Дисциплина: Функциональное и логическое программирование

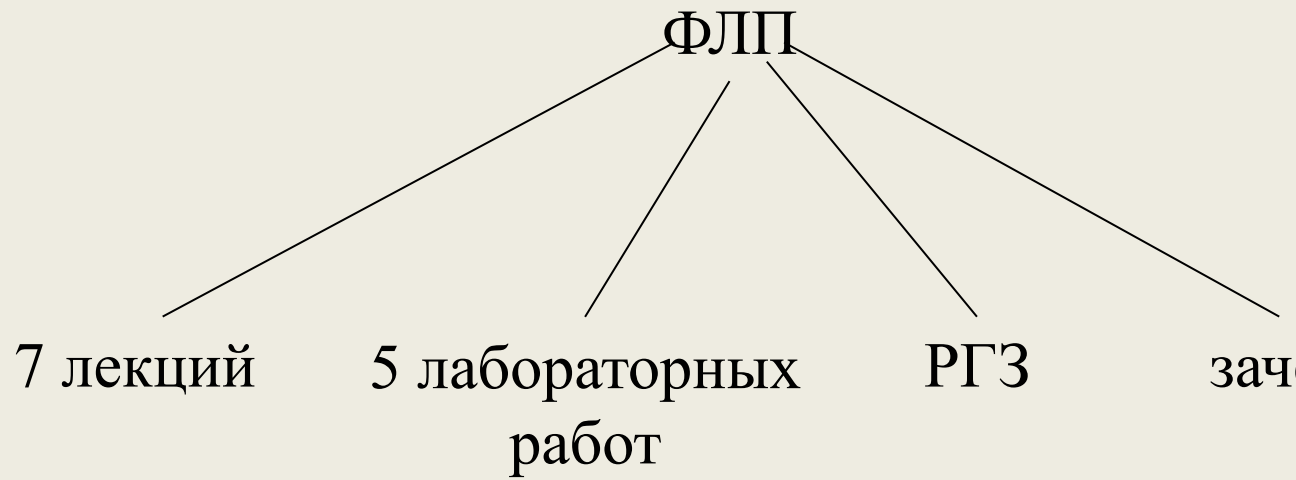
Лектор: Галкина Марина Юрьевна

Практические занятия:
группы ИП-811-812 Галкина Марина Юрьевна

группы ИП-813-814 Пащенко Анастасия Андреевна

группы ИП-815-816, ИВ-823 Агалаков Антон Александрович

группы ИВ-821-822, ИС-841 Бочкарев Борис Вячеславович



Максимальное количество баллов: $7+6 \cdot 5=37$

Зачет автоматом:

35-37 баллов + лабы + 2 задачи из РГЗ (на выбор)

31-34 балла + лабы + РГЗ

<31 балла + лабы + РГЗ + зачет (добираем баллы до 34, 1 задача – 2 балла)

Литература
(можно найти в библиотеке и Интернете):

1. Э.Хювёнен, Й.Сеппянен, Мир Лиспа, т.1, 2

2. И.Братко, Программирование на языке Пролог для искусственного интеллекта

Введение. Классификация языков программирования

- Процедурные (операторные);

  Бейсик, Паскаль, Си
- Непроцедурные:
  - Объектно-ориентированные;

    Object Pascal, C++, C#, Java, Python, Ruby
  - Декларативные:
    - функциональные (Lisp, Haskell, Erlang);
    - логические (Planner, Prolog).

Глава 1. Функциональное программирование. Основы языка Lisp

Язык Lisp (List processing) был разработан в Америке Дж.Маккарти в 1961 году (ориентирован на символьную обработку).

Свойства Lisp:

- Однообразная форма представления программ и данных.
- Использование в качестве основной управляющей конструкции рекурсии.
- Широкое использование данных «список» и алгоритмов их обработки.

Достоинства и недостатки Лиспа

Достоинство: простота синтаксиса

Недостатки:

- большое кол-во вложенных скобок
  (Lisp - Lots of Idiotic Silly Parentheses)
- множество диалектов

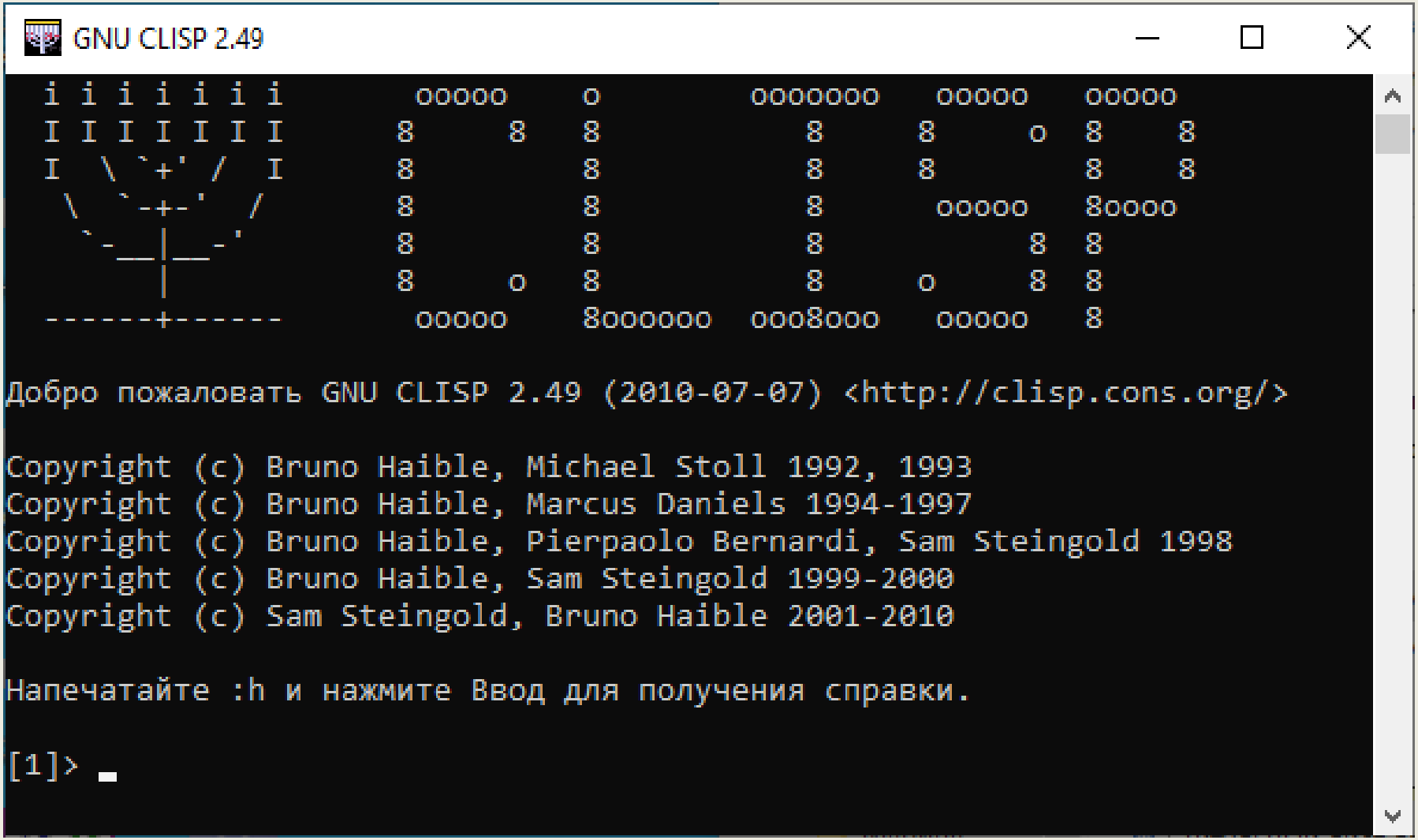
GNU Clisp 2.49

Реализован немецкими студентами Бруно Хайбле (Bruno Haible) и Михаэлем Штоллем (Michael Stoll). Он соответствует ANSI Common Lisp стандарту, работает под Unix, Windows и требует лишь 4 МБ памяти.

Запуск интерпретатора: clisp.exe

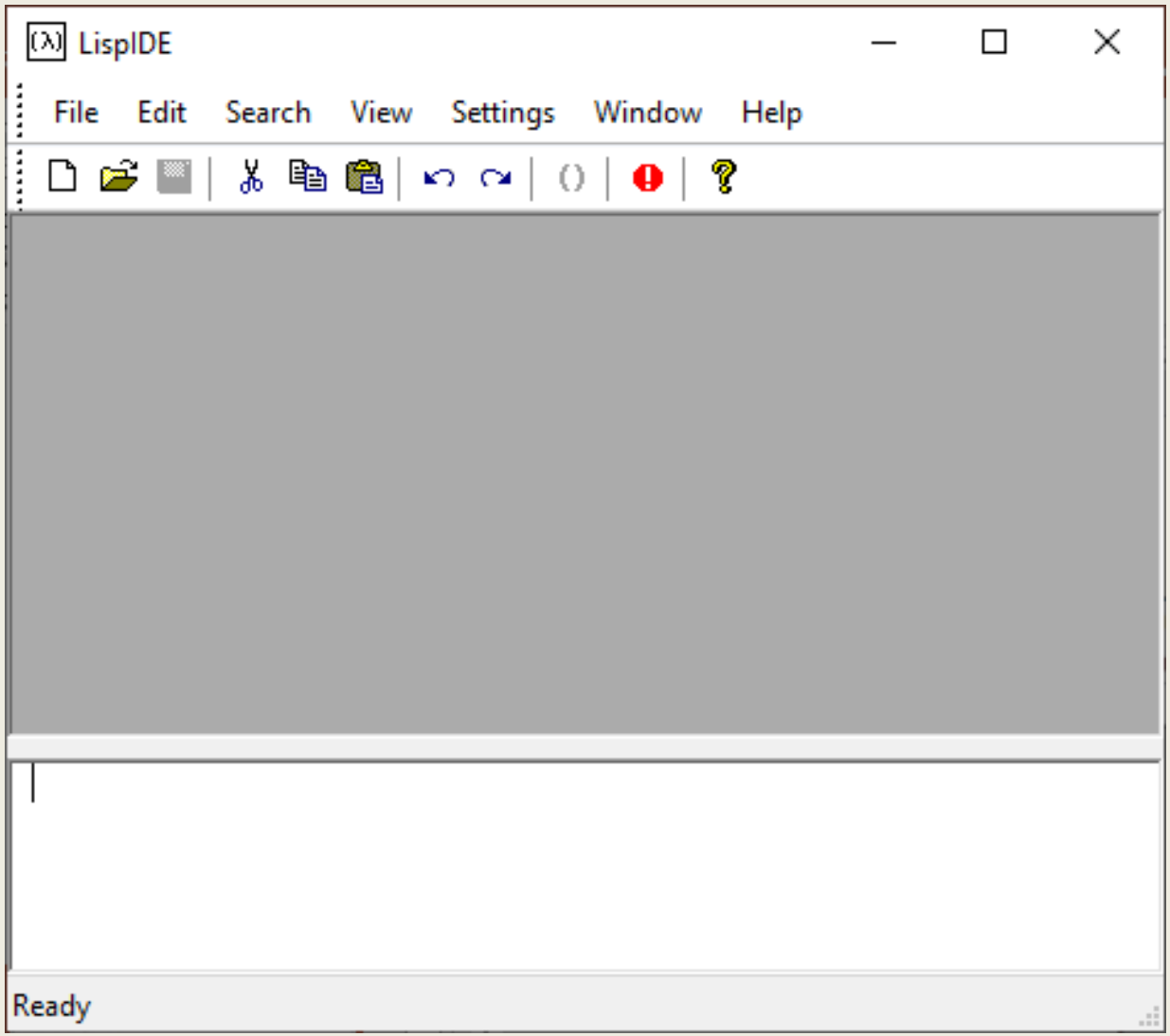
Если при выполнении команды возникла ошибка, то вернуться на предыдущий уровень (до ошибки) можно :r2

Выход: (exit) или (bye)

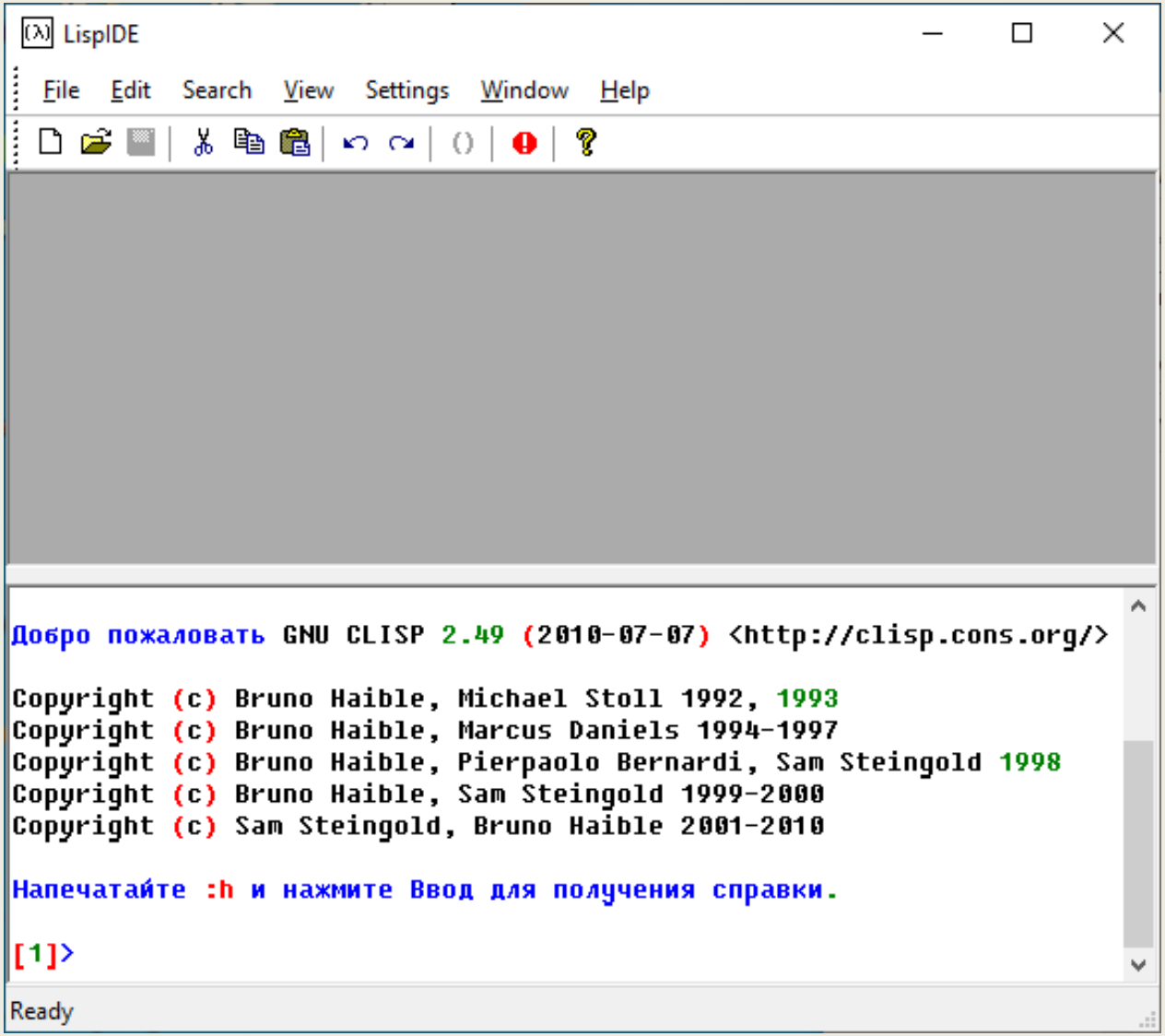

GNU CLISP 2.49
Добро пожаловать GNU CLISP 2.49 (2010-07-07) <http://clisp.cons.org/>
Copyright (c) Bruno Haible, Michael Stoll 1992, 1993
Copyright (c) Bruno Haible, Marcus Daniels 1994-1997
Copyright (c) Bruno Haible, Pierpaolo Bernardi, Sam Steingold 1998
Copyright (c) Bruno Haible, Sam Steingold 1999-2000
Copyright (c) Sam Steingold, Bruno Haible 2001-2010
Напечатайте :h и нажмите Ввод для получения справки.
[1]>

Редактор LispIDE

При его первом запуске (файл LispIDE.exe), запрашивается имя файла, который запускает интерпретатор Lisp.



После запуска редактора можно установить путь к интерпретатору выполнив команду Setting - Set List Path.



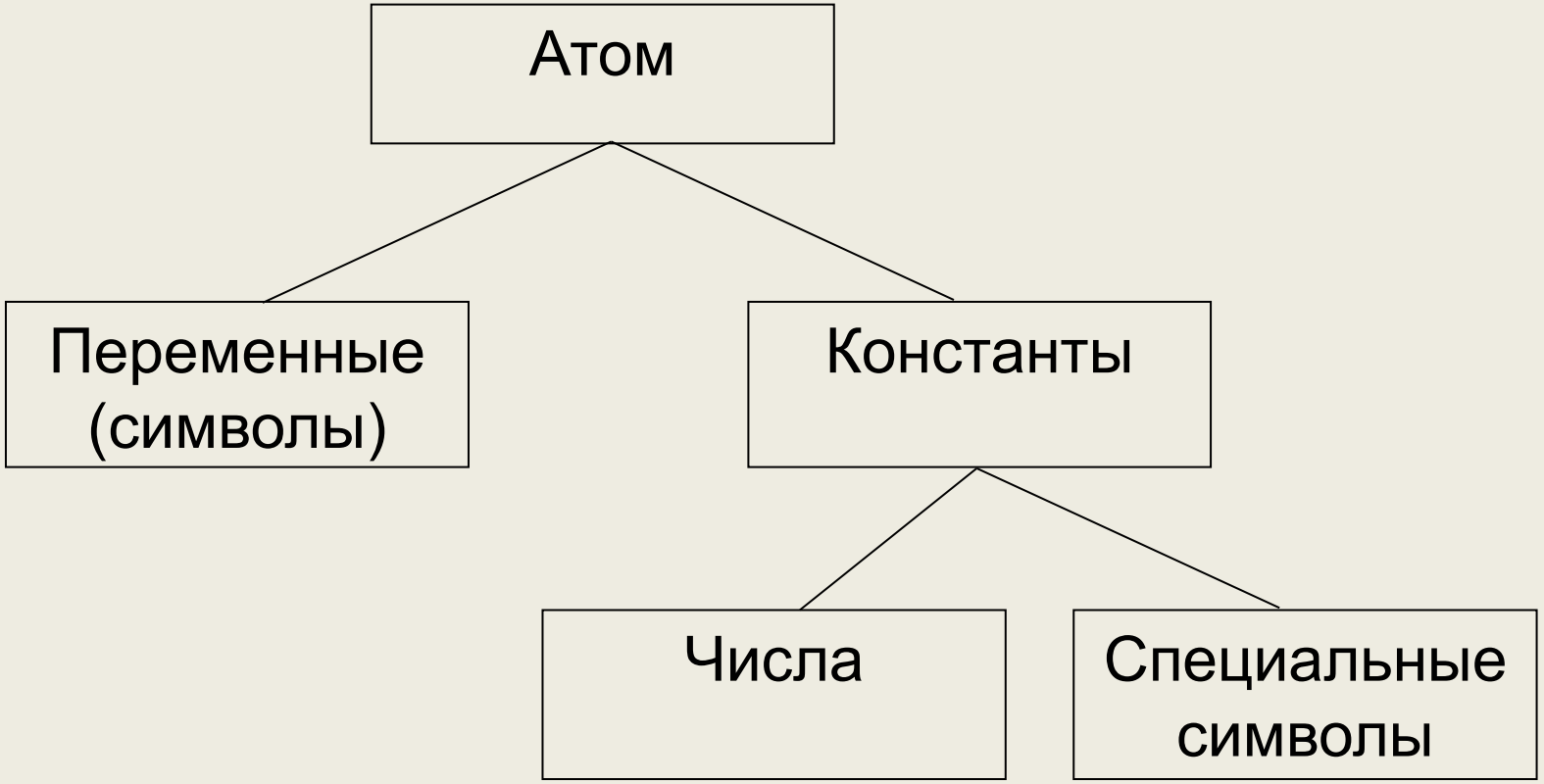
() – отправить интерпретатору выделение или до парной открывающейся скобки (если курсор стоит после закрывающейся скобки)

! – перезапустить интерпретатор

Сохранение только в англоязычные папки (и сам путь к папке должен быть англоязычным)

1.1 Типы данных в Lisp

Типы данных: атомы, списки, точечные пары



Атомы, списки – s-выражения

1.2 Функции

Вызов функции записывается в префиксной нотации.

(QUOTE s-выражение)

или

‘s-выражение

‘ - возле Enter

Математически функция QUOTE: f(x)=x

1.2.1 Арифметические функции

- +
- -
- *
- /
- ABS

Суперпозиция функций всегда вычисляется “изнутри наружу”.

1.2.2 Функции обработки списков

Голова и хвост списка.

(**CAR** список)

(**CDR** список)

Пример:

Выделить в списке ((a b c) (d e) (f)) элемент c;

(**CADDAR** '((a b c) (d e) (f)))

Пример: Выделить в списке (1(2 3((4 *)5)6))) элемент *.

(**CADAAR**(**CDDADR** '(1 (2 3 ((4 *) 5) 6))))

(**CONS** s-выражение список)

Функции **CAR** и **CDR** являются обратными для **CONS**

(**LIST** s_1 … s_n), где s_i – s-выражение

Пример: Из атомов 1, 2, 3, nil создадим список (1 (((2)) 3)) двумя способами:

а) с помощью композиций функций **CONS**;

б) с помощью композиций функций **LIST**.

а) (**CONS** 1 (**CONS** (**CONS** (**CONS** (**CONS** 2 nil) nil) (**CONS** 3 nil)) nil));

б) (**LIST** 1 (**LIST** (**LIST** (**LIST** 2)) 3)).

(**APPEND** $sp_1 \ldots sp_n$), где sp_i – список

(**LAST** список)

(**BUTLAST** список)

(**REVERSE** список)

1.3 Определение функций пользователем

1.3.2 Определение функций с именем

(**DEFUN** имя-функции лямбда-список S_1 S_2 … S_k)

Побочный эффект: связывание имени функции с лямбда-выражением
(LAMBDA лямбда-список S_1 S_2 … S_k)

Пример:

Определить функцию, которая меняет местами первый и четвертый элементы произвольного списка

```
(DEFUN ZAMENA (l)
  (APPEND
     (LIST (CADDDR l) (CADR l) (CADDR l) (CAR l))
     (CDDDDR l)
  )
)
```