Introduzione al corso di Informatica 3

Angelo Gargantini

AA 2019/2020

M0 Introduzione su syllabus

Argomenti

- Concetti generali dei linguaggi di programmazione
  - Computabilità
  - Tipi e sicurezza dei tipi
  - Visibilità, decomposizione
  - Semantica dei programmi
  - Usando il C principalmente
- Object Oriented programming
  - ereditarietà, polimorfismo
  - Java
  - C++
- Funzionale (Scala)
- Abstract state machines

Mappa del corso

- Vedi xmind

Docenti

- Io, angelo.gargantini@unibg.it
- cs.unibg.it/gargantini

Libri di testo per info3

- Concepts in Programming Languages (Cambridge Univ Press, 2002) John C. Mitchell
  - (disponibile in pdf)
    - Programming Language Concepts, Ghezzi e Jazayeri, Wiley
    - Programming Languages, Sebesta, Addison Wesley
    - Advances in Programming languages, Finkel, Addison Wesley – si può scaricare
- Versione in Italiano in pdf disponibile con tutto il materiale

Altre info

- ricevimento
  - Martedì pomeriggo alle 17.00
- Laboratori, pochissime volte
  - per fare esercizi
-

Materiale

- Registro: su google calendar

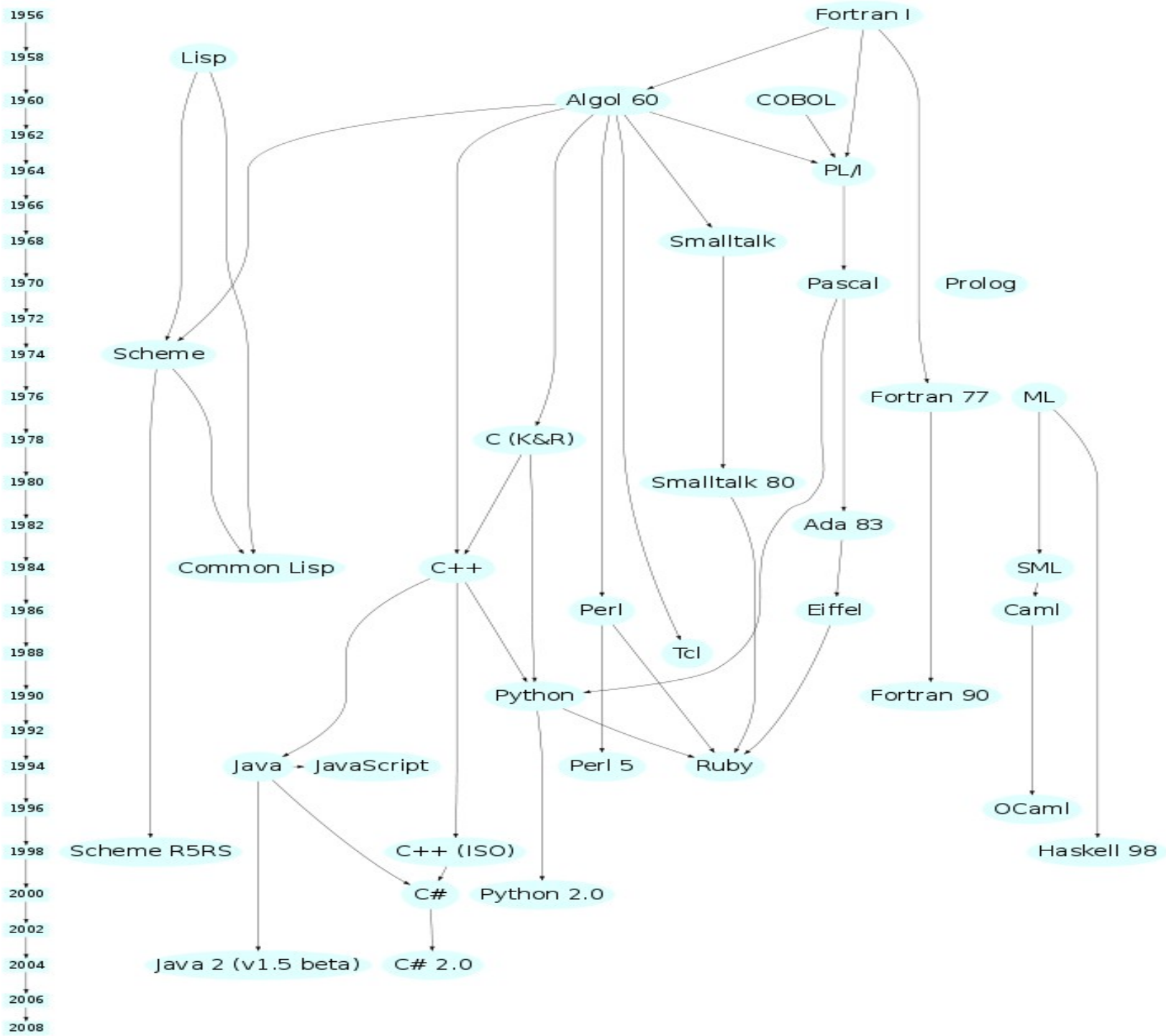
- Dal sito web cs.unibg.it/gargantini/
  - Appunti delle lezioni, come pdf,
    - Aggiornamenti sono benvenuti
  - Syllabus: indice degli argomenti con i riferimenti al meteriale dove studiare
  - pdf delle slides lezioni (vecchi prima, aggiornati dopo)
  - User info30809
  - Password jcminfo3
- Codice su dropbox
  - https://www.dropbox.com/sh/wz4dcehp5ga7fig/cTm16u2naU

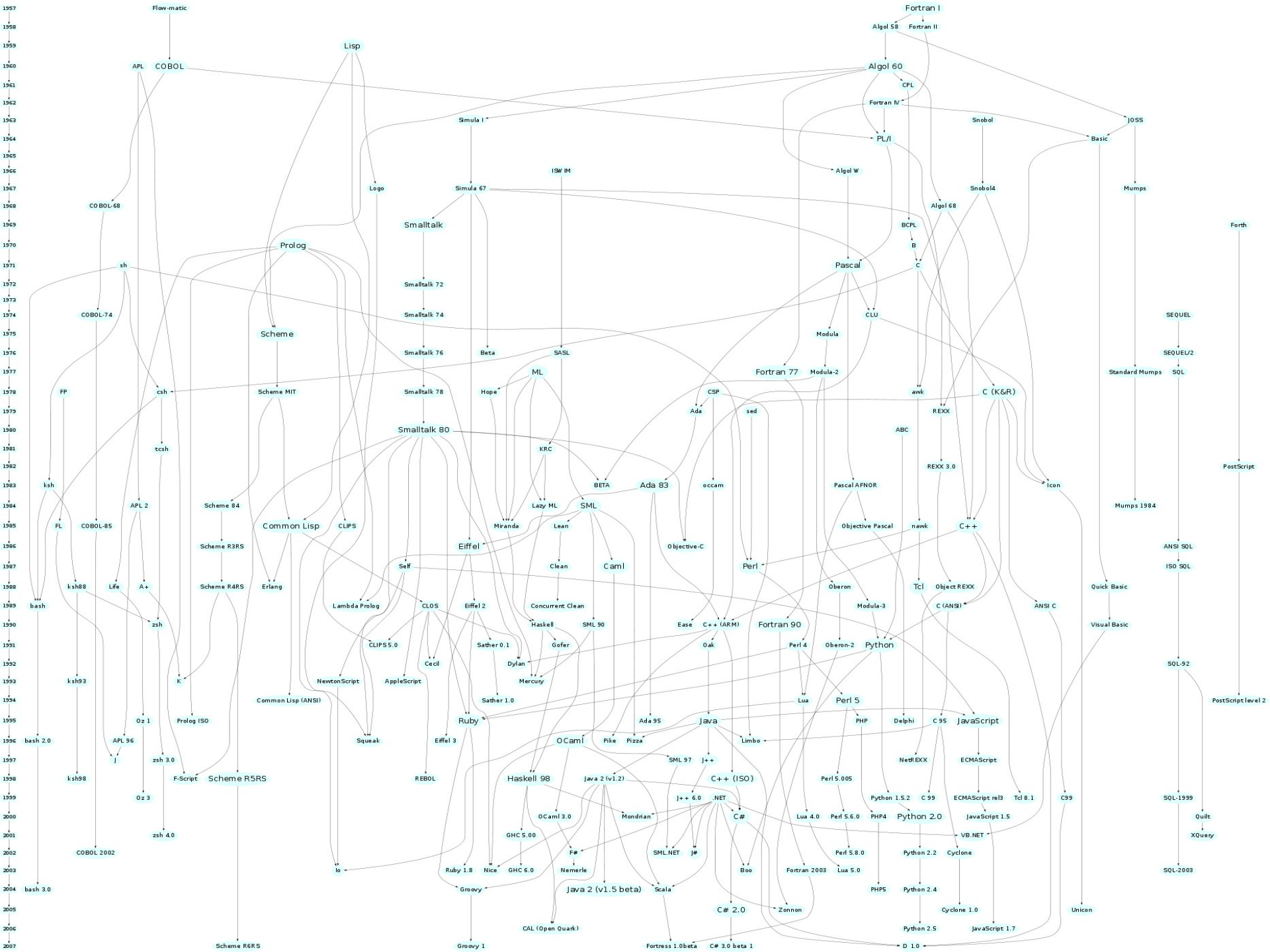
Esame di Info3

- L'esame è composto da quattro parti
  - Domande teoriche tipo quiz su ilias
  - Esercizi pratici al computer su ilias con eclipse
  - Domande orali
  - Progettini
- Peso indicativo:
  - Ogni prova da un voto fino a 33 (minimo 14)
  - Con peso diverso:
    - Domande teoriche: 0.25
    - Esericizi programmazione: 0.4
    - Orale: 0.2
    - Progettini: 0.15

Iniziamo ...
Linguaggi di programmazione

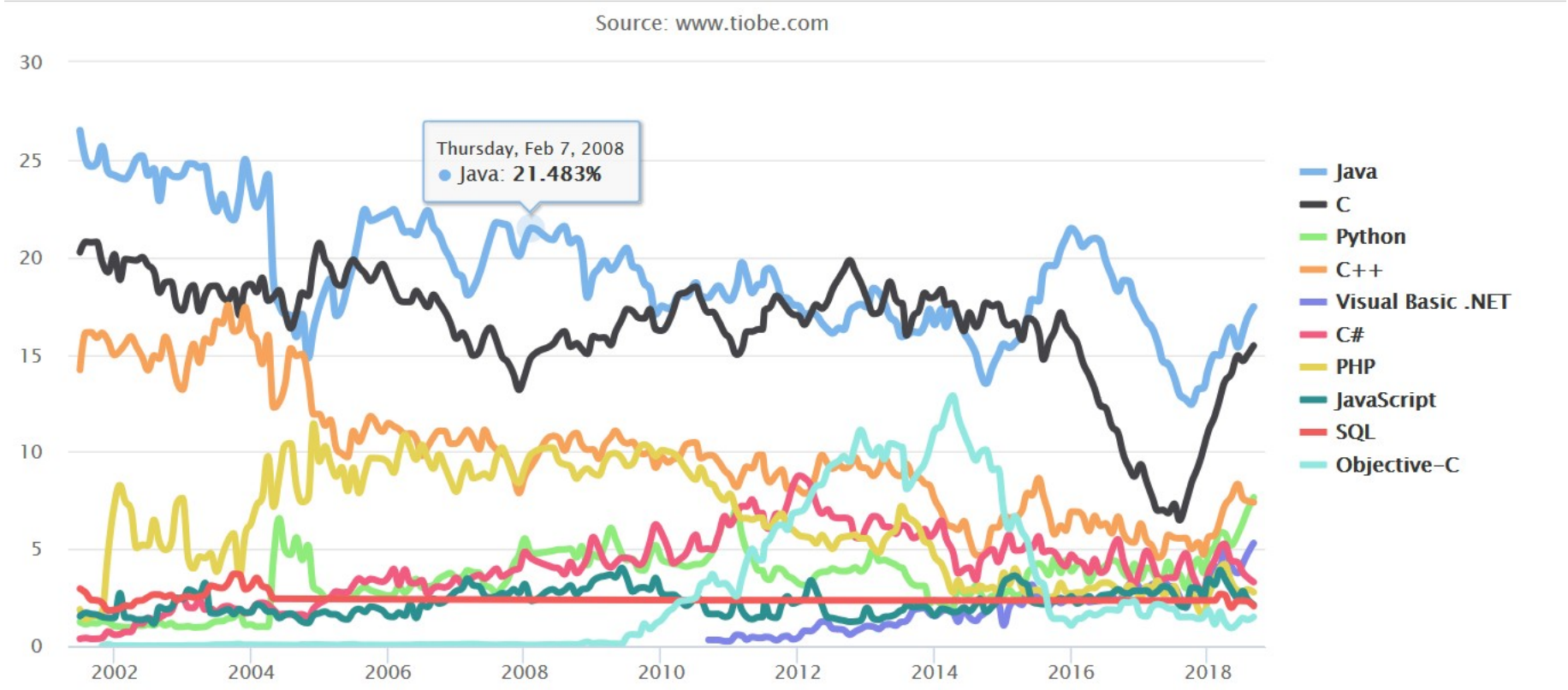
- Libro 1.3
- wikipedia
- C'è anche una lista
- C'è anche in time table ...


1956
1958
1960
1962
1964
1966
1968
1970
1972
1974
1976
1978
1980
1982
1984
1986
1988
1990
1992
1994
1996
1998
2000
2002
2004
2006
2008
Fortran I
Lisp
Algol 60
COBOL
PL/I
Smalltalk
Pascal
Prolog
Scheme
Fortran 77
ML
C (K&R)
Smalltalk 80
Ada 83
Common Lisp
C++
SML
Perl
Eiffel
Caml
Tcl
Python
Fortran 90
Java
JavaScript
Perl 5
Ruby
OCaml
Scheme R5RS
C++ (ISO)
Haskell 98
C#
Python 2.0
Java 2 (v1.5 beta)
C# 2.0


1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
Flow-matic
Fortran I
Algol 58
Fortran II
Lisp
APL
COBOL
Algol 60
CPL
Fortran IV
Simula I
Snobol
JOSS
PL/I
Basic
Algol W
ISWIM
Logo
Simula 67
Snobol4
Mumps
Algol 68
COBOL-68
Smalltalk
BCPL
B
Forth
Prolog
C
sh
Pascal
Smalltalk 72
Smalltalk 74
CLU
SEQUEL
COBOL-74
Modula
Scheme
Smalltalk 76
Beta
SASL
SEQUEL/2
Fortran 77
Modula-2
Standard Mumps
SQL
FP
csh
Scheme MIT
Smalltalk 78
ML
Hope
awk
C (K&R)
CSP
Ada
sed
REXX
Smalltalk 80
ABC
tcsh
KRC
PostScript
ksh
BETA
Ada 83
occam
Pascal APNOR
Icon
Scheme 84
APL 2
Lazy ML
SML
Mumps 1984
FL
COBOL-85
Common Lisp
CLIPS
Miranda
Lean
Objective Pascal
nawk
C++
Scheme R3RS
Eiffel
Objective-C
ANSI SQL
Self
Clean
Caml
Perl
ISO SQL
ksh88
Life
A+
Scheme R4RS
Erlang
Oberon
Tcl
Object REXX
Quick Basic
bash
Lambda Prolog
CLOS
Eiffel 2
Concurrent Clean
C (ANSI)
Modula-3
ANSI C
zsh
Haskell
SML 90
Ease
C++ (ARM)
Fortran 90
Oberon-2
Visual Basic
CLIPS 5.0
Sather 0.1
Gofer
Oak
Perl 4
Python
SQL-92
ksh93
K
Cecil
Dylan
NewtonScript
AppleScript
Mercury
Common Lisp (ANSI)
Sather 1.0
Lua
Perl 5
PostScript level 2
Oz 1
Prolog ISO
Ruby
Ada 95
Java
PHP
Delphi
C 95
JavaScript
bash 2.0
APL 96
Squeak
Eiffel 3
OCaml
Pike
Pizza
Limbo
NetREXX
ECMAScript
J
SML 97
J++
zsh 3.0
ksh98
F-Script
Scheme R5RS
REBOL
Haskell 98
Java 2 (v1.2)
C++ (ISO)
Perl 5.005
Python 1.5.2
C 99
ECMAScript rel3
Tcl 8.1
SQL-1999
Oz 3
J++ 6.0
.NET
C#
Perl 5.6.0
PHP4
Python 2.0
JavaScript 1.5
Quilt
zsh 4.0
OCaml 3.0
Mondrian
Lua 4.0
VB.NET
XQuery
COBOL 2002
GHC 5.00
F#
SML.NET
J#
Perl 5.8.0
Python 2.2
Cyclone
SQL-2003
bash 3.0
Io
Ruby 1.8
Nice
GHC 6.0
Nemerle
Boo
Fortran 2003
Lua 5.0
PHP5
Python 2.4
Groovy
Java 2 (v1.5 beta)
Scala
Cyclone 1.0
C# 2.0
Zonnon
Unicon
CAL (Open Quark)
Python 2.5
JavaScript 1.7
Scheme R6RS
Groovy 1
Fortress 1.0beta
C# 3.0 beta 1
D 1.0

http://www.tiobe.com

Quali sono quelli più diffusi??



E' un po' una moda

- Year Winner
- 2017 C
- 2016 Go
- 2015 Java
- 2014 JavaScript
- 2013 Transact-SQL
- 2012 Objective-C
- 2011 Objective-C
- 2010 Python
- 2009 Go
- 2008 C
- 2007 Python
- 2006 Ruby
- 2005 Java
- 2004 PHP
- 2003 C++

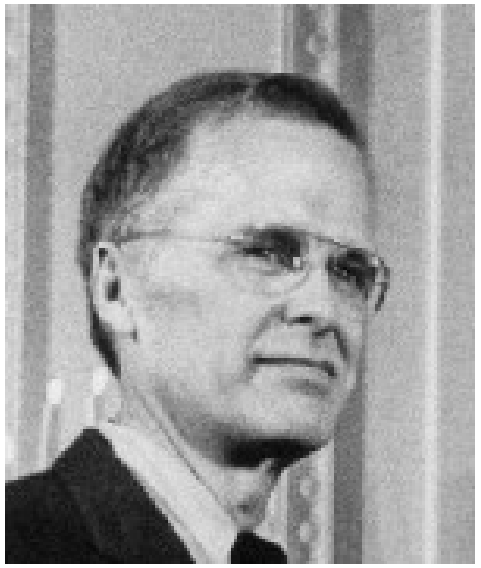
Programming paradigms

- Procedural/Imperative-style programming
  - FORTRAN, Algol, Pascal, C, ...
- Functional/Applicative-style programming
  - LISP, Scheme, ML, Haskell, ...
- Declarative/Logic programming
  - Prolog, ...
- Object-oriented programming
  - C++, C#, Java, ...
- Hybrids
  - concurrent, parallel, dataflow, intensional, domainspecific,
    ...
    scripting & extension languages

Key language milestones

- Assembly languages
  - invented by machine designers in the early 1950s
  - shift from binary machine code to mnemonics
  - first occurrence of reusable macros & subroutines
- FORTRAN - FORmula TRANslation
  - designed by John Backus at IBM in the mid-1950s
  - first high-level “algebraic” language with a compiler
- LISP - LISt Processor
  - designed by John McCarthy in 1958
  - first language to be based on the theory of recursive functions

FORTRAN



- John Backus, b. 1924
  - 1977 Turing Award
    On FORTRAN: “We did not know what we wanted and how to do it. It just sort of grew. The first struggle was over what the language would look like. Then how to parse expressions - it was a big problem and what we did looks astonishingly clumsy now.... “
- Defined BNF: “The syntax and semantics of the proposed international algebraic language of the Zurich ACM GRAMM conference.” ICIP Paris, June 1959.
  - influenced by Chomsky’s work on context-free grammars

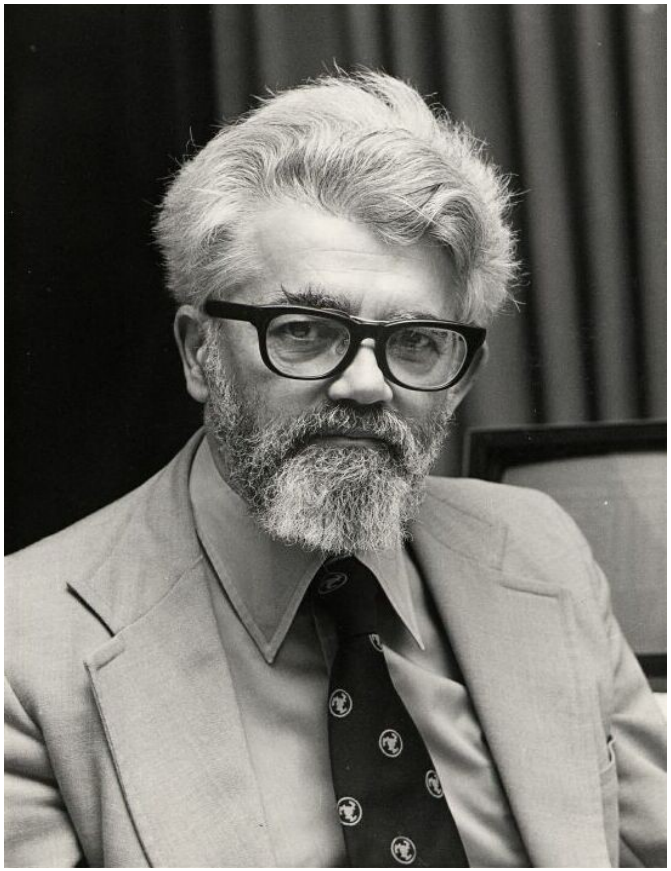
<letter> ::= a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<identifier> ::= <letter> |
<identifier> <letter> |
<identifier> <digit>

LISP

- John McCarthy, b. 1927
  - 1971 Turing Award
    “In the course of its development the LISP system went through several stages of simplification and eventually came to be based on a scheme for representing the partial recursive functions of a certain class of symbolic expressions.”
- Recursive Functions of Symbolic Expressions and their Computation by Machine, Part I, CACM, April 1960.



The roots of modern languages

- Algol 60 - International Algorithmic Language
  - designed by IFIP Working Group 2.1 in 1958-1960
  - earlier versions: IAL, Algol 58
  - John Backus, Peter Naur, John McCarthy, Alan Perlis & others
  - formally specified syntax using Backus-Naur Form (BNF)
  - significant influence on all of today's modern languages
  - introduced explicit variable type declarations, block structure (begin-end), nested lexical scopes & recursive procedures
- Pascal, Modula, Ada, C, C++, & Java are direct descendants of Algol
- Scheme adopted lexical scoping from Algol
- Simula 67 - first object-oriented language
  - designed by Ole-Johan Dahl and Kristen Nygaard
  - influenced all subsequent OO programming languages
  - objects & classes
  - inheritance (subtyping) & virtual methods (subtype polymorphism)

Other important languages

- Algol-like
  - Jovial, Euler, Pascal, Algol-68, Forsythe, Clu, Ada
- Functional
  - ISWIM, FP, SASL, Miranda, Haskell
  - LCF, ML, SML, Caml, OCaml
  - Scheme, Common LISP
- Object-Oriented
  - Smalltalk, Objective-C, C++, Eiffel, Modula-3, Self, C#, CLOS
- Logic programming
  - Prolog, Gödel, LDL, automated theorem provers (ACL2)
- Research-oriented
  - Dylan, ABCL/1, ACT, and literally hundreds more …

Ada

- Primarily used by the US Dept of Defense
  - designed by a French language design team as
    part of an open competition
  - Named after Ada Byron (Lady Lovelace), 1815-1851
    - At a young age, Ada learned of Charles Babbage's ideas for a new calculating engine, the Analytical Engine. Babbage conjectured: what if a calculating engine could not only foresee but could act on that foresight. Ada was impressed by the universality of this idea. She suggested the idea of writing a plan for how this new calculating engine could be used to calculate Bernoulli numbers. This plan, is now regarded as the first "computer program."
    - see the book: Ada, The Enchantress of Numbers, by Betty Alexandra Toole

Application specific languages

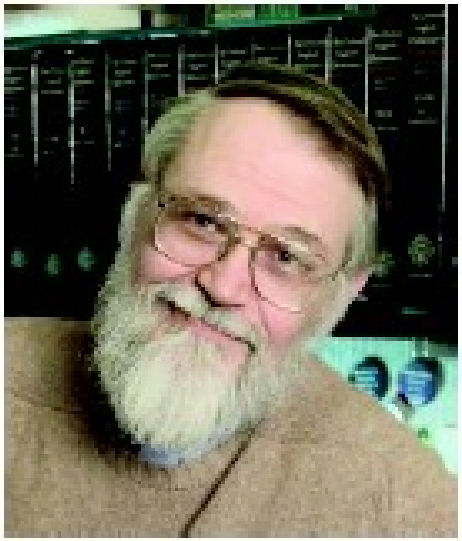
- Commercial data processing & database querying
  - Cobol, SQL, 4GLs, XQuery
- Systems programming
  - PL/I, PL/M, BCPL, BLISS, Modula, Modula-2, Oberon
- Specialized applications
  - BASIC, APL, Forth, Icon, Logo, SNOBOL4, GPSS, VisualBasic
- Concurrent, Parallel & Distributed
  - Concurrent Pascal, Concurrent C, C*, SR, Occam, Erlang, Obliq
- Command shells, scripting & “web” languages
  - sh, csh, tcsh, ksh, zsh, bash, ...
  - Perl, Php, Python, Rexx, Ruby, Tcl, AppleScript, VBScript, etc.
  - HTML/XML are markup languages not programming languages
  - but they often imbed executable scripts like Active Server Pages (ASPs) & Java Server Pages (JSPs)
- Programming tool “mini-languages”
  - awk, make, lex, yacc, autoconf, ...

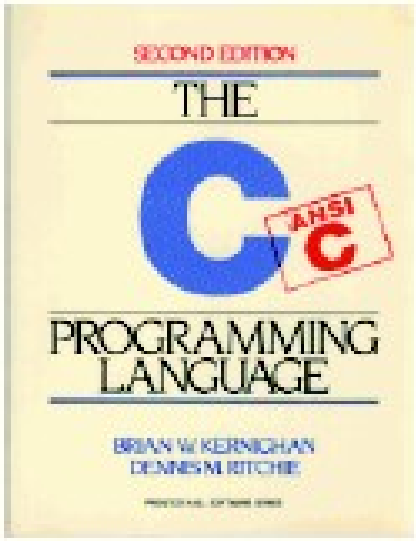
Cobol

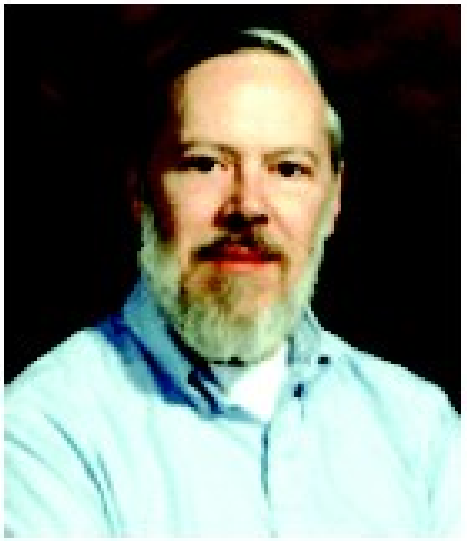
- Common Business Oriented Language
  - invented in the 1950's
  - primarily used for business data processing applications
  - billions spent to fix Y2K issues in old Cobol programs
- Admiral Grace Murray Hopper, 1906-1992
  - PhD Mathematics, Yale, 1934
  - joined the Navy in 1943 and worked at Harvard with Howard Aiken on the Mark I and Mark II computers
    - called the "mother of Cobol" for her contributions to the standardization of the language
    - credited with inventing an early compiler (1952)
    - She did this, she said, because she was lazy and hoped that "the programmer may return to being a mathematician."
    - Conference on Women in Computing is regularly held in her honor "ACM Grace Murray Hopper Award" see http://www.acm.org/awards for winners

From K&R C to ISO/ANSI C

- Brian Kernighan
  - also the ‘K’ in AWK
- Dennis Ritchie
  - 1983 Turing Award winner (with Ken Thompson)

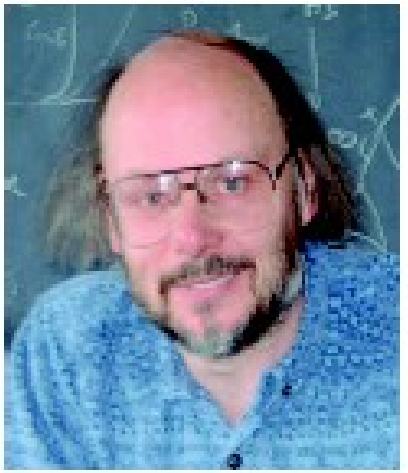


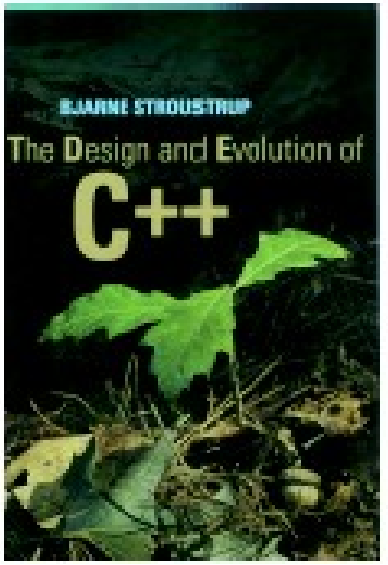




From K&R C to “C with Classes” to C++

- Bjarne Stroustrup
  - Ph.D Univ. of Cambridge
  - used Simula in Ph.D. research and he knew about BCPL
  - then he went to Bell Labs & created “C with classes” in 1979
  - ‘++’ in C++ due to A. Koenig first “Cfront” translator from C++ to C around 1983
  - released to Universities in 1985-86





And then came Java...

- James Gosling (and “Duke”)
  - Gosling Emacs
  - Oak => Java
- Java is more influenced by C (syntax) and Modula-3 (object model) than by C++
  - Unlike C++
    - no operator overloading
    - no templates (but in Java 1.5)
    - no multiple inheritance
  - Like Modula-3
    - explicit interfaces
    - single class inheritance
    - exception handling
    - built-in threading model
    - references & automatic garbage collection (no pointers!)

From BASIC to C# to .NET

- The “un-Java” for Windows
  - an aside: the politics of language adoption
  - use of a programming language to win the mindshare of the software developer community to gain or maintain commercial market share
  - “open” language design/evolution process vs proprietary ownership of a language
  - this is not a new thing: IBM tried to do this with PL/I in the 60s, but free implementations appeared: e.g., PL/C - Cornell PL/I
  - C# has an interesting run-time environment
    - .NET CLR - common language runtime for Visual Basic, C++, C#, and future Microsoft languages

Why so many languages?

- In cosa differiscono tra di loro i linguaggi?
- Cosa hanno in comune?

Alcuni concetti in comune

- Variabile
- Istruzione
  - un comando, una funzione, oppure una regola descrittiva
- Espressione
- Strutture di controllo
  - Per governare il flusso dell'esecuzione del programma,
- Sottoprogramma
  - un blocco di codice che può essere richiamato da qualsiasi altro punto del programma.
- Strutture dati
  - meccanismi che permettono di organizzare e gestire dati complessi.

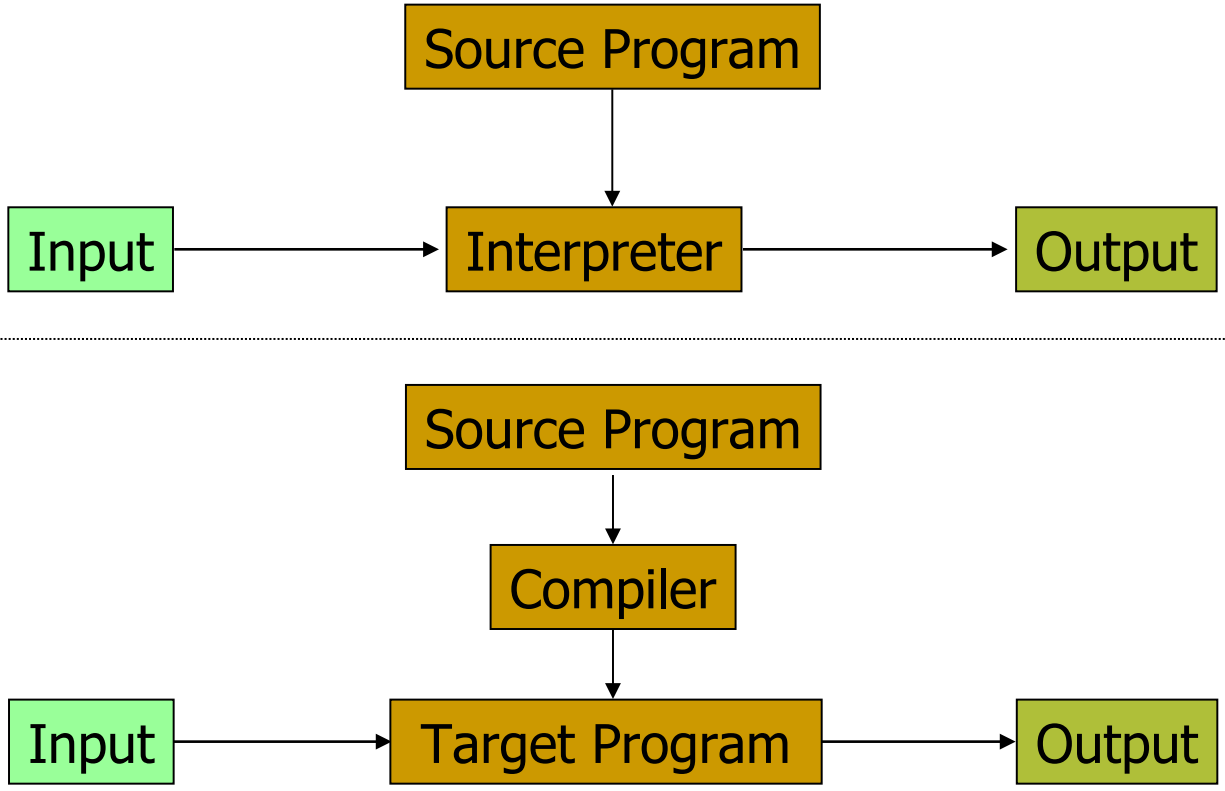
In comune – da approfondire

- Da approfondire invece abbiamo:
- Sintassi
  - Compilatore, ... --> Corso di linguaggi
  - Se volete, vedete il libro 4.1
  - Dovete sapere cosa fa un compilatore, interprete, ecc.
- Tipi --> M1

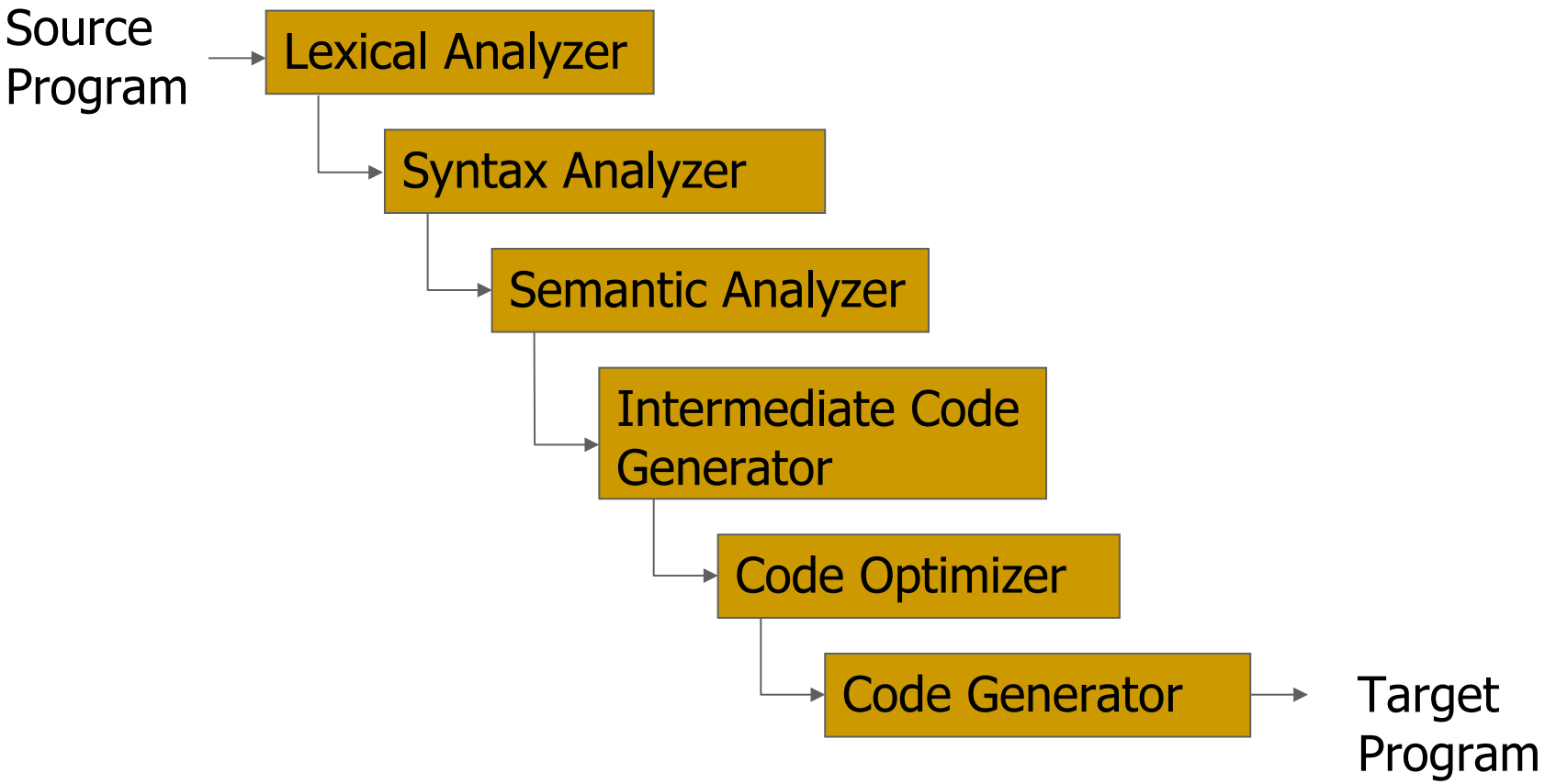
Syntax and Semantics of Programs

- Syntax
  - The symbols used to write a program
- Semantics
  - The actions that occur when a program is executed
- Programming language implementation
  - Syntax $\rightarrow$ Semantics
  - Transform program syntax into machine instructions that can be executed to cause the correct sequence of actions to occur

Interpreter vs Compiler



Typical Compiler



See summary in course text, compiler books

Syntax and correctness

- Cosa vuol dire che un programma è corretto?
  - (ing. del sw): soddisfa i requisiti
  - (noi) un po' meno: termina, non crash, accessi memoria corretti ...
- Not all the programs that are syntactically correct are correct
  - Correttezza di sintassi
    - Analisi statica
  - Con l'analisi statica posso trovare tutti gli errori nel mio programma?

Caso del C

```
[cairngorm:scrap] cat > crash.cc
void main() {
    ((int *) 5)[0] = 6;
}

[cairngorm:scrap] g++ crash.cc -o crash

[cairngorm:scrap]./crash
Segmentation fault
```

Java

```
[cairngorm:scrap] cat > Crash.java
class Crash {
    static public void main(String[] args) {
        ((int[]) 5)[0] = 6;
    }
}

[cairngorm:scrap] javac Crash.java
Crash.java:3: Invalid cast from int to int[].
        ((int[]) 5)[0] = 6;
         ^
```

Per cosa differiscono?

- Analisi
  - Analisi statica / analisi dinamica
    - Statica: analisi sul codice senza eseguirlo
    - Dinamica: durante l'esecuzione
- Espressività vs efficienza
- Le categorie sono:
  - Procedurali
    - Pascal, C, ...
  - Object oriented
    - Java, C#, ...
  - Funzionali
    - Lisp, ...
  - Logici
    - Prolog, ...

Linguaggi funzionali

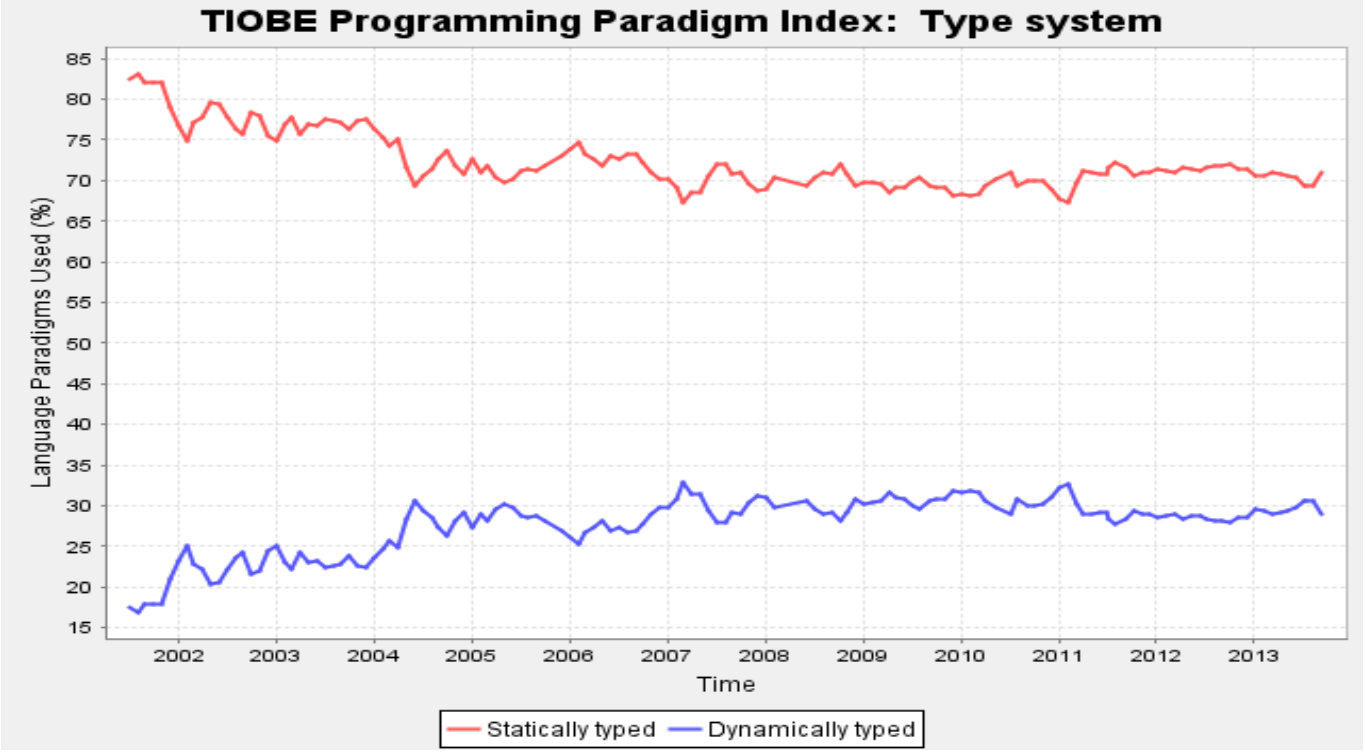
- Sez 4.4. del libro
- In molti programmi la base sono le istruzioni imperative:
  - Es: x := 5
- Esistono anche dichiarazioni e definizioni ...
  - Es: f(int x) { return 5}
- Si pyò esprimere un programma con solo dichiarazioni di funzioni?
  - Funzione: in stile matamatico, dati certi input calcola un certo output
  - es. f(int x) = 5,   f(int x) = x *2 + g(x-3)
  - Sì --> LISP

Quali famiglie sono più diffuse?

Category Ratings	Sept 2012	Delta Sept 2011
Object-Oriented Languages	56.0%	-1,1
Procedural Languages	37.3%	-0,9
Functional Languages	3.8%	+0,6
Logical Languages	3.0%	+1,3

Type system

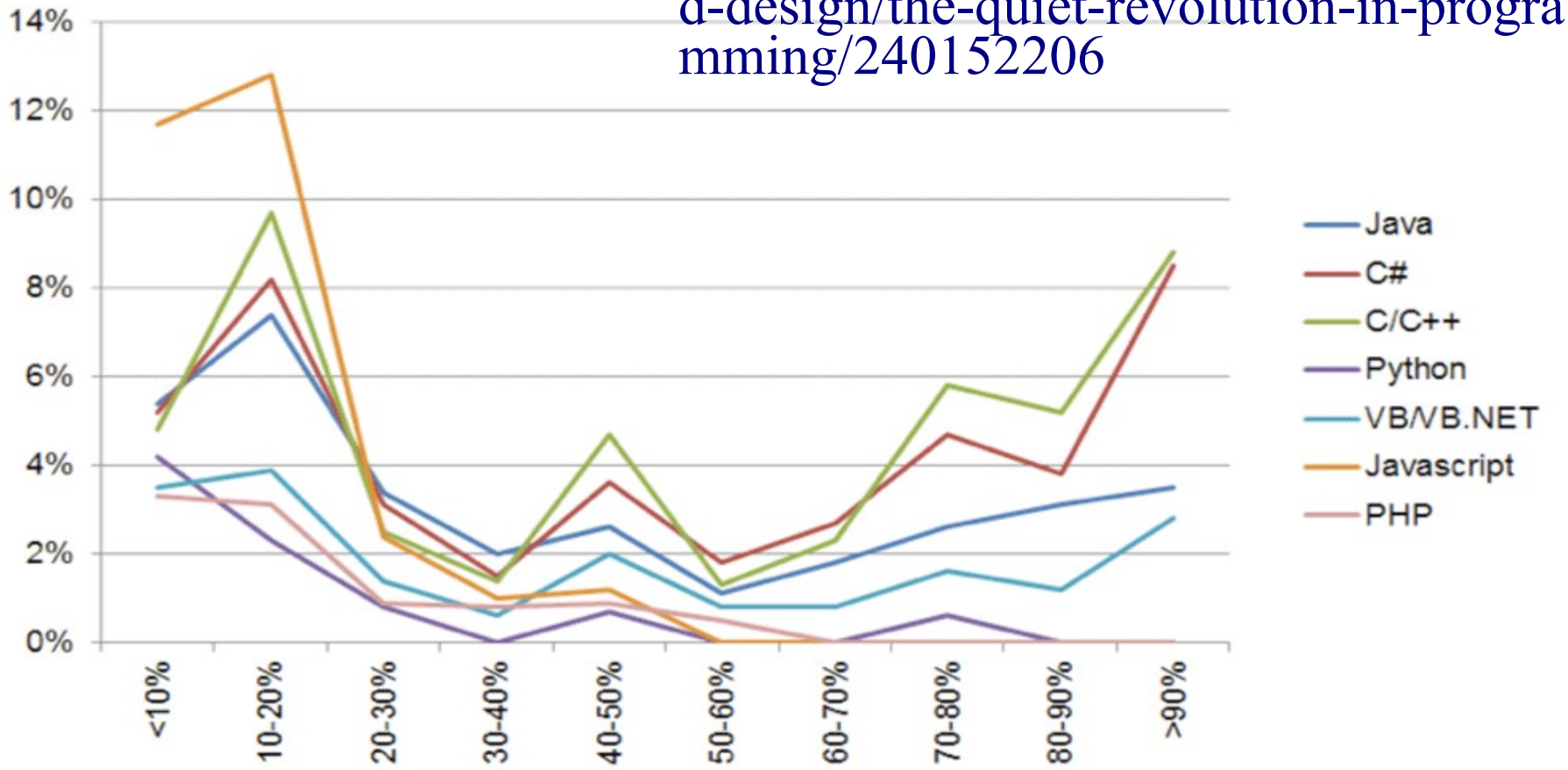
Statically Typed Languages	71.0%
Dynamically Typed Languages	29,0%



Dr.Dobb's
THE WORLD OF SOFTWARE DEVELOPMENT

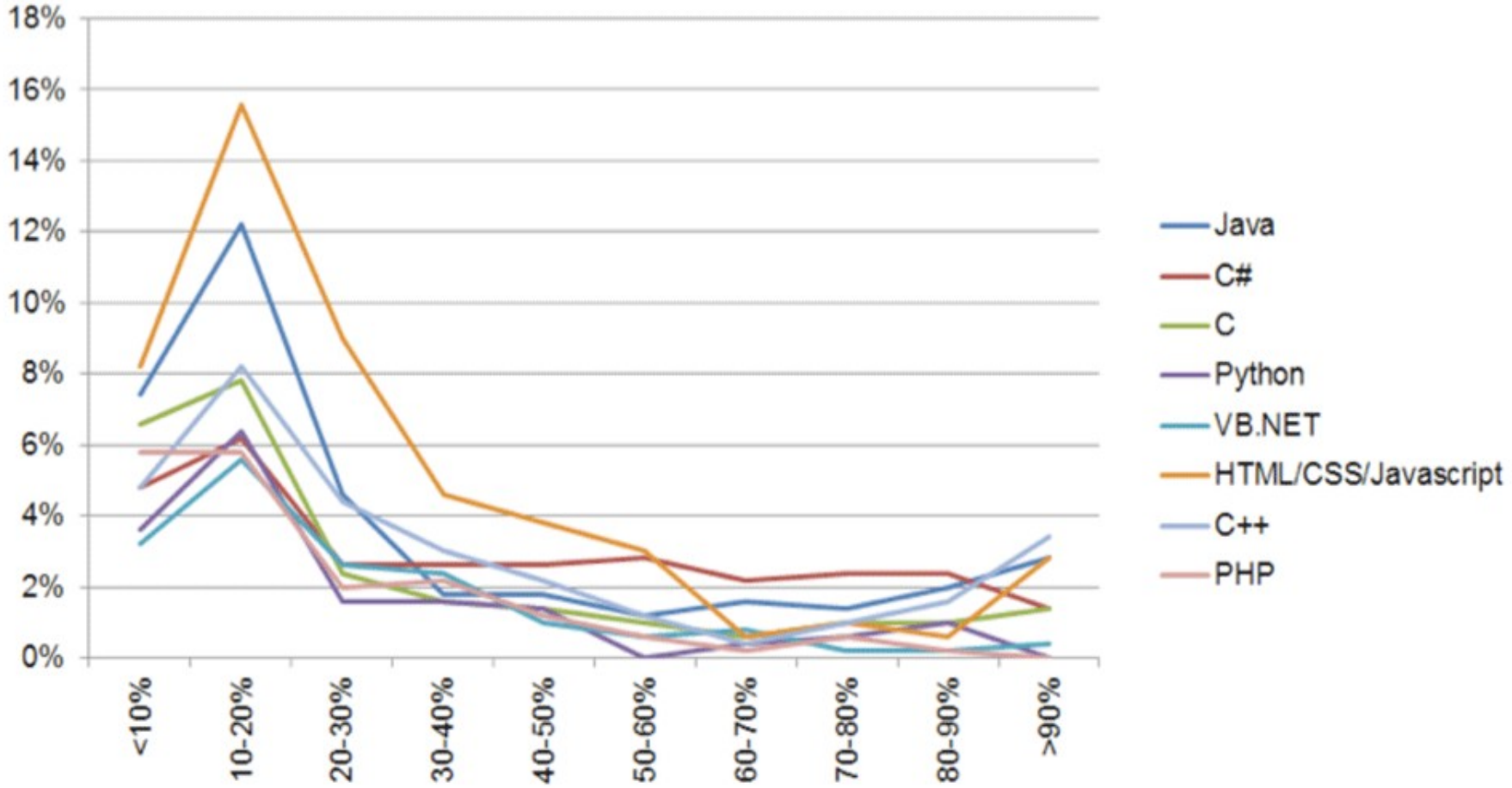
The Quiet Revolution in Programming

http://www.drdobbs.com/architecture-and-design/the-quiet-revolution-in-programming/240152206



Fraction of programmers (y-axis) who spend x amount of time coding in a given language in 2010.

The Quiet Revolution in Programming



Fraction of programmers (y-axis) who spend x amount of time coding in a given language in 2012.

- Multiple programming paradigms are changing simultaneously: the ubiquity of mobile apps; the enormous rise of HTML and JavaScript front-ends; and the advent of big data.
-
- it's now more difficult to find programmer talent that satisfies all the needs of a project; and it's more difficult as a programmer to be deeply fluent in all the necessary languages and idioms.

DZone / Java Zone

Over a million developers have joined DZone

REFCARDZ GUIDES ZONES JOBS | AGILE AI BIG DATA CLOUD DATABASE DEVOPS INTEGRATION IOT JAVA MOBILE PERFORMAN

Bounty Hunters Wanted: Win up to $100 Claim Your Writing Prompt

The Future: Polyglot Programmers

Let's be

by Dave Fecak MVB · Aug. 29, 12 · Java Zone

Like (0) Comment (10) Save Tweet 11.19k Views

Join the DZone community and get the full member experience. JOIN FOR FREE

Learn how to troubleshoot and diagnose some of the most common performance issues in Java today. Brought to you in partnership with AppDynamics.

I wrote an article three years ago, "Become a Better Java Programmer – Learn Something Else", that was subsequently picked up by JavaWorld and DZone and received moderate positive and a bit of negative feedback. The negative was primarily aimed at my assessment that "perhaps 80%" of the best local Java talent were familiar with at least one of a set of other languages I listed. Some readers felt the 80% was pulled from thin air (or pulled perhaps from somewhere else), and I concede the number was simply an estimate. And of course the "best local Java talent" line was also subjective, but time has shown I am a good judge of that.

Polyglot programming