрелку.

Правое поле последней ячейки списка на рис.2 в качестве признака конца списка ссылается на другой список, т.е. атом nil. Графически ссылку на пустой список часто изображают в виде перечеркнутого поля. Указатели из полей CAR ячеек списка ссылаются на структуры, являющиеся элементами списка. Если список содержит подсписки, то на месте атомов будут находиться первые ячейки подсписков (рис.3).

Структура многоуровневого списка.

(setq lst1 '((e f) a b c))

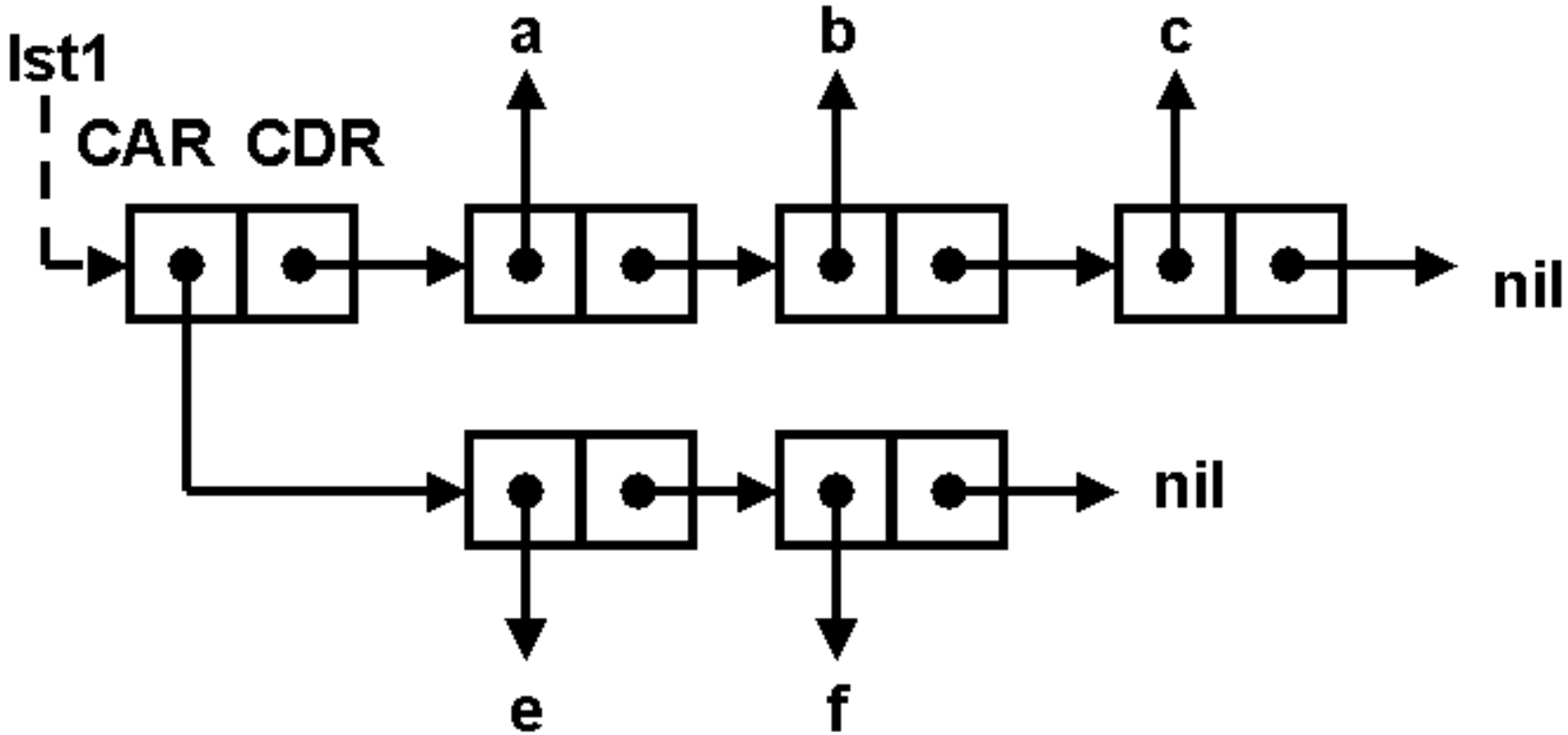

Рис. 3

Логическая и физическая структура списка.

Логически идентичные атомы содержатся в памяти виртуальной Лисп-машины один раз, однако логически идентичные списки могут быть представлены различными Лисп-ячейками. Рассмотрим две последовательности вызовов.

(setq lst2 '((b c) a b c)
Результат :

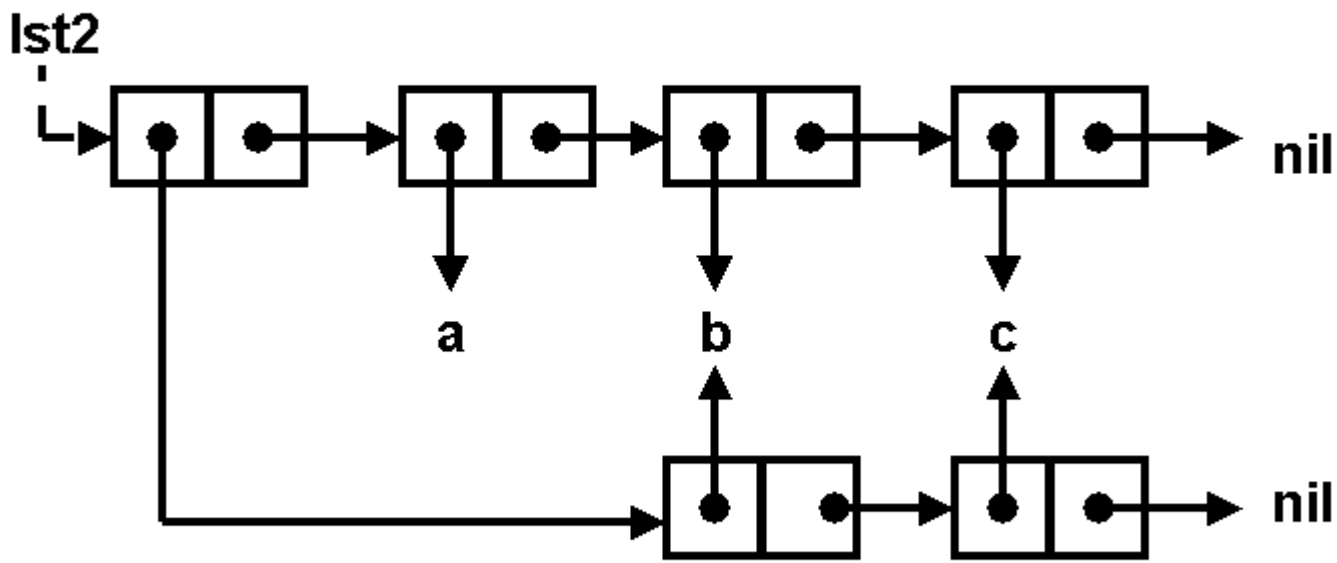

Рис. 4

Логическая и физическая структура списка.

(setq bc ‘(b c))
(setq abc (cons ‘a bc))
(setq lst3 (cons bc abc))
Результат :

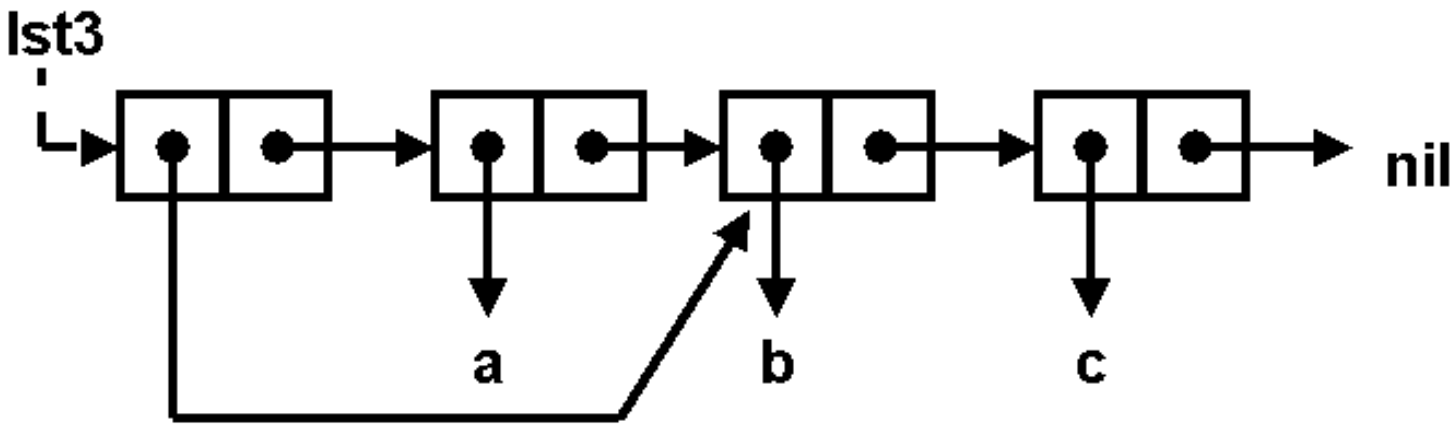

Рис. 5

Логическая структура списка представляется в скобочной нотации и всегда имеет форму дерева, в то время как физическая структура, изображаемая графически совокупностью Лисп-ячеек, в общем случае есть ациклический граф.

CONS и память.

CONS создает новую списочную ячейку. Содержимым левого поля новой ячейки становится содержание первого аргумента вызова, а правого – значение второго аргумента (рис. 6). Применение функции CONS не изменяет значения головы и хвоста.

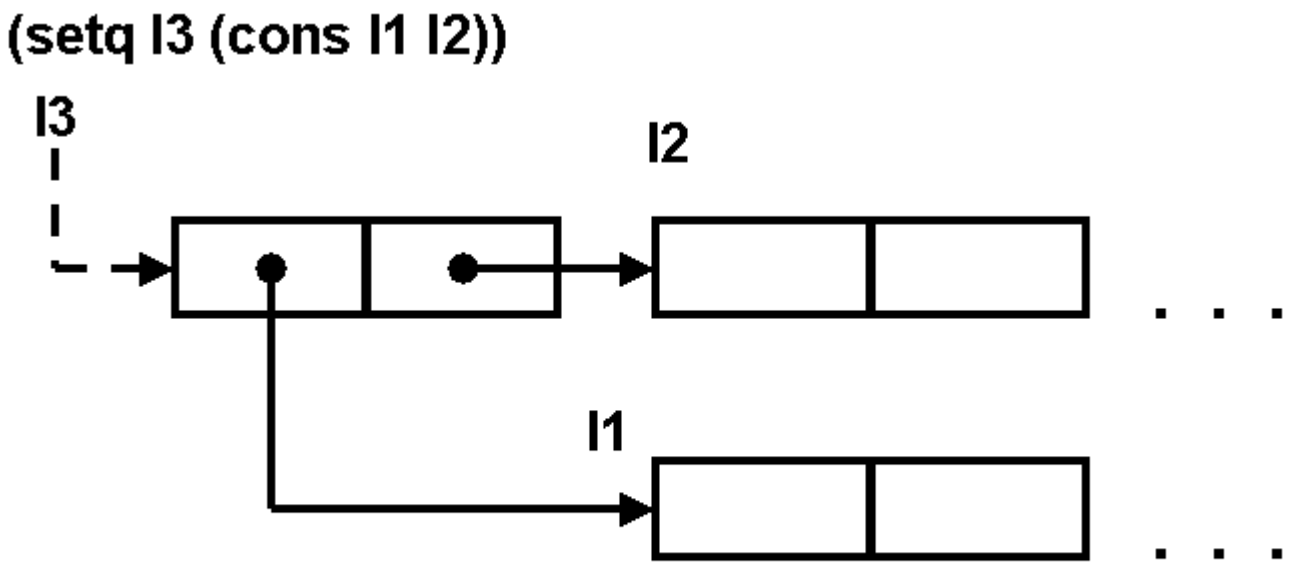

Рис. 6

Каждый раз при вызове конструктор CONS организует новую списочную ячейку, что ведет к дополнительному использованию памяти.

Управление памятью и сборка мусора.

Мусором называются Лисп-ячейки, недоступные через символы и указатели. Мусор образуется в двух случаях :

•Вычисленная структура не сохраняется с помощью SETQ;

•Теряется ссылка на старое значение в результате побочного эффекта нового вызова SETQ или другой функции.

Пример1.

Значение вызова (list ‘a ‘b) лишь печатается, после чего соответствующая ему структура ‘(a b) остается в памяти мусором.

Пример 2.

В результате последовательности вызовов

(setq lst3 ‘((d e) a b c))

(setq lst3 (cdr lst3)) структура (d e) становится мусором.

Мусорщиком в Лисп-системах называется средство автоматического сбора ячеек, ставших мусором, в список свободной памяти. Мусорщик может работать либо непрерывно на фоне вычислений, либо автоматически запускаться при недостатке места в рабочей памяти.

Точечная нотация.

В muLISP'е точечная пара образуется как результат вызова CONS со вторым аргументом-атомом. Примеры :

Вызов (cons 'a 'b) дает точечную пару : (a.b)

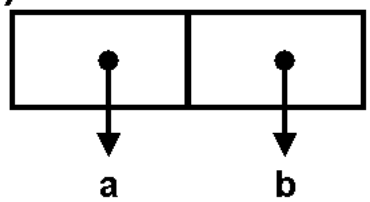

Рис. 7

Выполнение последовательности действий :
(cons 'c (cons 'a 'b))
ведет к образованию точечной пары :
(c.(a.b))

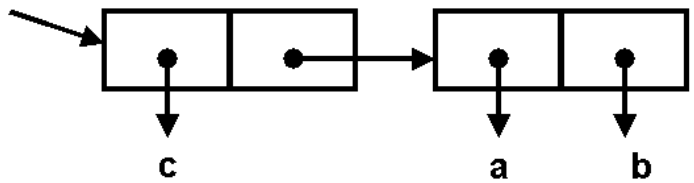

Рис.8

Любой список можно записать в точечной нотации.

Пример : '(a b c) ⟷ **'(a.(b.(c.nil)))**

Точечные пары применяются в теоретических исследованиях, в частности, по системному программированию для обозначения тех частей структуры, которые мы не знаем : (let letlist body). Точечные пары могут использоваться совместно с некоторыми типами данных и с ассоциативными списками.

Структуроразрушающие функции.

<u>Определение.</u> Структуроразрушающими называются функции, при помощи которых можно вносить изменения во внутреннюю структуру уже существующих выражений.

К основным структуроразрушающим функциям muLISP'а относят RPLACA (replace CAR) и RPLACD (replace CDR), которые уничтожают прежние и записывают новые значения в поля CAR и CDR списочной ячейки :

(rplaca <указатель на список> <новое значение головы>)

(rplacd <указатель на список> <новое значение хвоста>)

Примеры.

(setq l1 '(a b c)) Результат : l1→'(a b c)

(rplaca l1 'f) Результат : l1→'(f b c)

(rplacd l1 '(d)) Результат : l1→'(f d)

В newLISP-tk имеется универсальная структуроразрушающая функция (replace exp-key list exp-repl), которая замещает в списке list все элементы, удовлетворяющие выражению exp-key, значением выражения exp-repl.

Пример :

(setq l1 '(a b c)) Результат : l1→'(a b c)

(replace (first l1) l1 'f) Результат : l1→'(f b c)

Пример (muLISP) : преобразование линейного списка в кольцевой.

Задача. Дан линейный список lst. Получить кольцевой список (рис. 9).

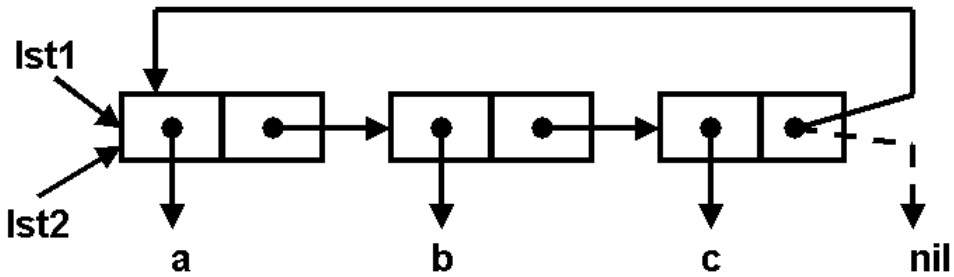

Рис. 9

```
; Преобразование линейного списка в кольцевой

(defun ring (lst)
 (d_r lst lst))

(defun d_r (lst1 lst2)
 ((null lst1) lst2)
 ((null (cdr lst1))(rplacd lst1 lst2))
 (d_r (cdr lst1) lst2))
```

Вызов (ring ‘(a b c)) дает в качестве результата :
‘(a b c a b c . . .)

Объединение списков.

Рассмотрим выполнение функции append.

(setq lst1 ‘(a b c))

(setq lst2 ‘(x y z))

(append lst1 lst2)

При вызове append создает копию своего первого аргумента (рис. 10).

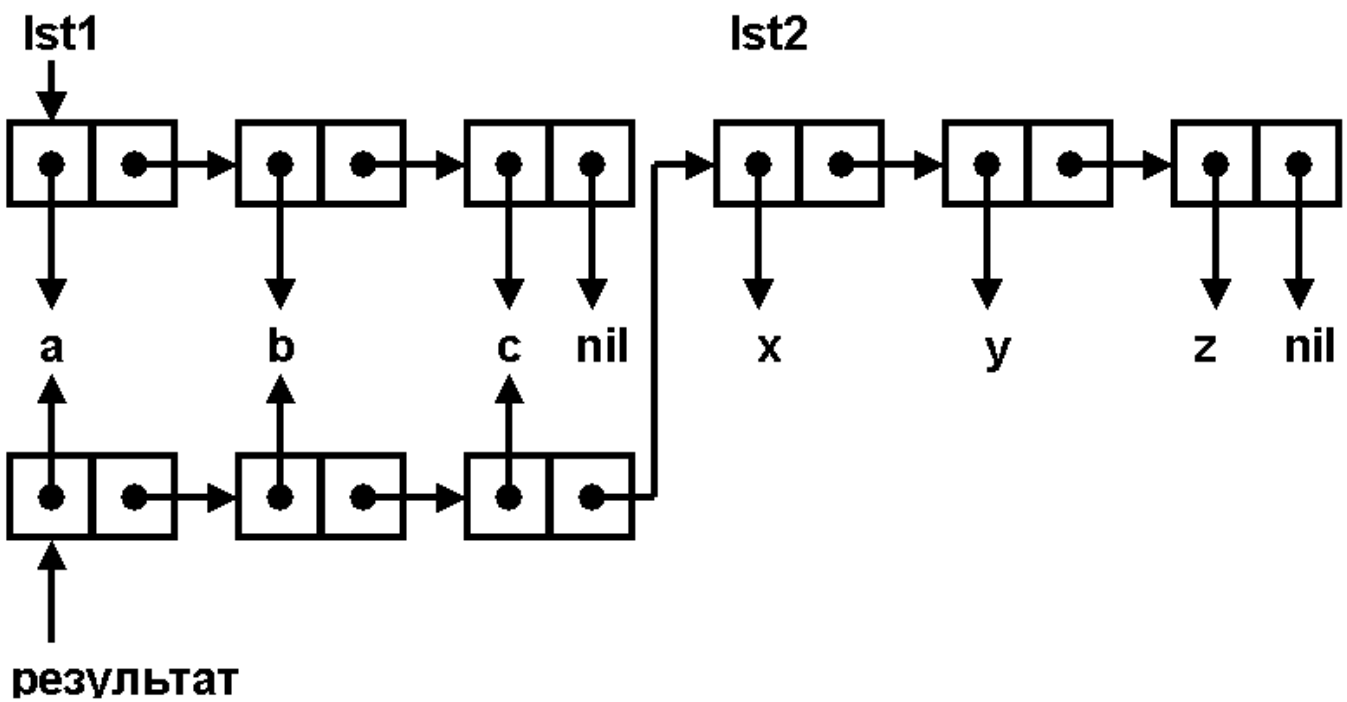

Рис. 10

Использование NCONC в muLISP’e.

В ряде случаев, когда изменение первого аргумента append не является существенным, в целях более рационального использования памяти и сокращения времени работы программы вместо append используют структуроразрушающую функцию NCONC. Функция NCONC аналогична APPEND, но не создает копию первого аргумента, а просто изменяет указатель в поле CDR последней ячейки первого аргумента-списка на начало второго аргумента-списка так, как это показано на рис. 11.

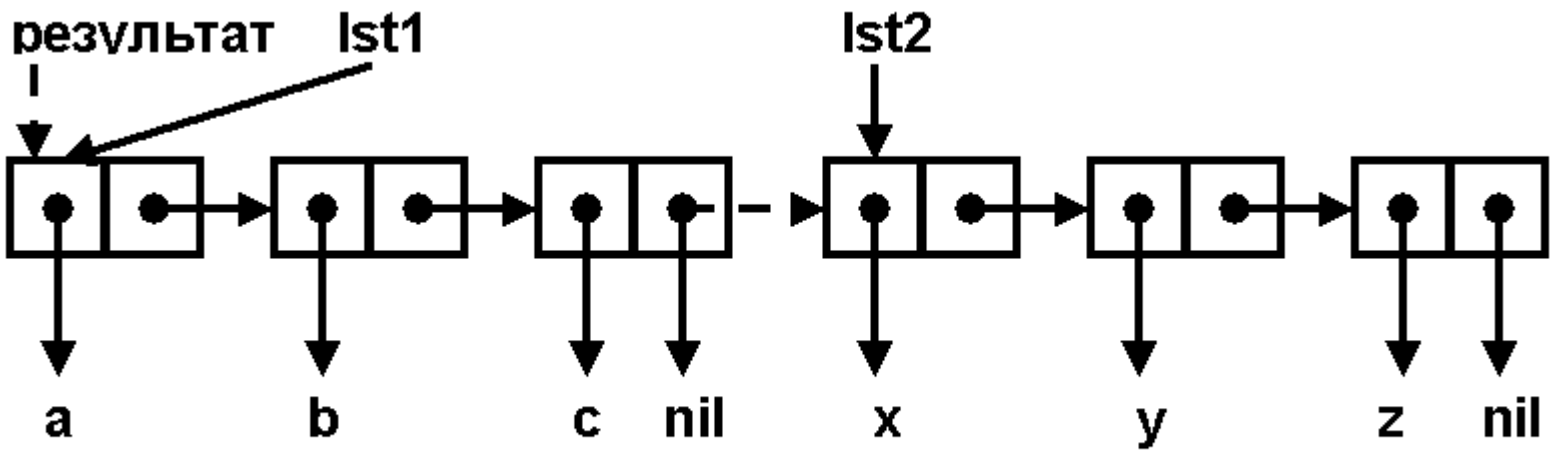

Рис. 11