Lenguajes de Programación

Éric Tanter
etanter@dcc.uchile.cl
oficina 310

https://pleiad.cl/teaching/cc4101

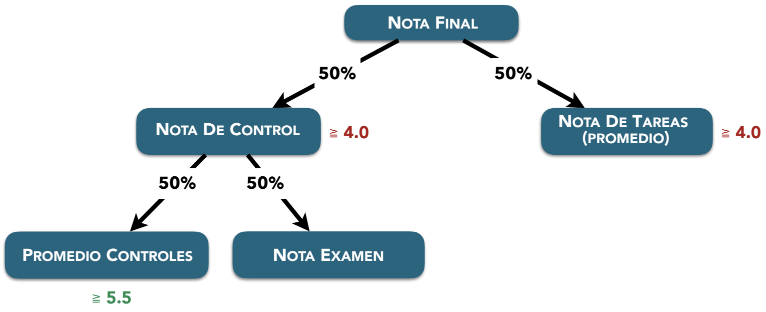
Clases Auxiliares

- Auxiliares: Bernardo Subercaseaux & Kenji Maillard
- Viernes horario 5.4 (excepto cuando hay control)
- *¡¡Empiezan este viernes 13 de marzo!!*

Evaluación

- 2 controles (s6 y s11) + examen
  - sin apuntes
  - nota de eximición 5.5
- 3 tareas (s4-5, s9-10, s13-14)
  - tareas se aprueban por separado
  - individuales (presentaciones)
  - atrasos: 0.5 pto / día, max 3 días
  - *no hay tarea recuperativa*

Evaluación



Aprobación del curso: Nota de control y nota de tareas ≥ 4.0 c/u

Eximición del examen: Promedio controles ≥ 5.5

Whiteboard Policy

https://pleiad.cl/teaching/cc4101

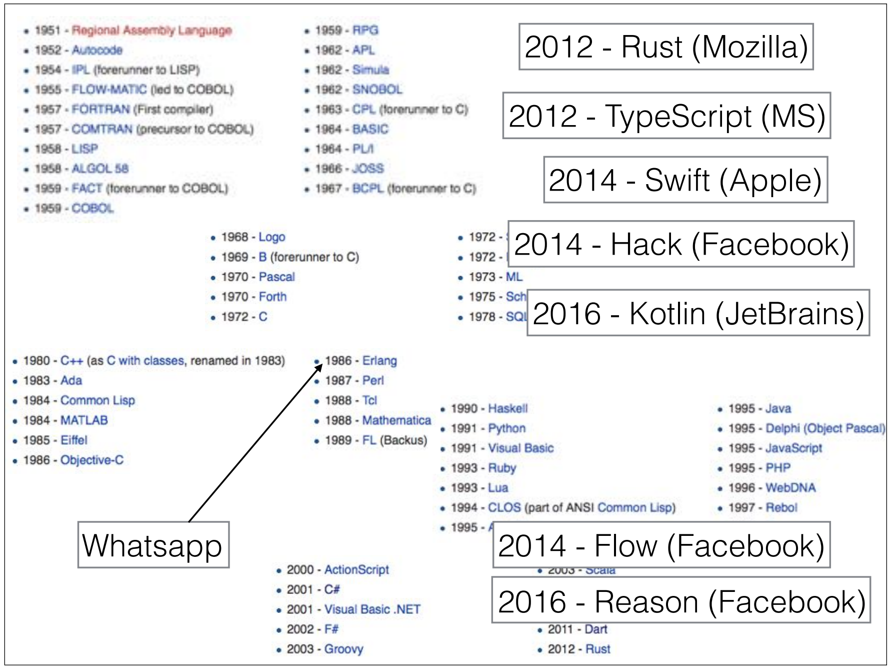
Programming

art?

engineering?

science?

Programming Languages



- 1951 - Regional Assembly Language
- 1952 - Autocode
- 1954 - IPL (forerunner to LISP)
- 1955 - FLOW-MATIC (led to COBOL)
- 1957 - FORTRAN (First compiler)
- 1957 - COMTRAN (precursor to COBOL)
- 1958 - LISP
- 1958 - ALGOL 58
- 1959 - FACT (forerunner to COBOL)
- 1959 - COBOL

- 1959 - RPG
- 1962 - APL
- 1962 - Simula
- 1962 - SNOBOL
- 1963 - CPL (forerunner to C)
- 1964 - BASIC
- 1964 - PL/I
- 1966 - JOSS
- 1967 - BCPL (forerunner to C)

- 1968 - Logo
- 1969 - B (forerunner to C)
- 1970 - Pascal
- 1970 - Forth
- 1972 - C

- 1972 -
- 1972 -
- 1973 - ML
- 1975 - Sch
- 1978 - SQL

- 1980 - C++ (as C with classes, renamed in 1983)
- 1983 - Ada
- 1984 - Common Lisp
- 1984 - MATLAB
- 1985 - Eiffel
- 1986 - Objective-C

- 1986 - Erlang
- 1987 - Perl
- 1988 - Tcl
- 1988 - Mathematica
- 1989 - FL (Backus)

- 1990 - Haskell
- 1991 - Python
- 1991 - Visual Basic
- 1993 - Ruby
- 1993 - Lua
- 1994 - CLOS (part of ANSI Common Lisp)
- 1995 - A

- 1995 - Java
- 1995 - Delphi (Object Pascal)
- 1995 - JavaScript
- 1995 - PHP
- 1996 - WebDNA
- 1997 - Rebol

- 2000 - ActionScript
- 2001 - C#
- 2001 - Visual Basic .NET
- 2002 - F#
- 2003 - Groovy

- 2003 - Scala
- 2011 - Dart
- 2012 - Rust

2012 - Rust (Mozilla)

2012 - TypeScript (MS)

2014 - Swift (Apple)

2014 - Hack (Facebook)

2016 - Kotlin (JetBrains)

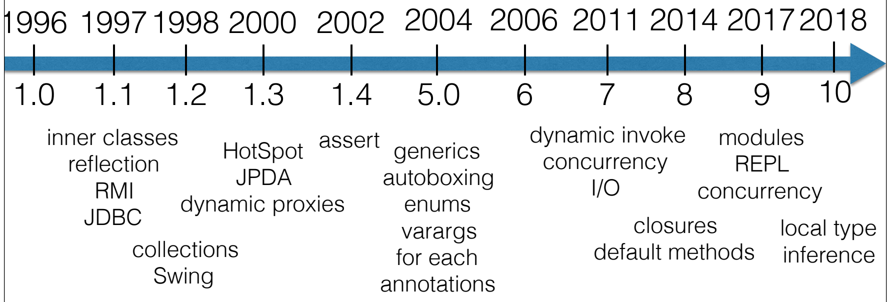
Whatsapp

2014 - Flow (Facebook)

2016 - Reason (Facebook)

Language evolution

Java



1996	1.0	
1997	1.1	inner classes reflection RMI JDBC
1998	1.2	collections Swing
2000	1.3	HotSpot JPDA dynamic proxies
2002	1.4	assert
2004	5.0	generics autoboxing enums varargs for each annotations
2006	6	
2011	7	dynamic invoke concurrency I/O
2014	8	closures default methods
2017	9	modules REPL concurrency
2018	10	local type inference

11…12…13…

Possible future features:
records, pattern matching, ADTs

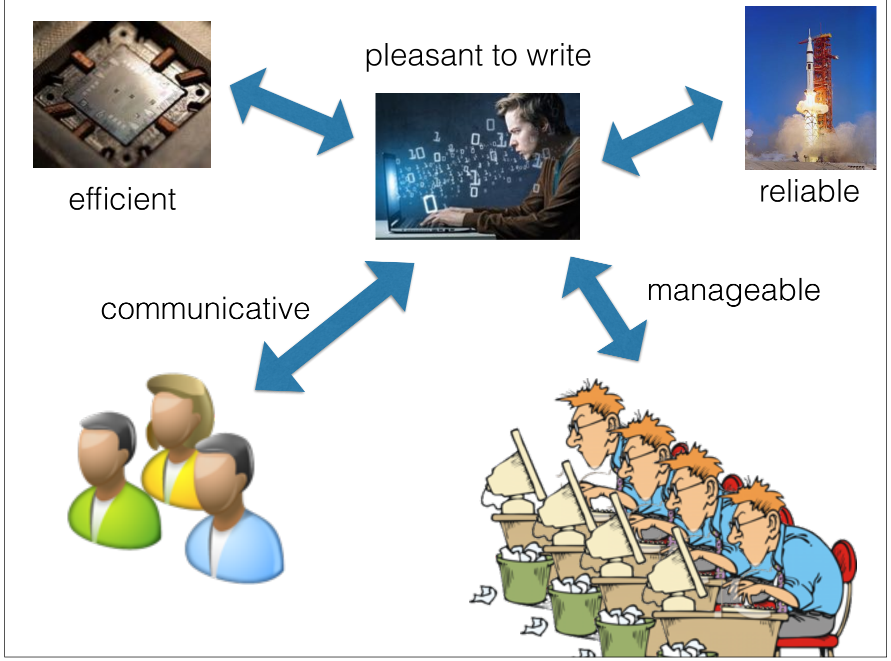
In theory…

- all languages are “equivalent”
  - they are all Turing complete
  - ie. they can all compute the same things

Programming Languages

why so many?

why so many changes?



pleasant to write

efficient

reliable

communicative

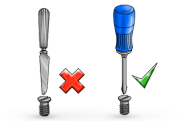
manageable

A most important, but also most elusive,
aspect of any tool is its influence
on the habits of those who train
themselves in its use.

If the tool is a programming language,
this influence is, whether we like it or not,
an influence on our thinking habits.

— Dijkstra

- **Know your tools**
  - Choose the right one for the right task
  - Use the tools effectively



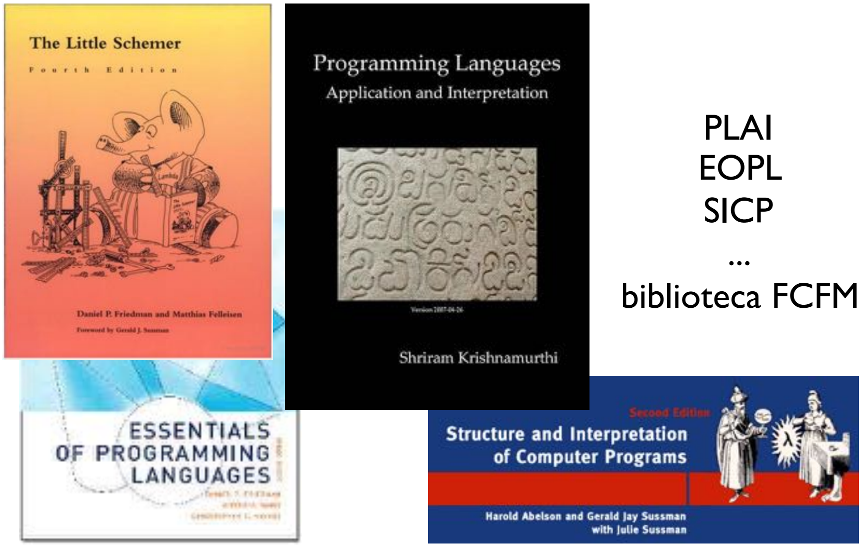
- **Be prepared for the future**
  - New languages keep coming

Temario

- Paradigmas vs. mecanismos
- Interpretación de lenguajes: variaciones semánticas
  - funciones, alcance, regimen de evaluación, recursión, mutación, objetos, tipos, compilación, etc.
  - extender un lenguaje: compilación y macros

C, Java, Lisp, Scheme, bash, Haskell, ML,
JavaScript, Self, Smalltalk, Python, Scala, Racket...

Bibliografía


The Little Schemer
Fourth Edition
Daniel P. Friedman and Matthias Felleisen
Foreword by Gerald J. Sussman
Programming Languages
Application and Interpretation
Shriram Krishnamurthi
ESSENTIALS OF PROGRAMMING LANGUAGES
Second Edition
Structure and Interpretation of Computer Programs
Harold Abelson and Gerald Jay Sussman
with Julie Sussman

PLAI
EOPL
SICP
...
biblioteca FCFM

A Programming Language?

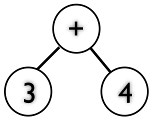
- Peculiar syntax
- Behavior associated to the syntax
- Useful libraries
- Programming idioms

Syntax

- Does not tell much about behavior of programs
- E.g. which two are most similar?
  - a [25] + 5
  - (+ (vector-ref a 25) 5)
  - a [25] + 5
- Point: *express and understand more by saying less*

Modeling Syntax

- Don't be too emotional about syntax
  - 3 + 4 *infix*
  - 3 4 + *postfix*
  - (+ 3 4) *parenthesized prefix*
  - all this *means* the same! the idealized action of adding the idealized numbers (represented by) "3" and "4":



Libraries and Idioms

- Libraries are important to programmers
  - not so relevant for a language study
- Idioms are interesting sociologically

Just Semantics!

Modeling Meaning

- Which language to use to describe meaning?
  - natural language is not well-suited
- Existing formalisms
  - denotational semantics
  - operational semantics
  - axiomatic semantics
  - *interpreter semantics*

Interpreter Semantics

- To explain a language, write an interpreter for it!
  - writing forces understanding (like mathematics)
  - once written, interpreter can be *executed*
  - allows for *incremental* modifications/exploration
- But: interpreter is a program, written in a language!
  - practice: use a simple, succinct, well-understood language
  - theory: mathematical foundations of this language has been built already

λ Scheme

- Dialect of Lisp, developed during 1975-80
- Based on Church's lambda calculus
- Minimalist design:
  - small core + powerful tools for extension (macros)

www.racket-lang.org

Racket solve p
AGES
DOWNLOAD
Batteries included
Powerful macros & languag
DrRacket IDE & tons of doc
COMMUNICATIONS OF THE ACM
CACM.ACM.ORG
03/2018 VOL.61 NO.03
A Programmable Programming Language
How Can We Trust a Robot?
Policy Intervention Fosters Cybersecurity Intervention
Here Comes Everyone ... to Communications
Q&A with Yann LeCun