Common Lisp

«Функции» языка

Универсальные

eval – вычислить форму

quote – заблокировать вычисления

Quote

Синтаксис

```
quote object => object
quote == '
```

Примеры

```
(quote (+ 3 4 5)) =>
'(+ 3 4 5) =>
''(+ 3 4 5) =>
(quote (quote (+ 3 4 5))) =>
```

Eval

Синтаксис

```
eval form => result
```

Примеры

```
(eval (+ 3 4 5)) => 12
(eval '(+ 3 4 5)) => 12
(eval ''(+ 3 4 5)) =>
```

Базовые

cons – сформировать списочную ячейку («точечную» пару); построитель списков

car – извлечь первый элемент списочной ячейки (первый элемент списка)

cdr – извлечь второй элемент списочной ячейки («хвост» списка)

eq – проверяет идентичность объектов

atom – проверяет атомарность объекта

+

defun и if (cons) и можно построить все!

Cons

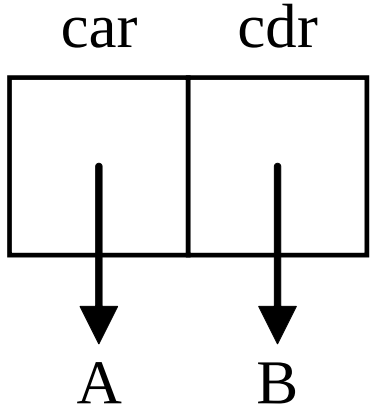
Синтаксис

```
cons object-1 object-2 => result
```

Примеры

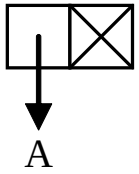
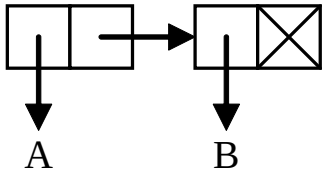
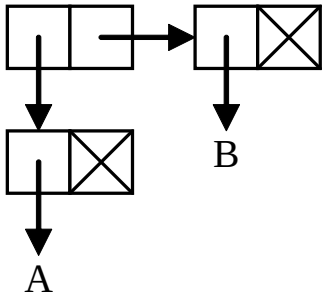
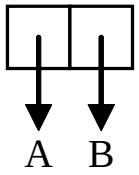
```
(cons 1 2) => (1 . 2)
(cons 1 '(2)) => (1 2)
(cons 1 '()) => (1)
(cons 3 '(A (B) C)) => (3 A (B) C)
```

Списочная ячейка и списки



Настоящий список – это либо пустой список (nil), либо списочная ячейка, первый элемент которой указывает на данные (возможно на другой список), а второй элемент указывает на другой настоящий список.

Внутреннее представление списков

Традиционная нотация	Графическое представление	Точечная нотация
`(A)`		`(A . nil)`
`(A B)`		`(A . (B . nil))`
`((A) B)`		`((A . nil) . (B . nil))`
`(A . B)`		`(A . B)`

Car

Синтаксис

```
car x => result || List => El
```

Примеры

```
(car nil) => nil
(car '(2)) => 2
(car '(1 . 2)) => 1
(car '(0 1 2 3 4)) => 0
(car ''(0 1 2 3 4)) => QUOTE
```

Cdr

Синтаксис

```
cdr x => result || List => List
```

Примеры

```
(cdr nil) => nil
(cdr '(2)) => nil
(cdr '(1 . 2)) => 2
(cdr '(0 1 2 3 4)) => (1 2 3 4)
(cdr ''(0 1 2 3 4))=>((0 1 2 3 4))
```

Eq

Синтаксис

```
eq x y => result
```

Примеры

```
(eq nil nil) => t
(eq '(1 . 2) '(1 . 2)) => nil
(eq '1 '1)) => t
```