Citazioni dalla Babele di Martini.

*“a good programming language is a conceptual*

*universe for thinking about programming"*

Ogni linguaggio è (innanzitutto) uno strumento per pensare.

I linguaggi di programmazione sono strumenti concettuali per pensare la programmazione come attività di problem solving che prevede di usare (interagire e comunicare con) il computer.

*“a language that doesn’t affect the way you think*

*about programming, is not worth knowing”*

Nell’ambito dell’informatica, questo concetto ha generato e guidato nel tempo tutta la produzione dei linguaggi di programmazione da quelli adatti per la costruzione dei computer a quelli di più alto livello adatti per il problem solving; nel campo più ampio della attività scientifica, questo stesso concetto ha generato (come evoluzione del linguaggio naturale) i gerghi per pensare (nelle attività di ricerca) e comunicare (nei processi educativi) le diverse discipline. Una disciplina senza un proprio gergo non può essere una disciplina scientifica, vedi la filosofia!

*“there will always be things we wish to say in*

*our programs that in all known languages*

*can only be said poorly”.*

Questa tesi lascia aperta la possibilità di trovare linguaggi sempre più specifici e adatti per il problem solving del futuro. Parafrasando Hilbert, questo è uno dei problemi che garantisce vitalità all’informatica!

[Vannevar Bush](http://en.wikipedia.org/wiki/Vannevar_Bush) (1890-1974), famoso per il proprio trattato **As We May Think** in cui ipotizza una macchina automatica chiamata Memex

Quando si parla di intelligenza artificiale e lavoro, l'interrogativo è sempre quello: come si salvano i lavori dall'innovazione della Ia? La società di consulenza Accenture ha provato a ribaltare la prospettiva, con una ricerca dove si registrano le figure professionali del tutto inedite che potrebbero emergere grazie all'espansione di modelli commerciali di artificial intelligence. **Le macrocategorie che emergono sono tre: trainers, explainers, sustainers**. Il significato? I trainers si occupano di “formare” gli algoritmi abilitati dall'intelligenza artificiale per renderli meno asettici e più utili, ad esempio istruendo i chatbot ad esprimersi con un linguaggio più empatico. Gli explainers devono, appunto, spiegare portata e valenza delle innovazioni spinte dalle tecnologie di settore: è il caso di coach chiamati dall'aziende per descrivere in maniera semplice applicazioni e conseguenze dell'uso dell'Ia in un'impresa tradizionale. Infine i sustainers, gli esperti della sostenibilità. Il loro compito è tra i più delicati: valutare gli impatti positivi e negativi delle tecnologie impiegate, dal rapporto costi-benefici ai dilemmi etici che si possono creare con una

L’informatica ha una preistoria caratterizzata dalla esigenza di risolvere problemi ed è costellata dalla invenzione di strumenti operativi e concettuali che facilitano le elaborazioni di simboli coinvolte nei processi risolutivi.

La storia ha inizio con la proposta formulata da Alan Turing di un linguaggio (prototipo dei linguaggi di programmazione) che consente di descrivere qualsiasi algoritmo e con quella avanzata da John von Neumann di una macchina (il computer) capace di eseguire tutte le manipolazioni di simboli descritte (tutti gli ordini impartiti o tutte le istruzioni trasmesse) con un linguaggio di programmazione.

Ma alcuni linguaggi sono più adatti di altri a determinati scopi. Alcuni sono più evocativi, altri più sintetici, altri ancora più astratti. Tra gli aforismi di Alan Perlis troviamo:

“a good programming language is a conceptual universe for thinking about programming"

Che ricorda in modo sorprendente il «Parlando, creiamo mondi» di Steiner.

E ancora:

“a language that doesn’t affect the way you think about programming, is not worth knowing”

e

“there will always be things we wish to say in our programs that in all known languages can only be said poorly”.