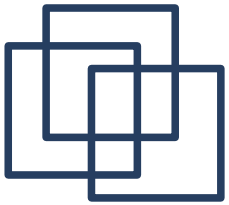
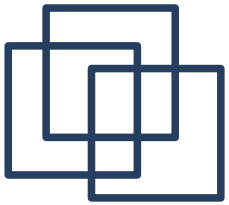


Кафедральный практикум 5 семестр

Часть 2. Язык Scheme (продолжение)

http://sp.cmc.msu.ru/~kornykhin/fp/slides/part2-2.pdf



План

Часть 1. Введение.

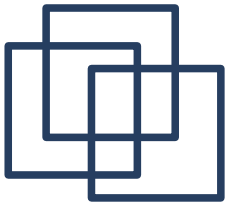
1) Организационные вопросы
2) Функциональный стиль программирования

Часть 2. Язык программирования Scheme.

1) **Быстрый старт**
2) **Более внимательный взгляд на Scheme**

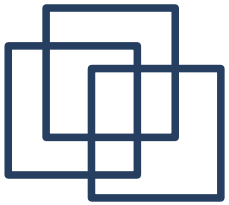
Часть 3. Функции высшего порядка. «Векторное» мышление.

Часть 4. Теоретический фундамент ФП.



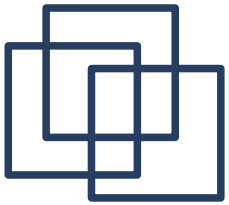
Исправление

- Предыдущее определение — бесстековая рекурсия
- Хвостовая рекурсия позволяет вызывать рекурсивные функции не только в return, но если это другая рекурсивная функция или невзаимнорекурсивная функция



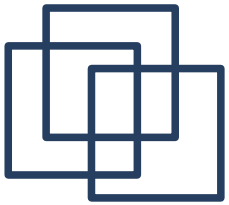
Анализ Д.3.

- Проверка списка на пустоту: (= (length x) 0) долго, лучше (null? x)
- Писать #lang racket (иначе проблемы с display, именами функций)
- Неполные постановки задач → уточнять постановку у нас (судьи в ejudge, kornevgen@cmc.msu.ru, amonakov@ispras.ru)



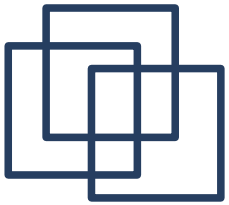
I2TP

- для тех, кто хочет!
- соединить практическую часть спецкурса Дениса Турдакова и практикум по функциональному программированию
- получаем 3 бонуса:
  - сдаем два курса сразу
  - получаем настоящую практику по своему интересу
  - изучаем язык Python на практике

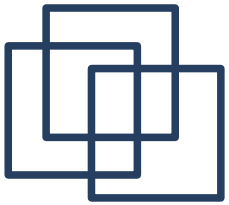


Задача

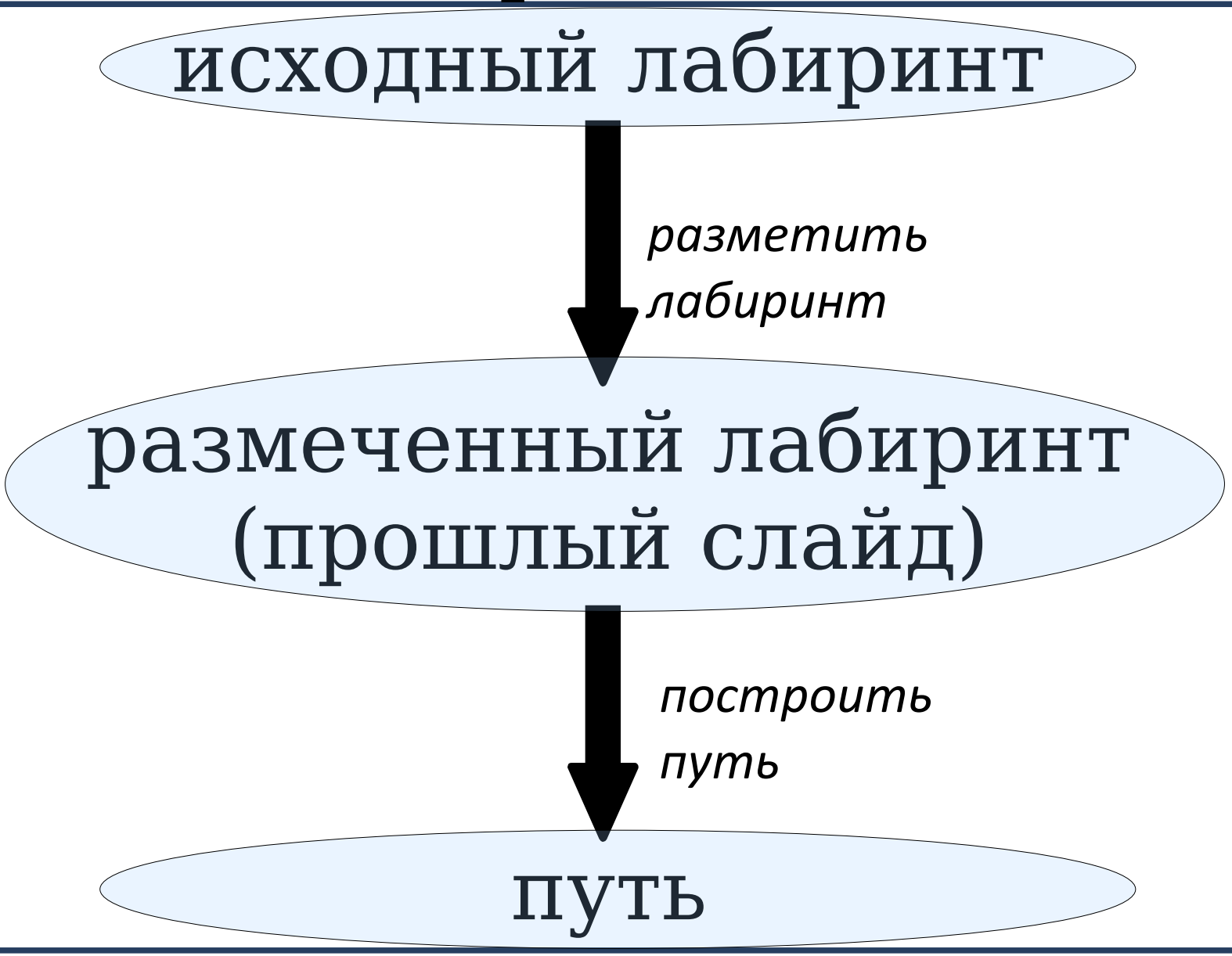
- Есть квадратное поле, в котором выделены
  - клетка-старт
  - клетка-финиш
  - клетки, на которых нельзя находиться ("запрещенные клетки").
- Построить путь из клетки-старт в клетку-финиш, не посещая запрещенные клетки, или сообщить, что такого пути нет.

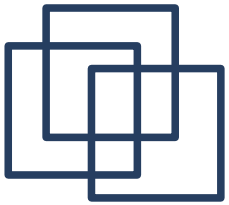


Идея решения



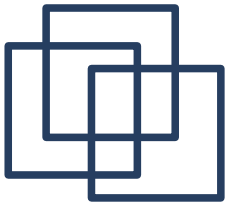
Идея решения





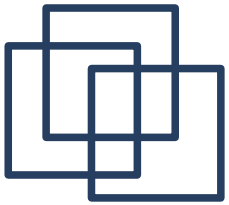
Уравнения

- *путь-в-лаб*(лаб) = *разметить*(лаб, (финиш), 0)
- *разметить*(лаб, верш, N) =
  разметить(разметить-вершины(лаб,верш,N),
  допустимые-соседи(верш, лаб),
  N + 1), если верш не пусто
- *разметить*(лаб, (), N) = если покрашен(старт, лаб) то *строить-путь*(лаб,финиш,()) иначе #f
- *строить-путь*(лаб, верш1, конец) = *строить-путь*(лаб, мин-сосед(верш1,лаб), cons(верш1,конец)), если верш1 — не старт
- *строить-путь*(лаб, старт, путь) = путь



Представление данных

- Представление неразмеченного лабиринта: ((1 2) (4 2)) список координат «запрещенных клеток»
- Представление размеченного лабиринта: **(((1 2) (4 2)) ((3 5)) …) список списков (внутренний список — места с одинаковой меткой: у первого списка метка равна 1, у следующего — 2 и т.д.)**



"Expression only"

имя функции *формальные параметры*

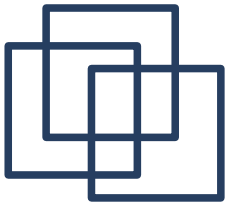
```
(define (function arg1 arg2 … argN)
  (let (   (var1 expr1)
           (var2 expr2)
              …
           (varM exprM) )

       (if  expr
            expr
            expr  ) ) )
```

локальные имена

условный оператор

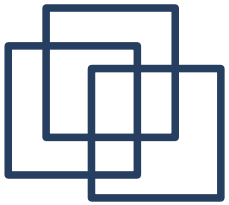
функция

- всюду списочный синтаксис



Demo

- Очень простой ввод/вывод; главное — закодить логику построения пути в лабиринте



Домашнее задание

- ejudge : http://earth.ispras.ru
- В решении запрещается использовать
  - set! и его аналоги
  - векторы