

1. let в Scheme используется для создания локальных переменных и выполнения выражений внутри локальной области видимости.

let полезен, когда нужны временные переменные для выполнения операций или работы с данными в локальной области видимости, изолируя их от глобального контекста программы, позволяя избежать конфликта имен и других типичных ошибок.

```
(define circle-area radius)
  ((let ((pi 3.14) (area (* pi (* radius radius))))
    (display "Площадь равна ")
    (display area)))
(circle-area 3)
```

2.

$$1) a \rightarrow (1) \rightarrow (2) \rightarrow (3) \rightarrow 5$$

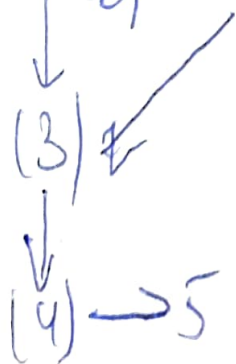
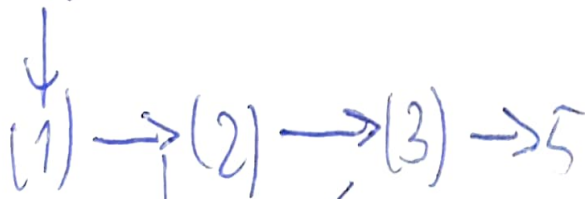
Задается a



$$(4) \rightarrow 5$$

$$2) b \rightarrow (2) \rightarrow (3) \rightarrow 5$$

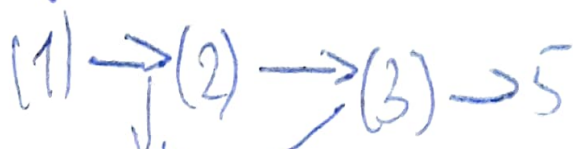
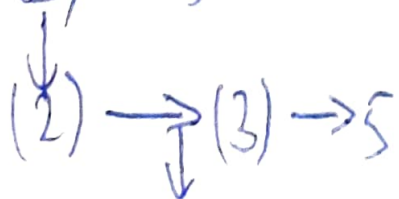
Задается b



$$(4) \rightarrow 5$$

$$3) c \rightarrow (3) \rightarrow 5$$

Задается c



3. Аппликативный порядок вычисления (Call by-value)

Это строгое вычисление, при котором аргументы вычисляются перед применением функции, и только затем в нее передаются

Порядок следующий:

- 1) Вычисление всех аргументов ф-ии
- 2) Применение ф-ии к аргументам

`(define (square x) (* x x))`

`(square (+ 2 5))` ; Сначала `(+ 2 5)`, а затем `(* 7 7)`

Вторичная доработка

`(* (+ 2 5) (+ 2 5))`

Примечания:

1) Предуказанность. Порядок вычисления не, что определяет отладку

2) Избегание лишнего вычисления (если ф-ия аргумент не использует)
используя ф-ию `if` использовать аргумент не стоит.
Рекурсия:

1) Не позволяет реализовать отмену, так как вычисление происходит сразу

2) Если вычисл. аргумента происходит сразу не арг. в ф-ии, то неэффективно