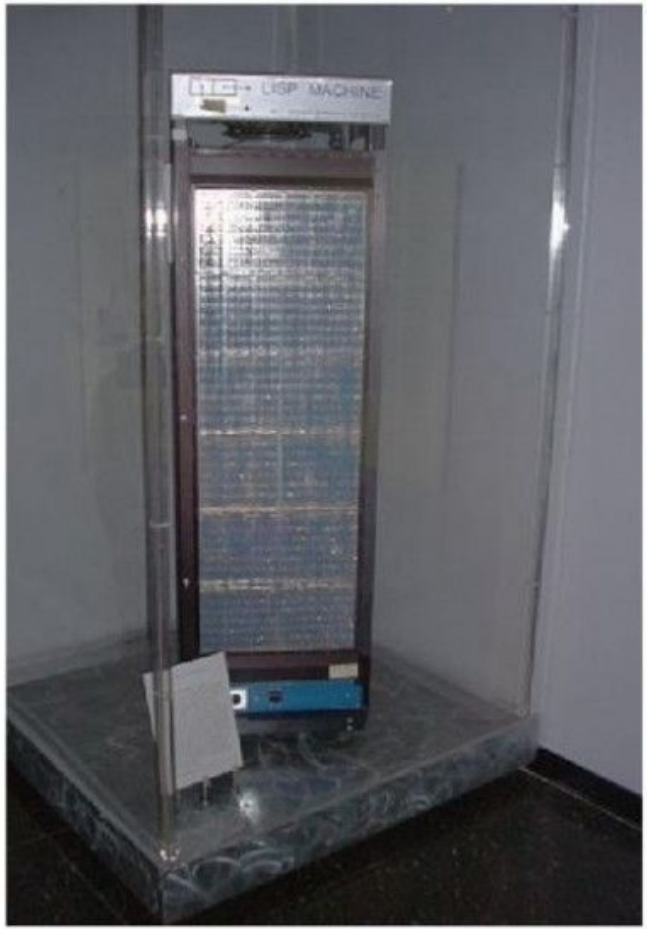
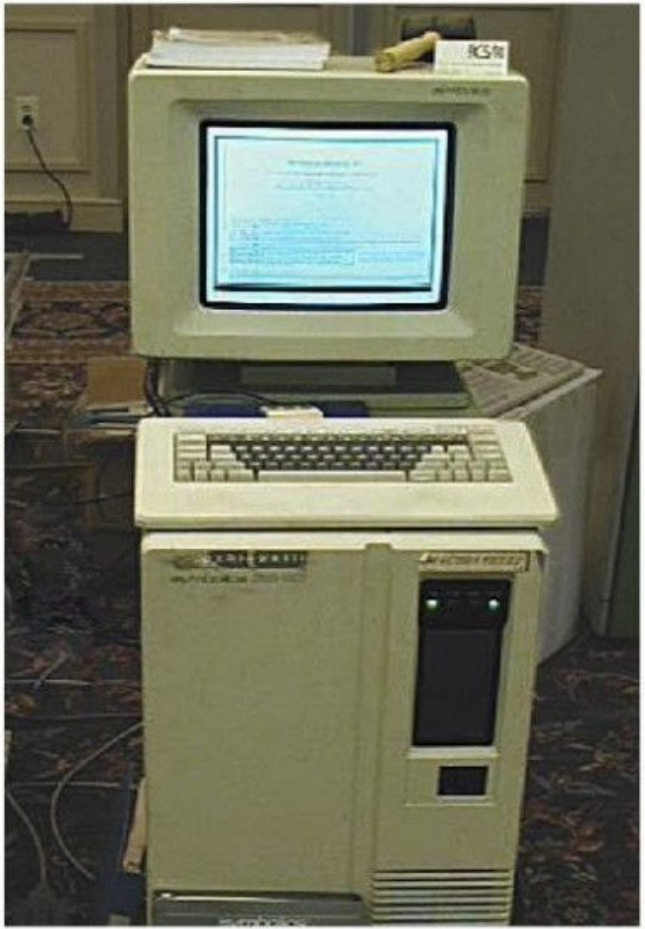
Сравнение Lisp и Haskell

Владимирова Элина
20.Б13-мм

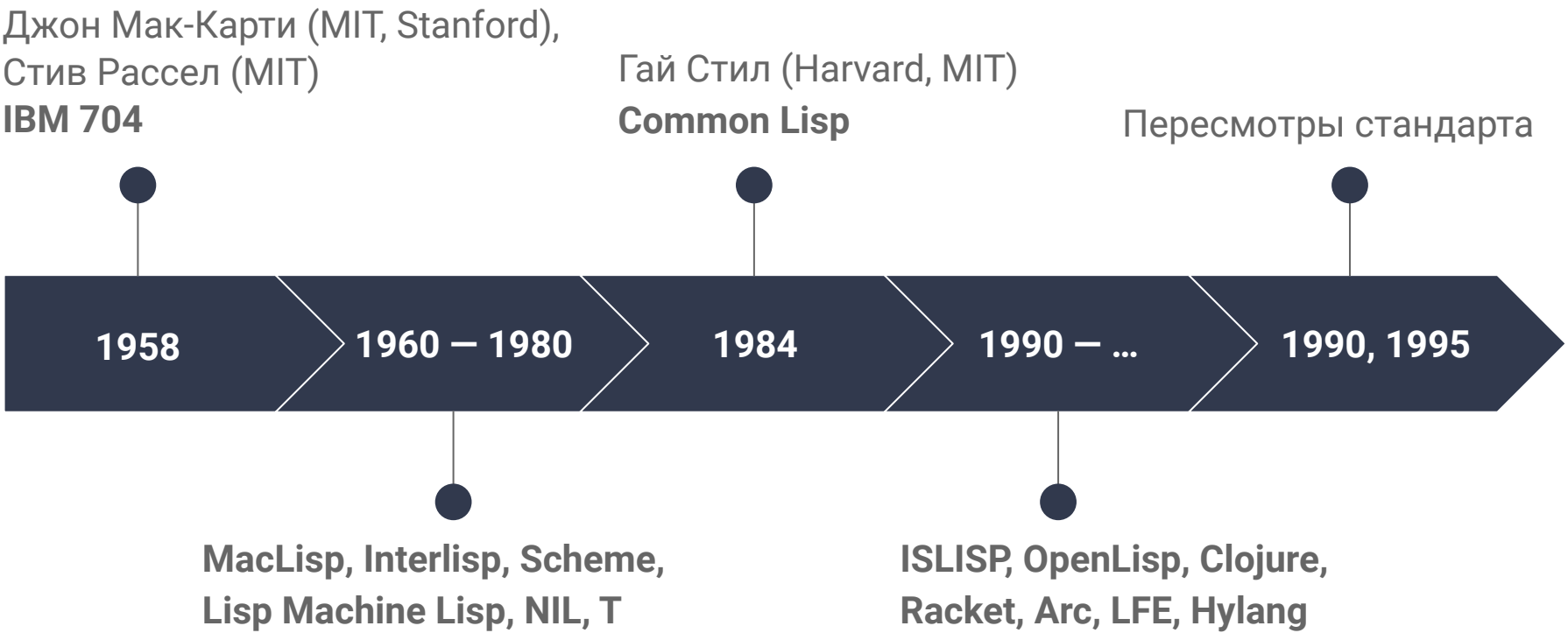


Лисп-машина в музее MIT



Лисп-машина Symbolics 3640

История


Джон Мак-Карти (MIT, Stanford),
Стив Рассел (MIT)
IBM 704
Гай Стил (Harvard, MIT)
Common Lisp
Пересмотры стандарта
1958
1960 — 1980
1984
1990 — ...
1990, 1995
MacLisp, Interlisp, Scheme,
Lisp Machine Lisp, NIL, T
ISLISP, OpenLisp, Clojure,
Racket, Arc, LFE, Hylang

Применение

- Искусственный интеллект (символьный подход)
  - +: Гибкость, эффективность, скорость
  - —: Низкая читаемость кода, сложность языка

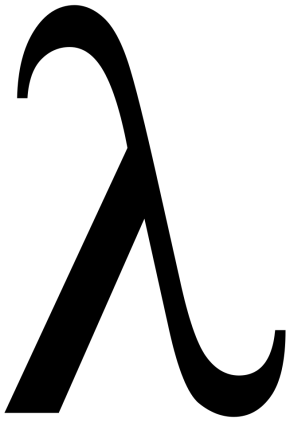
```
(defun make-perceptron (n m &optional (g  #'(lambda (i) (step-function 0 i)))
    &aux (l nil))

    (dotimes (i m (list l))
        (push (make-unit :parents (iota (1+ n))
            :children nil
            :weights (random-weights (1+ n) -0.5 +0.5)
            :g g)
        l)))
```

Применение

- Искусственный интеллект (символьный подход)
- Промышленное программирование (встроенные скрипты, веб-разработка, приложения)

Common Lisp

Clojure



Scheme

Racket

Применение

- Искусственный интеллект (символьный подход)
- Промышленное программирование (встроенные скрипты, веб-разработка, приложения)

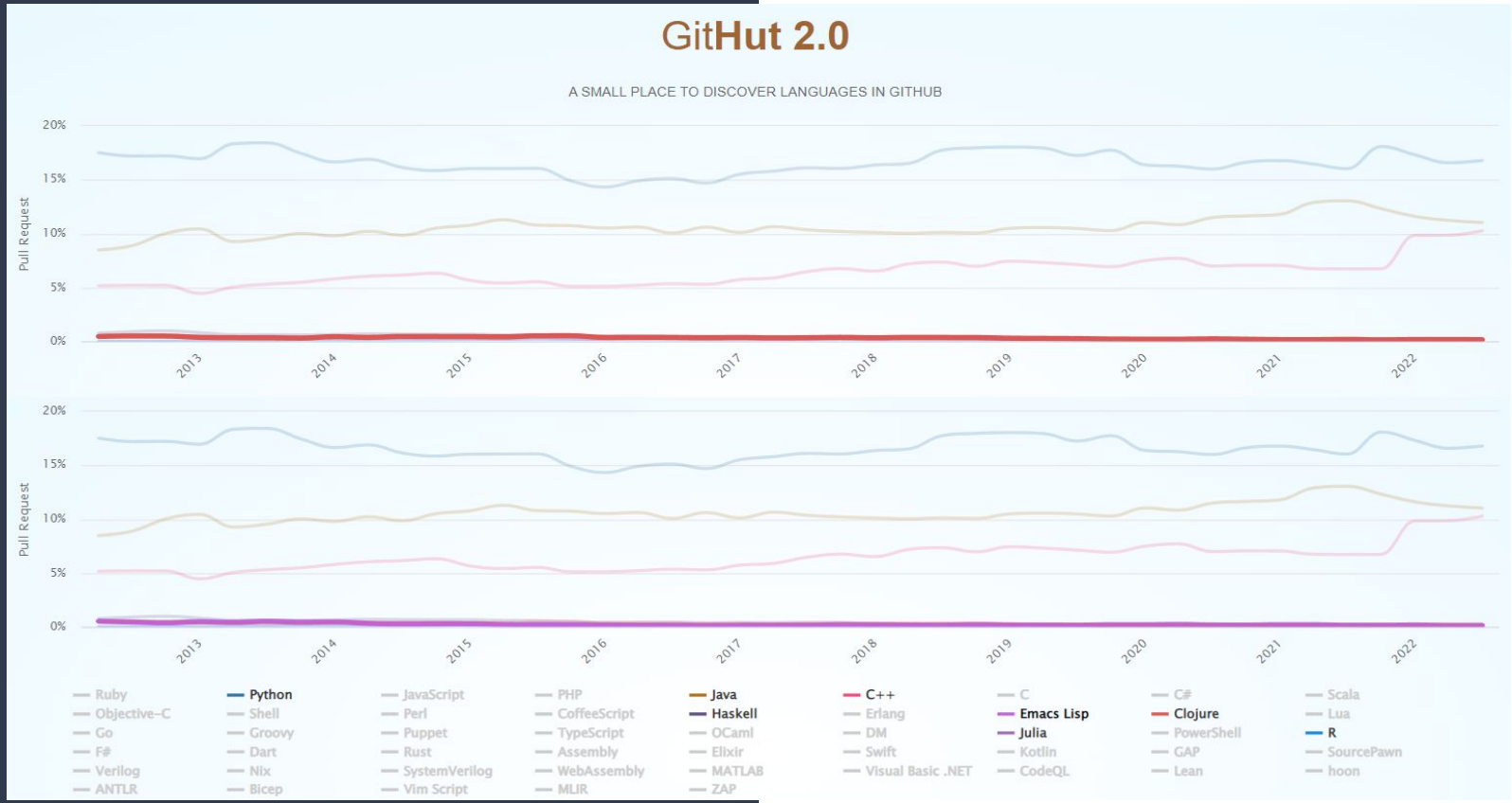
RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

APACHE STORM™

grammarly

ATLASSIAN

Применение


GitHut 2.0
A SMALL PLACE TO DISCOVER LANGUAGES IN GITHUB
Pull Request
20%
15%
10%
5%
0%
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
Ruby
Python
JavaScript
PHP
Java
C++
C
C#
Scala
Objective-C
Shell
Perl
CoffeeScript
Haskell
Erlang
Emacs Lisp
Clojure
Lua
Go
Groovy
Puppet
TypeScript
OCaml
DM
Julia
PowerShell
R
F#
Dart
Rust
Assembly
Elixir
Swift
Kotlin
GAP
SourcePawn
Verilog
Nix
SystemVerilog
WebAssembly
MATLAB
Visual Basic .NET
CodeQL
Lean
hoon
ANTLR
Bicep
Vim Script
MLIR
ZAP

Сравнение
Общие соображения

	Haskell	Lisp
Парадигма	Чисто функциональный язык	Мультипарадигмальный функциональный язык
Типизация	Статическая, строгая, явная	Динамическая, строгая, явная/неявная
Тип по способу запуска	Компилируемый (GHC) или интерпретируемый (HUGS)	Компилируемый или интерпретируемый
IDE	VSCode, IntelliJ, Emacs, Leksah, Atom, Vim, etc.	Emacs, Slime, Sublime Text, Eclipse Dandelion, Vim, etc.

Сравнение
Lisp императивный

- “**неявный progn**” — форма, возвращающая значение последнего переданного лямбда-выражения
- **progn (prog1, prog2, …)** — форма, возвращающая значение n-го переданного лямбда-выражения
- **let, defvar** — задание локальных переменных блока
- выполнение последовательных операций с переменными
- **set, setf, setq** — конструкции присваивания
- все виды циклов
  - **dotimes** (~ for i, i++), **loop for** (~ for i, i+=t),
  - **do** (~ do {...} while), **loop** (~ while),
  - **dolist** (~ for item:list)

Сравнение
Lisp императивный

```
1  ; have a = 0, b = 20
2  ; loop with a += 2, b -= 2
3  ; until (a == b)
4
5
6  (do ((a 0 (+ 2 a))
7     (b 20 ( - b 2)))
8     ((= a b)(- a b))
9     (format t "~% a = ~d   b = ~d" a b)
10 )
11 (write-line "")
```

Output:

```
a = 0   b = 20
a = 2   b = 18
a = 4   b = 16
a = 6   b = 14
a = 8   b = 12
```

Сравнение
Lisp императивный

```
1  ; with n = 3 loop over n
2  ; check necessary condition
3
4  (defvar n 3)
5  (loop
6      (print n)
7      (setq n (+ n 1))
8      (when (> n 6) (return n))
9  )
10 (write-line "")
11
12
13 ; loop over x from 12 to 25
14 ; by x += 3
15
16 (loop for x from 12 to 25 by 3 do
17     (print x)
18 )
19 (write-line "")
```

Output:

```
3
4
5
6

12
15
18
21
24
```

Сравнение
Макропрограммирование в Lisp

- Макросы задаются ключевым словом **defmacro** и имеют аналогичный функциям синтаксис
- Каждый вызов макроса раскрывает его код при трансляции так, будто этот код написан вместо вызова макроса
- Аргументы макросов по умолчанию не вычисляются

```
1  ; define new macros to construct function
2  ; from lambda-expression
3  ; define function with this macros
4  ; call function
5
6  (defmacro назвать (name lv)
7      (cons 'defun (cons name (cdr lv)))
8  )
9  (назвать сложить (lambda (x y) (+ x y)))
10
11 (print (сложить 12 8))
12 (write-line "")
```

```
Output:


20
```

- Макросы чтения

Сравнение
Lisp объектно-ориентированный

- ООП “уже было” в базовом Lisp
  - Свойства (слоты) символов ⇒ инкапсуляция
  - Свобода работы с функциями ⇒ связь кода (методов) с данными (объектами)
  - Динамический характер + функциональные свойства языка ⇒ полиморфное поведение кода и данных
- Принципы наследования реализованы дополнительно
- Известные промышленные ООП-расширения Lisp:
  - Flavors (диалект Lisp Machine Lisp)
  - LOOPS (диалект Interlisp)
  - CLOS (диалект Common LIsp)

Сравнение: user experience

Haskell	Lisp
Сложно учить	Сложно читать и писать
Красивый язык с лаконичным синтаксисом	Обратная польская запись операций, (шутки про (скобки)))
Среди функциональных языков рекомендуется учить первым для понимания принципов парадигмы	Чересчур гибок для обучения, но предоставляет свободу разработчику
Сложный процесс установки, настройки и отладки	Сложный процесс установки, настройки и отладки

Take Lisp, you know it's the most beautiful language in the world — at least until Haskell came along.

Larry Wall (создатель Perl)

Спасибо за внимание!