

**evpλu**

Графический интерпретатор языка LISP и  
платформа обучения функциональному  
программированию

ЧЕГАРКА.КО

# Что это такое?

define  
potion

$\lambda$   
bottle  
fruit

potion

-

+

✕

Сохранить

Загрузить

▼ Управляющие конструкции

define

lambda

quote

if

list

0

begin

▼ Программные примитивы

Строка

Символ

Число

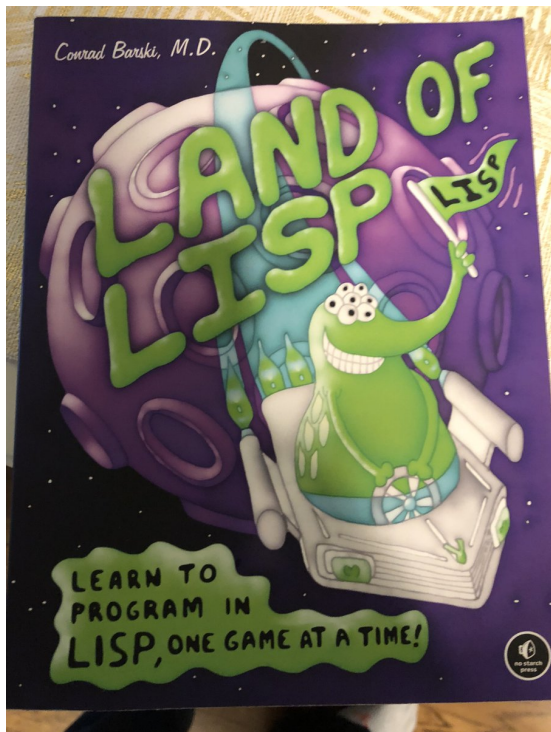
Изображение

Логическое значение

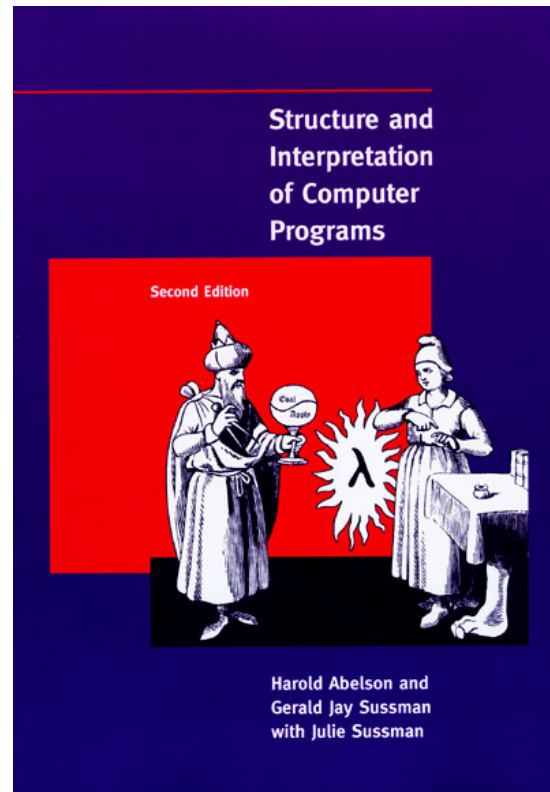
Зачем?

- За функциональным программированием будущее:
  - Выразительность => меньше кода
  - Типобезопасность (soundness) => меньше ошибок
  - Отсутствие изменяемого состояния программы (иммутабельность) => можно безопасно распараллелить

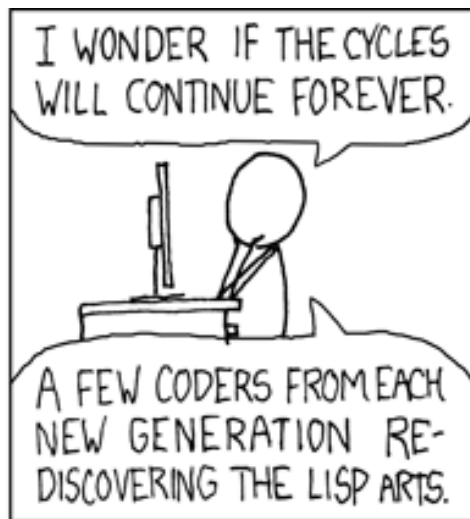
- На примере LISP можно обучить как основам алгоритмизации, так и принципам работы вычислительных систем



```
1 #lang racket
2 (require racket/osc 2htdp/image math/statistics)
3
4 (define (bay-of-biscay img)
5   (let* ([para (append-params (list (color->color-list img)) (cons 'height (image->height img))
6     (color->color-list img) (image-width img) (image-height img)))]
7     (color->color-list img) (image-width img) (image-height img))))
8
9 (define (get-alpha p) (cdr (assoc 'alpha p)))
10 (define (get-red-alpha p) (let ([o (cdr (assoc 'others p))]) (struct-copy color c [green o] [blue o] [alpha (get-alpha p)])))
11 (define (get-green-alpha p) (let ([o (cdr (assoc 'others p))]) (struct-copy color c [red o] [blue o] [alpha (get-alpha p)])))
12 (define (get-blue-alpha p) (let ([o (cdr (assoc 'others p))]) (struct-copy color c [red o] [green o] [alpha (get-alpha p)])))
13 (define (get-red-green p) (struct-copy color c [blue (cdr (assoc 'others p))]) [alpha (get-alpha p)])
14 (define (get-red-blue p) (struct-copy color c [green (cdr (assoc 'others p))]) [alpha (get-alpha p)])
15 (define (get-green-blue p) (struct-copy color c [red (cdr (assoc 'others p))]) [alpha (get-alpha p)])
16 (define (black-and-white-avg p)
17   (let ([g (exact-truncate (mean (list (color->red c) (color->green c) (color->blue c)))))
18         (color g g g (get-alpha p))])
19
20 (define source-path (string-append
21   (if (string=? (gethostname) "BayOfBiscay") "D:\\Omat\\\" " "C:\\")
22   "Voluptu\\Images\\BayOfBiscay\\protocol\\sample4.jpg"))
23 (define original (scale 0.17 (bitmap/file source-path)))
24 (define original-large (scale 1.7 original))
25 (beside
26   (above
27     (apply beside (map (curry map2 original '({others . 0} (alpha . 255))
28       (list get-red-alpha get-green-alpha get-blue-alpha)))
29     (apply beside (map (curry map2 original '({others . 0} (alpha . 255))
30       (list get-red-green get-red-blue get-green-blue)))))
31   (overlay/offset
32     (rotate 30 (flip-vertical (map2 original-large '({others . 255} (alpha . 255)) get-red-green)))
33     80 0 (rotate -30 (map2 original-large '({others . 255} (alpha . 255)) get-red-blue)))
34   (rotate 20 (overlay/offset
35     (flip-horizontal (map2 original-large '({alpha . 140}) black-and-white-avg))
36     10 -10 (flip-horizontal (map2 original-large '({others . 255} (alpha . 255)) get-green-blue)))))
37
38 > (gethostname)
39 "BayOfBiscay"
40
41 Determine language from source
```



- Это круто!



# Рабочее пространство


The screenshot shows a web browser window at `localhost:8080/#`. The main workspace has a light yellow background. At the top left, there is a toolbar with three icons: a cursor, a hand, and a magnifying glass. Below this toolbar, the text Управление поведением курсора is displayed. At the bottom left, the text Масштабирование области is shown above a small toolbar with minus and plus buttons. On the right side, a dark gray sidebar is open, featuring a close button (X) at the top right. The sidebar contains two buttons: `Сохранить` and `Загрузить`. Below these are two sections of controls:

- Управляющие конструкции** (Control Structures):
  - `define`, `lambda`, `quote`, `if`
  - `list`, `()`, `begin`
- Программные примитивы** (Program Primitives):
  - `Строка` (String), `Символ` (Symbol), `Число` (Number)
  - `Изображение` (Image), `Логическое значение` (Boolean value)

# Присвоить изображению имя - 1

localhost:8080

2. Выбрать картинку из предложенных



Выход

Сохранить Загрузить

Управляющие конструкции

- define
- lambda
- quote
- if
- list
- ()
- begin

Программные примитивы

- Строка
- Символ
- Число
- Изображение
- Логическое значение

1. Открыть меню



# Присвоить изображению имя - 2

The screenshot shows a web browser window at localhost:8080/#. The main area has a dark gray background. At the top, it says "3. Картинка появилась на экране" (The picture appeared on the screen) above a small red key icon. In the center, a light yellow dialog box is open with the text "ключ" (key) in a white input field and a "Принять" (Accept) button. Below the dialog, it says "5. Ввести название символа" (Enter the symbol name). On the right side, there is a sidebar menu with a close button (X) at the top. The menu contains two buttons: "Сохранить" (Save) and "Загрузить" (Load). Below these are two sections: "Управляющие конструкции" (Control structures) with buttons for "define", "lambda", "quote", "if", "list", "()", and "begin"; and "Программные примитивы" (Program primitives) with buttons for "Строка" (String), "Символ" (Symbol), "Число" (Number), "Изображение" (Image), and "Логическое значение" (Logical value). At the bottom right, it says "4. Открыть меню ввода символов" (Open the symbol input menu). The bottom of the main area has a dark gray bar with minus and plus buttons.

3. Картинка появилась на экране

5. Ввести название символа

4. Открыть меню ввода символов

Сохранить Загрузить

Управляющие конструкции

define lambda quote if

list () begin

Программные примитивы


Строка Символ Число

Изображение Логическое значение

# Присвоить изображению имя - 3

6. Символ появился на экране

7. Выделить объекты комбинацией клик+Shift

ключ 

8. Применить «define»

Сохранить Загрузить

Управляющие конструкции

- define lambda quote if
- list () begin

Программные примитивы

- Строка Символ Число
- Изображение Логическое значение

# Присвоить изображению имя - 4

←

→

↺

🏠

localhost:8080/#

⋮

🔒

★

↓

🔍

📄

☰

🖱️

👤

🔍

▶️

10. Исполнить код

define

КЛЮЧ 🔑

9. Объекты объединяются в новое выражение

-

+

✕

Сохранить

Загрузить

▼ Управляющие конструкции

define

lambda

quote

if

list

0

begin

▼ Программные примитивы

Строка

Символ



Число

Изображение

Логическое значение

# Присвоить изображению имя - 5

11. При обращении к символу «ключ» появляется графический объект



Сохранить

Загрузить

▼ Управляющие конструкции

define

lambda

quote

if

list

()

begin

▼ Программные примитивы

Строка

Символ

Число

Изображение

Логическое значение

# Технические задачи, с которыми мы столкнулись

- Написание простого интерпретатора
- DOM-независимый GUI
- Развертывание сервера, API для чтения/записи в БД Redis

# Реализовано из задуманного

- Редактор кода — готов полностью
- Интерпретатор языка — все работает, кроме анонимных функций, нет стандартных операций
- Серверная часть готова, но в проект не интегрирована
- Интерактивные задачи — готовы, но не интегрированы в среду

# Используемые технологии

- Клиентская часть — Haxe, PixiJS
- Серверная часть — Python, Django, Redis

# Что дальше?

- Добить стартовые задачи
- Переделать интерфейс и структуру с учетом полученного опыта
- Развить образовательную составляющую