Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчет по лабораторной работе №1 по курсу "Функциональное и логическое программирование"

Тема Списки в Lisp. Использование стандартных функций

Студент Гурова Н.А.

Группа ИУ7-64Б

Оценка (баллы) _____

Преподаватель Толпинская Н.Б., Строганов Ю.В.

Москва — 2023 г.

1 Теоретические вопросы

1.1 Элементы языка: определение, синтаксис, представление в памяти

В программировании на языке Лисп используются символы и построенные из них символьные структуры.

Символ — это имя, состоящее из букв, цифр и специальных знаков, которое обозначает какой-нибудь предмет, объект, вещь, действие из реального мира.

Примеры символов:

```
1  x
2  defun
3  STep-1984
```

Символы **T** и **NIL** имеют в Лиспе специальное назначение: T обозначает логическое значение истина (true), а NIL — логическое значение ложь (false). Символом NIL также обозначается пустой список.

Наряду с символами в Лиспе используются и **числа**, которые как и символы, записываются при помощи ограниченной пробелами последовательности знаков.

Примеры чисел:

```
1  746
2  -3.14
3  3.055E8
```

Символы и числа представляют собой те простейшие объекты Лиспа, из которых строятся остальные структуры. Поэтому их называют атомарными объектами или просто **атомами**.

Атомы = символы + T + NILL + самоопределимые атомы

Самоопределимые атомы — натуральные числа, дробные числа, вещественные числа, строки — последовательность символов, заключенных в двойные апострофы (например "abc")

Более сложные данные — **списки** и **точечные пары**, которые строятся из унифицированных структур — блоков памяти — бинарных узлов.

```
1    Точечная пара ::= (<атом> . <атом>) |
2                      (<атом> . <точечная пара>) |
3                      (<точечная пара> . <атом>) |
4                      (<точечная пара> . <точечная пара>)
```

Примеры точечных пар:

```
1    (A . B)
2    (A . (B . (C . NIL)))
```

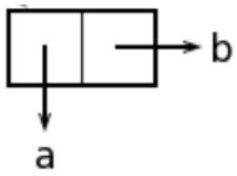

Рисунок 1.1 – Представление в памяти точечной пары (A . B)

```
1    Список ::= <пустой список> | <непустой список>, где
2
3    <пусой сисок> ::= ( ) | Nil,
4    <непустой список>::= (<первый элемент> . <хвост>),
5    <первый элемент> ::= <S–выражение>,
6    <хвост> ::= <список>
```

Примеры списков:

```
1    (A B C)
2    (A (B (C NIL)))
```

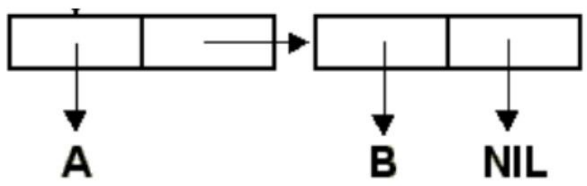

Рисунок 1.2 – Представление в памяти списка (A B)

Атомы и точечные выражения называются **символьными выражениями** или **S-выражениями**.

```
1    S–выражение ::= <атом> | <точечная пара>
```

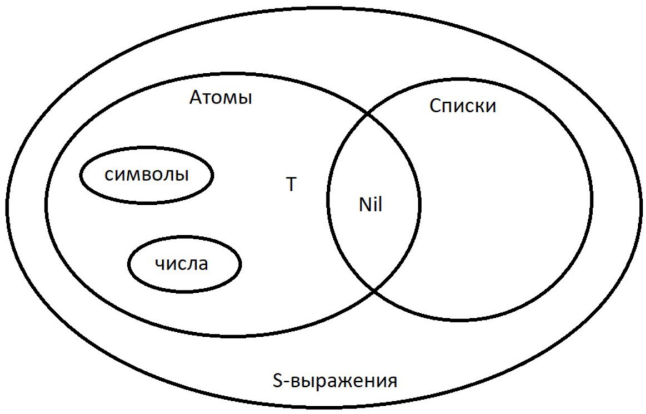

Рисунок 1.3 – Символьные выражения

Синтаксис

Лисп является регистронезависимым языком. Универсальным разделителем является пробел. Наличие скобок является признаком структуры — списка или точечной пары. Любая структура заключается в круглые скобки:

```
1    (A . B)          ; точечная пара
2    (A)              ; список из одного элемента
3    () или Nil       ; пустой список
4    (A B C D)        ; одноуровневый список
5    (A (B C))        ; структурированный список
```

1.2 Особенности языка Lisp. Структура программы. Символ апостроф.

Отличительные особенности языка Лисп: все можно представить в виде функций; только символьная обработка.

В зависимости от контекста одни и те же значения могут играть роль переменных или констант.

Символ **апостроф** — синоним **quote**. quote блокирует вычисление своего аргумента. В качестве значения выдает сам аргумент не вычисляя его. Интерпретатор Лиспа, считывая начинающееся с апострофа выражение, автоматически преобразует его в соответствующий вызов функции quote.

Примеры:

```
'(+ 2 4)              ; (+ 2 4)
'(a b '(c d))         ; (a b (c d))
(quote quote)         ; quote
'quote                ; quote
```

1.3 Базис языка Lisp. Ядро языка.

Базис языка — минимальный набор инструментов и структур данных, который позволяет решать любые задачи.

```
Базис Lisp = атомы + структуры +
                базовые функции + базовые функционалы
```

Функция — правило, по которому каждому значению одного или нескольких аргументов ставится в соответствие конкретное значение результата.

Примеры функций:

```
CAR          ; возвращает головную часть списка
CDR          ; возвращает хвостовую часть списка
CONS         ; включить новый элемент в начало списка
ATOM         ; проверить, является ли аргумент атомом
EQ           ; проверить тождественность двух символов
```

Функционал (функция высшего порядка) — функция, аргументом или результатом которой является другая функция.

Примеры функционалов:

```
APPLY        ; применить функцию к списку аргументов
FUNCALL      ; вызвать функцию с аргументами
```

Ядро — основные действия, которые наиболее часто используются. Такие функции системы обычно реализовываны в виде машинных подпрограмм.

2 Практические задания

2.1 Представить следующие списки в виде списочные ячеек

1. '(open close halph)

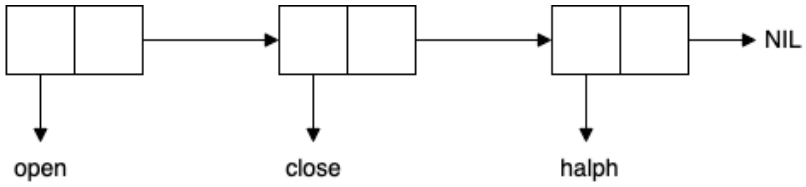

Рисунок 2.1

2. '((open1) (close2) (halph3)))

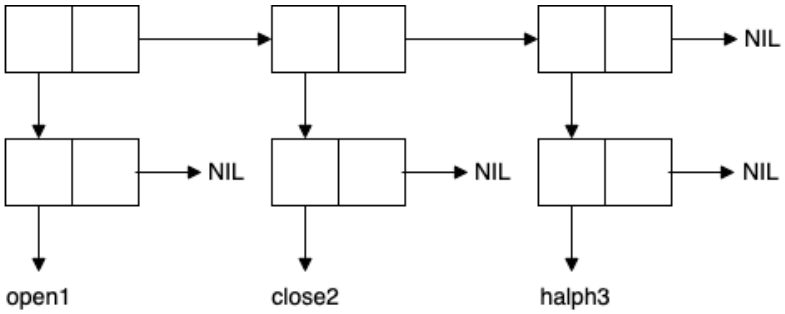

Рисунок 2.2

3. '((one) for all (and (me (for you))))

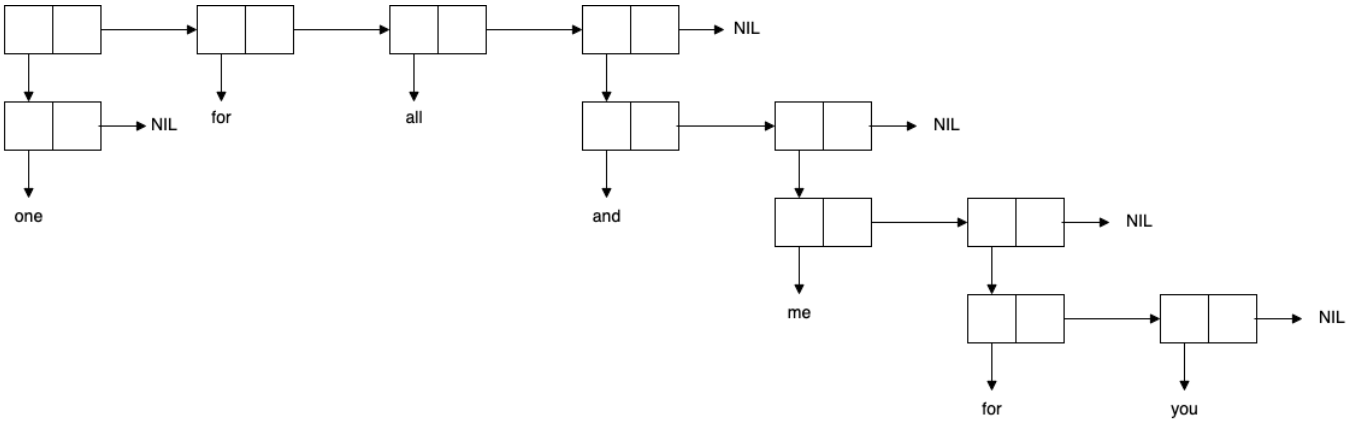

Рисунок 2.3

4. ’((TOOL) (call))

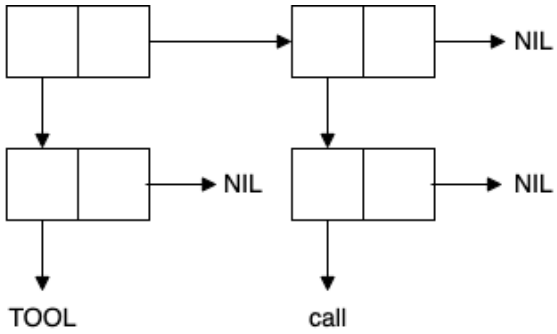

Рисунок 2.4

5. ’((TOOL1) ((call2)) ((sell)))

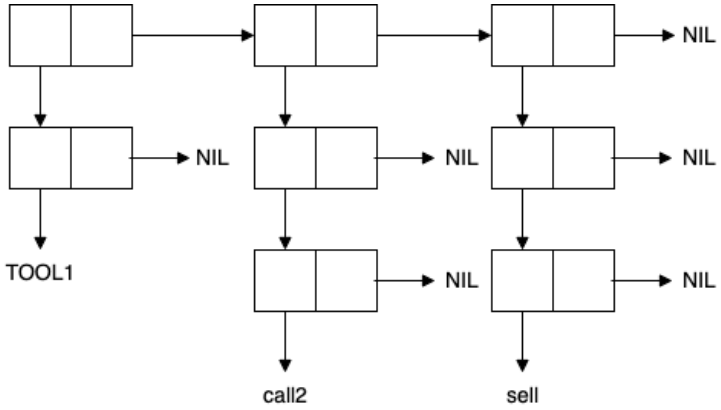

Рисунок 2.5

6. ’(((TOOL) (call)) ((sell)))

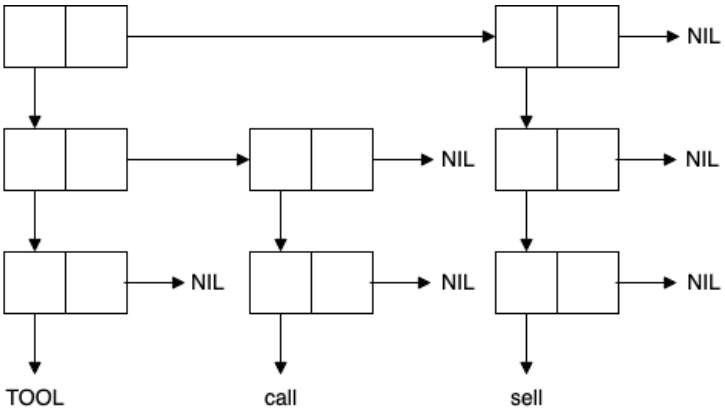

Рисунок 2.6

2.2 Используя только функции CAR и CDR, написать выражения, возвращающие i-ый элемент списка

1. второй

```
(CAR (CDR '(1 2 3)))        ; 2
(CADR     '(1 2 3) )        ; 2
```

2. третий

```
(CAR (CDR (CDR '(1 2 3))))  ; 3
(CADDR         '(1 2 3)  )  ; 3
```

3. четвертый

```
(CAR (CDR (CDR (CDR '(1 2 3 4 )))))  ; 4
(CADDR               '(1 2 3 4)    )  ; 4
```

2.3 Что будет в результате вычисления выражений?

1. (CAADR '((blue cube) (red pyramid)))

```
(CDR '((blue cube) (red pyramid)))      ; ((red pyramid))
(CAR '((red pyramid)))                   ; (red pyramid)
(CAR '(red pyramid))                     ; red

Ответ: red
```

2. (CDAR '((abc) (def) (ghi)))

```
(CAR '((abc) (def) (ghi)))              ; (abc)
(CDR '(abc))                            ; NIL

Ответ: NIL
```

3. (CADR '((abc) (def) (ghi)))

```
(CDR '((abc) (def) (ghi)))              ; ((def) (ghi))
(CAR '((def) (ghi)))                    ; (def)

Ответ: (def)
```

4. (CADDR '((abc) (def) (ghi)))

```
1   (CDR '((abc) (def) (ghi)))          ; ((def) (ghi))
2   (CDR '((def) (ghi)))                ; ((ghi))
3   (CAR '((ghi)))                      ; (ghi)
4
5   Ответ: (ghi)
```

2.4 Напишите результат вычисления выражений и объясните как он получен

Апостроф (quote) — блокирует вычисление своего аргумента.

Функция list создает и возвращает список, у которого голова — это первый аргумент, хвост — все остальные аргументы.

Функция cons включает новый элемент в начало списка. Если вторым аргументом передан атом, а не список, то создает точечную пару.

```
1   (list 'Fred 'and 'Wilma)            ; (Fred and Wilma)
2   (list 'Fred '(and Wilma))           ; (Fred (and Wilma))
3   (cons Nil Nil))                     ; (Nil)
4   (cons T Nil)                        ; (T)
5   (cons Nil T)                        ; (Nil . T)
6   (list Nil)                          ; (Nil)
7   (cons '(T) Nil)                     ; ((T))
8   (list '(one two) '(free temp))      ; ((one two) (free temp))
9   (cons 'Fred '(and Wilma))           ; (Fred and Wilma)
10  (cons 'Fred '(Wilma))               ; (Fred Wilma)
11  (list Nil Nil)                      ; (Nil Nil)
12  (list T Nil)                        ; (T Nil)
13  (list Nil T)                        ; (Nil T)
14  (cons T (list Nil))                 ; (T Nil)
15  (list '(T) Nil)                     ; ((T) Nil)
16  (cons '(one two) '(free temp))      ; ((one two) free temp)
```

2.5 Написать лямбда-выражение и соответствующую функцию

1. Написать функцию (f ar1 ar2 ar3 ar4), возвращающую список: ((ar1 ar2) (ar3 ar4))

```
(
defun
    f1
    (ar1 ar2 ar3 ar4)
    (list  (list ar1 ar2) (list ar3 ar4)  )
)
(f1 1 2 3 4)        ; ((1 2) (3 4))
```

```
(
    (
    lambda
        (ar1 ar2 ar3 ar4)
        (list  (list ar1 ar2) (list ar3 ar4)  )
    )
    1 2 3 4
)                   ; ((1 2) (3 4))
```

2. Написать функцию (f ar1 ar2), возвращающую ((ar1) (ar2))

```
(
defun
    f2
    (ar1 ar2)
    (list (list ar1) (list ar2))
)
f2 (1 2)          ; ((1) (2))
```

```
(
    (
    lambda
        (ar1 ar2)
        (list (list ar1) (list ar2))
    ) 1 2
)                 ; ((1) (2))
```

3. Написать функцию (f ar1), возвращающую (((ar1)))

```
(
 defun
    f3
    (ar1)
    (list (list (list ar1)))
)
f3(1)            ; (((1)))
```

```
(
    (
    lambda
        (ar1)
        (list (list (list ar1)))
    )
    1
)                  ; (((1)))
```

Представить результаты в виде списочных ячеек

1. ((ar1 ar2) (ar3 ar4))

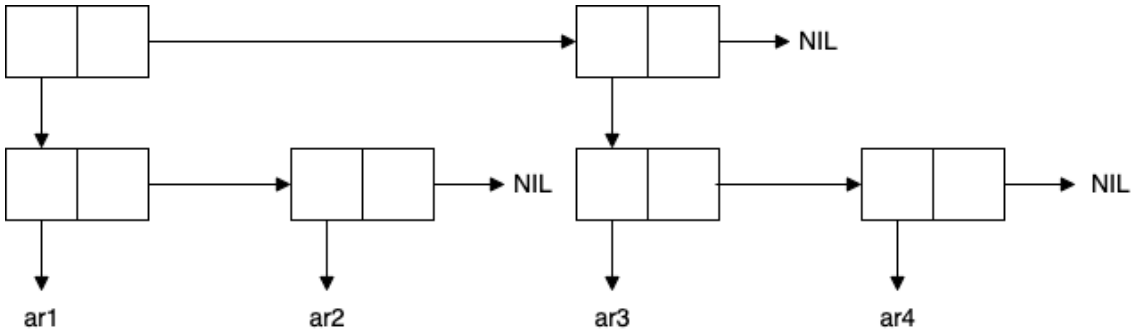

Рисунок 2.7

2. ((ar1) (ar2))

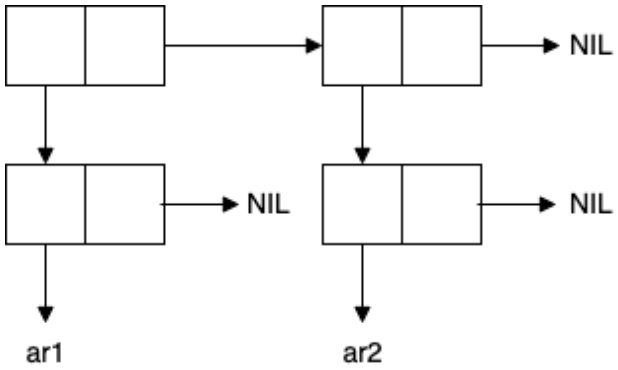

Рисунок 2.8

3. (((ar1)))

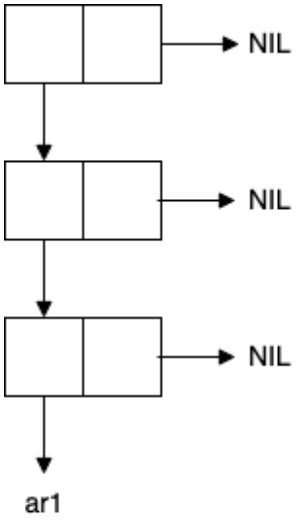

Рисунок 2.9