

Sommario: 26/2/2020

- Funzioni Calcolabili, Programmi, Algoritmi (ancora 2 slides)
- LP: Perché tanti Linguaggi?
- LP: Cosa significa studiare i Linguaggi e Perché farlo?
- LP: Classificazione e Spettro Applicazioni dei Linguaggi
- Esercizi

Gli **Strumenti Formali, Fondamentali** che introducono i Programmi per:

- **Esprimere Tutte** le *Computer Applications* ...
- **Definire Tutte** le *Funzioni Calcolabili*, \mathcal{F}
- **Implementare** gli *Algoritmi* che sono:
 - **Procedimenti Effettivi**, ovvero ([Knuth 1968 - Stone 72]¹):
 - **Finitezza** Descritto in modo finito mediante
 - **Definitezza** *passi* rigorosamente e non ambigualmente definiti che
 - **Effettività** usano *operazioni elementari* che devono essere riproducibili con un *agente di calcolo*² e che
 - **Input/Output** si applicano a dati/informazione di dimensione finita ma non necessariamente limitata.

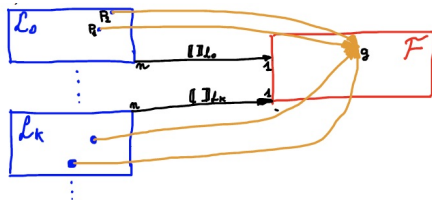
¹https://en.wikipedia.org/wiki/Algorithm_characterizations#1967_Rogers.27_characterization

²che usa energia/tempo finito

Gli **strumenti fondamentali** per:

- **Esprimere Tutte** le *Computer Applications* ...
- **Definire Tutte** le *Funzioni Calcolabili*, \mathcal{F}
- **Implementare** gli *Algoritmi* che permettono di:
 - Studiare proprietà delle applicazioni (*problemi*) o delle stesse funzioni che le calcolano, quali:
 - *Costo* dei diversi procedimenti per una stessa applicazione
 - *Bounds* e *Trattabilità* di funzioni di \mathcal{F} , ad esempio:
 - $g \in \mathcal{F}$ non può essere calcolato in meno di un Costo C_g
 - $g \in \mathcal{F}$ può essere calcolato in un Costo inferiore a C_g
 - $g \in \mathcal{F}$ è calcolabile in Costo esponenziale con l'input
 - Senza far riferimento ad alcun specifico Linguaggio di Programmazione.

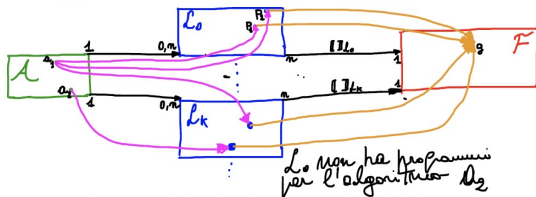
- Un Linguaggio di Programmazione è un
 - **Formalismo**
 - Sintassi* per la forma dei termini del linguaggio
 - Semantica* per il significato da associare a ciascun termine
 - per definire **Programmi**
 - sono la forma principale dei termini esprimibili del linguaggio
 - definiscono tutte e solo le funzioni calcolabili
 - implementano un **algoritmo**
 - dotato di un **esecutore** dei suoi programmi (Macchina Astratta)



Notazione. Le frecce sono relazioni e gli indici alle estremità indicano molteplicità, n , unicità, 1 , assenza, 0 , (o combinazioni: $0/n$, $0/1\dots$), di elementi nell'insieme considerato (sinistra, out, o destra, in).

Relazioni tra Algoritmi, \mathcal{A} , Linguaggi, \mathcal{LP} , Funzioni, \mathcal{F}

- Una **coppia di relazioni** ed una loro **composizione**:



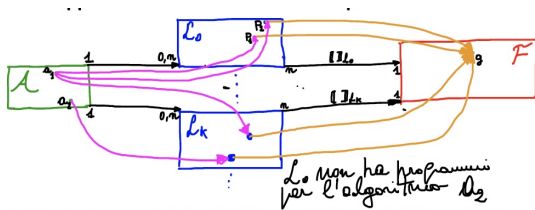
Notazione. Le frecce sono relazioni e gli indici alle estremità indicano molteplicità, n , unicità, 1 , assenza, 0 , (o combinazioni: $0/n$, $0/1\dots$), di elementi nell'insieme considerato (sinistra, out, o destra, in).

- Esistono Funzioni Calcolabili di cui non si conoscono, ad oggi, algoritmi?
vedi Esercizio 3.
- Esistono algoritmi di cui non si hanno, ad oggi, programmi (in un qualche $\mathcal{L} \in \mathcal{LP}$) che li rendano *processi automatici*?
vedi Esercizio 4.
- Esistono Funzioni di cui non si conosce, ad oggi, la calcolabilità?
vedi Esercizio 5.

Rispondiamo in 1 slide

- LP sono **equipotenti**
 - **Esprimono tutto e solo** \mathcal{F} , ovvero hanno programmi per tutte e sole le funzioni $g \in \mathcal{F}$
- LP **non hanno la stessa espressività**
 - **Diversi nel modo di** esprimere la *Struttura dei Programmi*
 - **Esistono** algoritmi con cui $g \in \mathcal{F}$ può essere calcolata che sono impraticabili in un dato linguaggio.
 - Mancanza di costrutti adeguati:
 - Necessità di emulare strutture presenti in altri linguaggi:

- LP sono **equipotenti**
- LP **non hanno la stessa espressività**
 - **Mancanza di costrutti adeguati:**
 - Esempio. Ricorsione per algoritmi induttivi (quicksort, ...)
 - Esempio. QuBit state in algoritmi quantistici.
 - Esempio. Molecular Annealing per algoritmi biologici ³
 - **Necessità di emulare strutture presenti in altri LP:**
 - Esempio. memoria dinamica per algoritmi su stack
 - Emulazione Applicabile:
 - Al tempo di Annealing e replicazione di una stringa di DNA?
 - Allo stato di un qubit di una struttura quantica (o alla precisione del suo trasferimento)?



³(Hamiltonian Path, 3-Colorability... in Jonoska N., C.C.Seeman, Computing by Molecular Self-Assembly, Interface Focus (2012) 2, doi:10.1098/2011.0117

Rispondiamo in 2 slides e
un corso di 9 crediti

Per conoscere in cosa e quanto sono diversi ed essere consapevoli di vantaggi e limiti del linguaggio che si sta usando.

- **Espressività.** Perché esprimono funzioni calcolabili in modo anche, molto diverso
- **Sviluppo** Permettono di realizzare Computer Applications con caratteristiche diverse per
 - risorse necessarie per la realizzazione
 - prestazioni del sistema realizzato (facilità di uso, tempi e risorse richieste dall'uso...)
 - manutenzione (correzione errori, interventi di modifica e ri-adequamento...)
- **Metodologie** Supportano *Metodologie di Programmazione* diverse e/o usabili in modo maggiore o minore
- **Acquisire Competenza** (sintassi, semantica, implementazione, uso e metodologie supportare)

- **Acquire Competenza** (sintassi, semantica, implementazione, uso e metodologie supportate)
 - Nella programmazione e nell'uso anche di linguaggi già noti ed usati. Attraverso:
 - conoscenza dell'effettiva implementazione delle strutture usate
 - confronto sul modo di utilizzare strutture e meccanismi simili in linguaggi diversi
 - esperienza nell'analisi di caratteristiche oscure o difficili da usare in un linguaggio
 - stesura in linguaggi diversi di programmi e/o problemi noti
 - ...
 - Nella scelta del Linguaggio più appropriato per ogni specifica Applicazione da sviluppare
 - Per *classificare* e apprendere gli ultimi Linguaggi di Programmazione definiti

Una prima Classificazione: Prescriptive vs. Descriptive

- **Prescriptive = Come il calcolo deve procedere**
Imperative Languages: State + Mutable Value +
Assignment + Sequence Control

- **Descriptive = Cosa il calcolo deve produrre**
Declarative Languages: Immutable Value +
Application + Composition

Una classificazione più fine

I principali:

- **Procedural:** Fortran, Cobol, Algol, Pascal, C, ADA, ...
- **Functional:** Lisp, Scheme, ML, Haskell, OCAML, ...
- **Algebraic:** Lucid, OBJ, OPAL, ActOne
- **Logic, Constraint-based:** Prolog, LogLisp, Datalog, Parlog (SQL, spreadsheets languages,)...
- **Object Oriented:** Simula67, SmallTalk, C++, OCAML, Java, C#, Scala, F#, ...
- **Scripting:** Perl, Python, PHP, JavaScript...
- **Concurrent:** Lucid, OCCAM, C-Linda, PrologLinda, SPARK, Parlog, Java, C#, ...
- **Dataflow:** Lucid, C-Linda, PrologLinda,...
- **Multi-paradigms:** (i più recenti) F#, Ruby, ...

50 anni di Linguaggi di Programmazione 1955-2015 ⁴

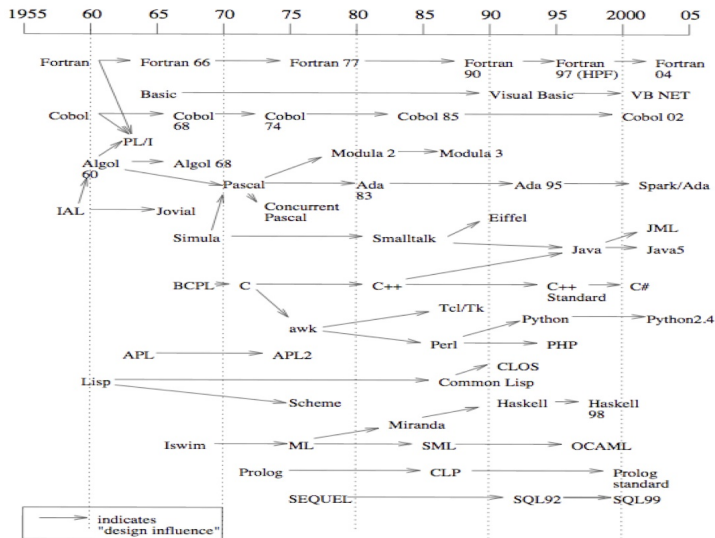
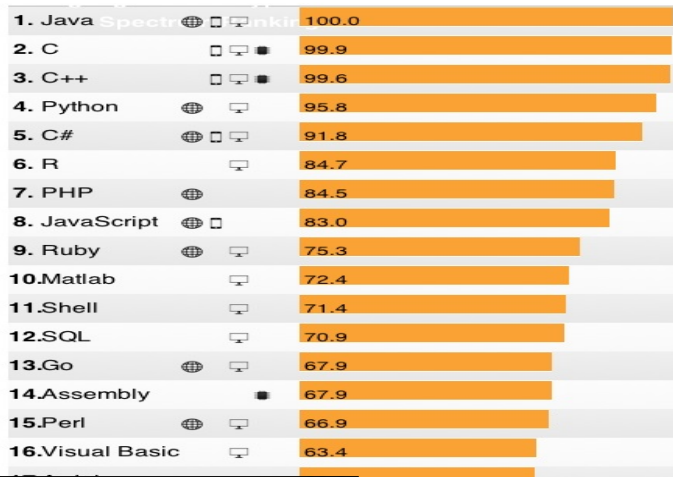


Figure 1.2: A Snapshot of Programming Language History

⁴ fonte ref....

I Linguaggi più usati nel 2015 ⁵ (variazioni, ↑, ↓, 2016) e Spettro di Applicazioni

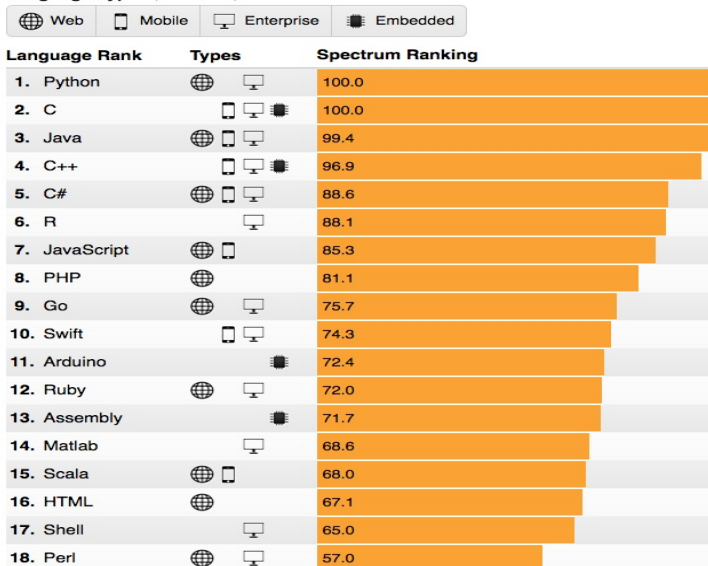
Web Mobile Enterprise Embedded



⁵

fonte IEEE: <http://spectrum.ieee.org/static/interactive-the-top-programming-languages-2015>

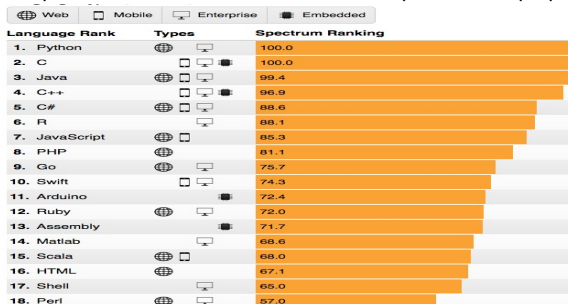
I Linguaggi più usati nel 2017 e Spettro Applicazioni⁶



⁶ fonte IEEE: <http://spectrum.ieee.org/static/interactive-the-top-programming-languages-2017>

I Linguaggi più usati nel 2015-2017 e Spettro Applicazioni

- L'elenco contiene i Linguaggi più utilizzati nello sviluppo di sw, nell'ultimo anno.
- Non tutti i citati sono Linguaggi di Programmazione:
 - Shell è il linguaggio di interazione con un sistema operativo;
 - SQL è il linguaggio per la definizione e la gestione di Basi di Dati Relazionali;
 - HTML è il linguaggio per la definizione di pagine WEB
- Per tutti questi, la relazione con \mathcal{F} non vale, al pari di altre proprietà degli LP



- Questi Linguaggi sono sempre più utilizzati nelle Computer Applications
- Nell'ultimo anno, Applicazioni WEB e Mobili sono state le più sviluppate

- 1 Quali sono le 4 caratteristiche di un procedimento perchè sia effettivo?
- 2 Si descriva l'algorithm del quicksort utilizzato in Esercizio L1.1 e si mostri che esso definisce un procedimento effettivo.
- 3 La funzione π_1 , descritta sotto, è calcolabile.

$$\pi_1(n) = \begin{cases} 1 & \text{se l'espansione decimale di } \pi \text{ contiene } n \text{ 1 in sequenza} \\ 0 & \text{altrimenti} \end{cases}$$

- (a) Sapreste fornire una classe di programmi contenente **un** programma per π_1 ?
 - (b) Sapreste giustificare l'affermata calcolabilità di π_1 ?
- 4 Si esamini la definizione data alle pagine 509-510 nell'articolo sulla 3-Colorabilità di N. Jonoska et al., citato a lezione e si risponda:
 - (a) Perchè il procedimento in 4 passi, in pag. 510, definisce un algoritmo?
 - (b) Sapreste fornire un programma in C che implementi tale algoritmo?
 - (c) Sapreste fornire un programma in C che introduca una rappresentazione dei flexible tiles ed emuli le operazioni anneal, ligate, cleave, extract ?
 - (d) Discutere la relazione tra la soluzione data in (b) e quella data in (c).
 - 5 Sia π , la funzione così definita:
$$\pi(n) = \begin{cases} 1 & \text{se l'espansione decimale di } \pi \text{ contiene } n \\ 0 & \text{altrimenti} \end{cases}$$
 - (a) Sapreste fornire una classe di programmi nei quali è certamente contenuto anche **un** programma che calcola π ?
 - (b) Sapreste fornire argomenti per affermare, o per negare la calcolabilità di π ?