ESERCZI

1. Scrivere un programma C che calcoli la funzione *ordinamento di sequenza* implementando l'algoritmo di QuickSort. Allo scopo completare il testo esponendo eventuali vincoli che si ritenga utile assumere.
2. Se e dove, nella formulazione $\mathcal{F}$, vediamo che $\mathcal{F}$ è numerabile.
3. Se e dove, nella formulazione $\mathcal{F}$, vediamo che la funzione $[|\_|]$ è calcolabile. E se si, cosa calcola.
4. Se e dove, nella formulazione $\mathcal{F}$, vediamo che $\mathcal{P}$ contiene un programma che calcola $[|\_|]$.
5. Se e quale relazione, nella formulazione $\mathcal{F}$, vediamo tra l'iniettiva $^{-} : \mathcal{P} \to \mathcal{D}$ e la suriettiva $[|\_|]$.
6. Se e dove, nella formulazione $\mathcal{F}$, vediamo che per ogni funzione calcolabile c'è un programma che la calcola.
7. Se e dove, nella formulazione $\mathcal{F}$, vediamo che per ogni programma c'è una funzione calcolabile che è la funzione calcolata dal programma.
8. Se e dove, nella formulazione $\mathcal{F}$, vediamo che $[|\_|]$ stabilisce una corrispondenza uno-uno tra $\mathcal{P}$ e $\mathcal{F}$.
9. Sia $\mathtt{u} \in \mathcal{L}$ un programma di $\mathcal{L}$ che calcola la funzione $[|\_|]$ (vedi anche esercizio 4). Si dica cosa calcola $\mathcal{U}(\mathtt{u})$.
10. Con riferimento alla formulazione $\mathcal{F}$. Sia $\mathcal{U}_{\mathcal{P}} \equiv \{\mathtt{p} \in \mathcal{P} \mid [|\mathtt{p}|] = \mathcal{U}\}$. Si discuta la cardinalità di $\mathcal{U}_{\mathcal{P}}$.

ESERCZI -24/2/2016

Esercizio (1)

Scrivere un programma C che calcoli la funzione ordinamento di sequenza *implementando l'algoritmo di QuickSort. Allo scopo completare il testo esponendo eventuali vincoli che si ritenga utile assumere.*

Soluzione: Requisiti aggiuntivi

```
/*
Esercizo 1-24/2/2016.
Scrivere un programma C che calcoli la funzione "ordinamento di sequenza" implementando l'algoritmo
QuickSort. Allo scopo, si completi il testo con fgli eventuali vincoli che si ritengano necessari.
**
* Marco Bellia
**
Soluzione.
Vincoli.
1) Esistenza bound alla dimensione della sequenza da ordinare. Il bound è trattato come:
        Parametro di Programma
2) Elementi della sequenza di tipo noto. Il tipo è int.
3) Sequenza realizzata come struttura ad accesso diretto: array (ad allocazion) statica.
*/
```

ESERCZI -24/2/2016 - Esercizio 1 - cont'd

```c
#include <stdio.h>
#include <stdlib.h>

int N = 15; // Parametro di programma (non modificabile)
void QuickSort(int Left, int Right, int A[]){// procedura ricorsiva
    int I=Left;  //indici in [Left..Right]
    int J=Right; //indici in [Left..Right]
    int temp=A[(Left+Right)/2];
    while (I<=J) {
        while (A[I]<temp) I++;
        while (A[J]>temp) J--;
        if (I<=J){
            int scambia=A[I];
            A[I]=A[J];
            A[J]=scambia;
            I++; J--;
        }
    }
    if (Left<J) QuickSort(Left,J,A);
    if (I<Right) QuickSort(I,Right,A);
}

int main(void){
    // Allocaione Sequenza da ordinare
    int Seq[N];
    int k;
    //lettura sequenza k<=N interi da ordinare;
    int i=0;
    while (i<N && scanf("%d",&Seq[i])==1) i++;
    k=i-1;
     //ordinamento QuickSort
    QuickSort(0,k,Seq);
    //stampa sequenza ordinata;
    for (int i=0; i<k; i++) printf("%d,",Seq[i]);
    printf("%d\n",Seq[k]);
    return(1);
}
```

ESERCZI -24/2/2016

Esercizio (2)

Se e dove, nella formulazione $\mathcal{F}$, vediamo che $\mathcal{F}$ è numerabile.

- $\mathcal{F} \equiv [\mathcal{D} \to \mathcal{D}]$ *with* $\mathcal{D} \cong [\mathcal{D} \to \mathcal{D}]$
  - *(Universale)* $\exists\, \mathcal{U} \in \mathcal{F}$,
  - *(Mapping)* $\exists\, [|\_|] : \mathcal{P} \to \mathcal{F}$, *suriettiva,*
  - $\forall \mathtt{p} \in \mathcal{P}$,
    $\mathcal{U}(\overline{\mathtt{p}}) = \mathtt{g} \in (\mathcal{D} \cong [\mathcal{D} \to \mathcal{D}] \equiv)\ \mathcal{F}$
  - $\forall \mathtt{g} \in \mathcal{F}$, *sia* $\mathtt{g} = [|\mathtt{p}|]$, *per qualche* $p \in \mathcal{P}$
    $\mathcal{U}(\overline{\mathtt{p}})(\mathtt{x}) = \mathtt{g}(\mathtt{x}) \quad \forall \mathtt{x} \in \mathcal{D}$

  With $\mathcal{L} \in$ **LP**, $\mathcal{P}$ *insieme programmi di* $\mathcal{L}$, $\overline{\ } : \mathcal{P} \to \mathcal{D}$, *iniettiva.*

Soluzione
Immediato da:
-assunzione: *Sia* $\mathcal{D}$ *numerabile*
-proprietà Isomorfismo: $\mathcal{D} \cong ([\mathcal{D} \to \mathcal{D}] \equiv)\mathcal{F}$

per transitività: $\mathcal{F}$ ha cardinalità numerabile.

ESERCZI -24/2/2016

Esercizio (3)

(a) *Se e dove, nella formulazione $\mathcal{F}$, vediamo che la funzione $[|\_|]$ è calcolabile?* (b) *E nel caso, quale funzione calcola?*

- $\mathcal{F} \equiv [\mathcal{D} \to \mathcal{D}]$ *with* $\mathcal{D} \cong [\mathcal{D} \to \mathcal{D}]$
  - *(Universale)* $\exists\, \mathcal{U} \in \mathcal{F}$,
  - *(Mapping)* $\exists\, [|\_|] : \mathcal{P} \to \mathcal{F}$, *suriettiva*,
  - $\forall \mathtt{p} \in \mathcal{P}$,
    $\mathcal{U}(\overline{\mathtt{p}}) = \mathtt{g} \in (\mathcal{D} \cong [\mathcal{D} \to \mathcal{D}] \equiv)\ \mathcal{F}$
  - $\forall \mathtt{g} \in \mathcal{F}$, *sia* $\mathtt{g} = [|\mathtt{p}|]$, *per qualche* $p \in \mathcal{P}$
    $\mathcal{U}(\overline{\mathtt{p}})(\mathtt{x}) = \mathtt{g}(\mathtt{x}) \quad \forall \mathtt{x} \in \mathcal{D}$

With $\mathcal{L} \in$ **LP**, $\mathcal{P}$ *insieme programmi di* $\mathcal{L}$, $\overline{\ } : \mathcal{P} \to \mathcal{D}$, *iniettiva.*

Soluzione (a)
Mapping $[|\_|]$ che soddisfano la formulazione $\mathcal{F}$ non sono unici. Sia $\mathcal{M}_{\mathcal{L}}$ l'insieme di tali mapping per $\mathcal{L}$.
Mostriamo che $\mathcal{M}_{\mathcal{L}} \cap \mathcal{F} \neq \{\}$. Sia $\mathtt{h}$ tale che:

$$\forall \mathtt{p} \in \mathcal{P}, \quad \mathtt{h}(\mathtt{p}) = \mathcal{U}(\overline{\mathtt{p}}).$$

Allora osserviamo che:
(1) $\mathtt{h} \in \mathcal{M}_{\mathcal{L}}$. Infatti soddisfa tutte le condizioni su $[|\_|]$.
(2) $\mathtt{h}$ è in $\mathcal{F}$ quando $\overline{\ } \circ \mathcal{U} \in \mathcal{F}$,
ovvero, quando $\overline{\ } : \mathcal{P} \to \mathcal{D} \in \mathcal{F}$ [5]
(3) Anche per $\overline{\ }$ possiamo trovare $\overline{\ } \in \mathcal{F}$ sebbene non tutte e invettive lo siano.

Soluzione (b)
Calcola la Funzione $\overline{\ } \circ \mathcal{U} \in \mathcal{F}$

[5] $\circ$ è la composizione in $\mathcal{F}$ ed è bf ovviamente in $\mathcal{F}$ (come si vede quando componiamo, a formare un unico programma, il codice, output su input, di 2 distinti programmi)

ESERCZI -24/2/2016

Esercizio (4)

Se e dove, nella formulazione $\mathcal{F}$, vediamo che $\mathcal{P}$ contiene un programma che calcola $[|\_|]$.

- $\mathcal{F} \equiv [\mathcal{D} \to \mathcal{D}]$ *with* $\mathcal{D} \cong [\mathcal{D} \to \mathcal{D}]$
  - *(Universale)* $\exists\, \mathcal{U} \in \mathcal{F}$,
  - *(Mapping)* $\exists\, [|\_|] : \mathcal{P} \to \mathcal{F}$, *suriettiva*,
  - $\forall \mathtt{p} \in \mathcal{P}$,
    $\mathcal{U}(\overline{\mathtt{p}}) = \mathtt{g} \in (\mathcal{D} \cong [\mathcal{D} \to \mathcal{D}] \equiv)\ \mathcal{F}$
  - $\forall \mathtt{g} \in \mathcal{F}$, *sia* $\mathtt{g} = [|\mathtt{p}|]$, *per qualche* $p \in \mathcal{P}$
    $\mathcal{U}(\overline{\mathtt{p}})(\mathtt{x}) = \mathtt{g}(\mathtt{x}) \quad \forall \mathtt{x} \in \mathcal{D}$

With $\mathcal{L} \in$ **LP**, $\mathcal{P}$ *insieme programmi di* $\mathcal{L}$, $^{-} : \mathcal{P} \to \mathcal{D}$, *iniettiva.*

Soluzione

Da: $[|\_|] : \mathcal{P} \to \mathcal{F}$, suriettiva,

Da: $[|\_|] = {}^{-} \circ \mathcal{U} \in \mathcal{F}$ (vedi esercizio 3)

Segue: $\exists \mathcal{L}_{\mathtt{sem}} \in \mathcal{P} : [|\mathcal{L}_{\mathtt{sem}}|] = {}^{-} \circ \mathcal{U} \in \mathcal{F}$

Linguaggi di Programmazione: Cosa sono? /4

Gli **strumenti fondamentali** per:

- **Esprimere Tutte** le *Computer Applications* che sono: ...
- **Definire Tutte** le *Funzioni Calcolabili*, $\mathcal{F}$. ...
- **Implementare** gli *Algoritmi* che sono:
  - **Procedimenti Effettivi**, ovvero ([Knuth 1968 - Stone 72][6]):
    - **Finitezza** Descritto in modo finito mediante
    - **Definitezza** *passi* rigorosamente e non ambiguamente definiti che
    - **Effettività** usano *operazioni elementari* che devono essere riproducibili con un *agente di calcolo* [7] e che
    - **Input/Output** si applicano a dati/informazione di dimensione finita ma non limitata.

[6] https://en.wikipedia.org/wiki/Algorithm_characterizations#1967_Rogers.27_characterization

[7] che usa energia/tempo finito

Linguaggi di Programmazione: Cosa sono? /4bis

Gli **strumenti fondamentali** per:

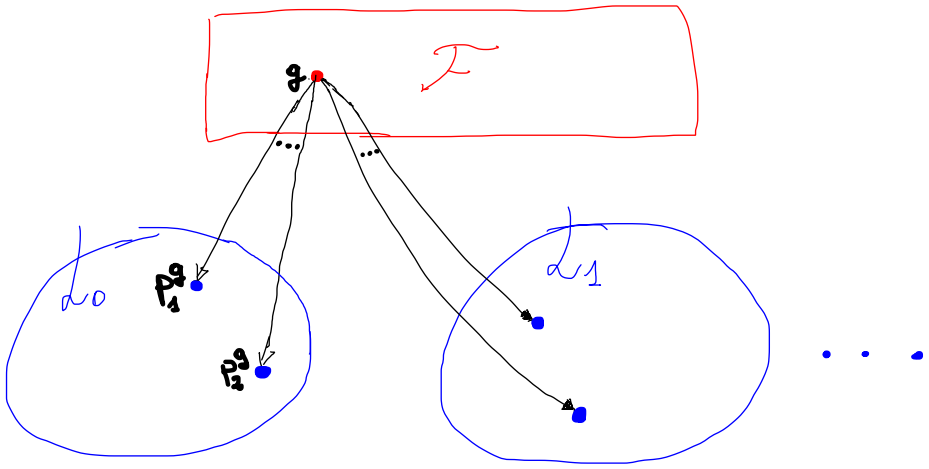
- **Esprimere Tutte** le *Computer Applications* che sono: ...
- **Definire Tutte** le *Funzioni Calcolabili*, $\mathcal{F}$. ...
- **Implementare** gli *Algoritmi* che permettono di:
  - Studiare proprietà delle applicazioni (*problemi*) o delle stesse funzioni che le calcolano, quali:
    - *Costo* dei diversi procedimenti per una stessa applicazione
    - *Bounds* e *Trattabilià* 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.

Linguaggi di Programmazione: Cosa sono? /5

- 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
    - esprimono tutte e solo le funzioni calcolabili
    - implementano un **algoritmo**
  - dotato di un **esecutore** dei suoi programmi (Macchina Astratta)



Linguaggi di Programmazione: Perché più d'uno?

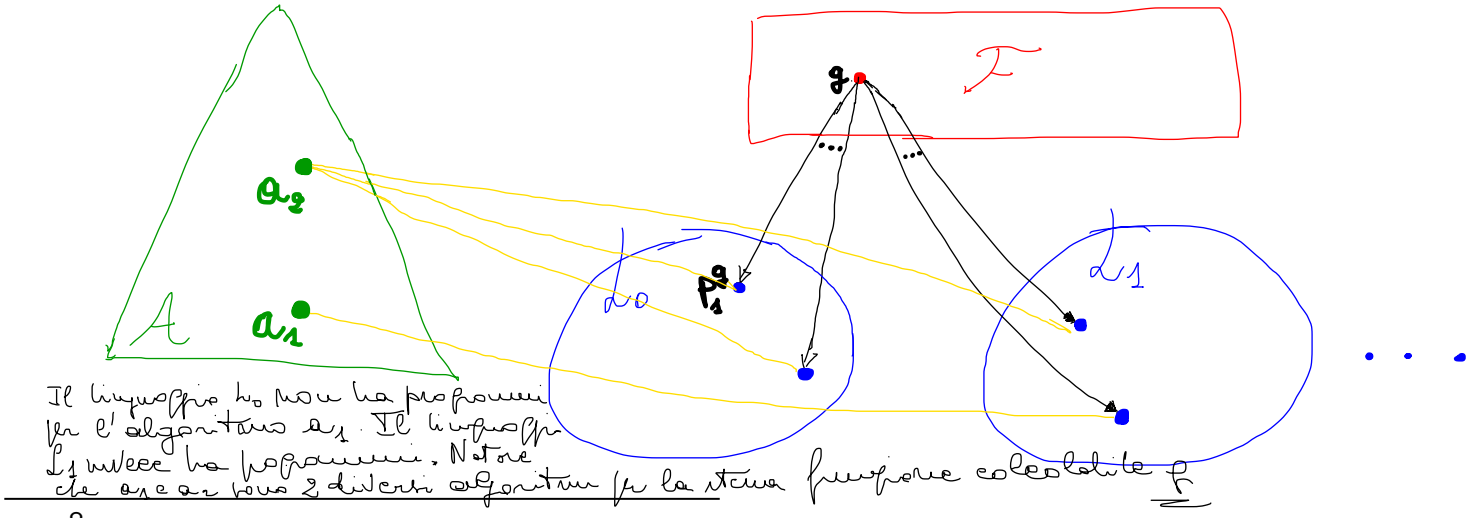
Rispondiamo in 1 slide

Linguaggi di Programmazione: Perché più d'uno?

- 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:

Linguaggi di Programmazione: Perché più d'uno? - cont'd

- LP sono **equipotenti**
- LP **non hanno la stessa espressività**
  - **Mancanza di costrutti** adeguati:
    Esempio. Ricorsione per algoritmi induttivi (quicksort, ...)
    Esempio. Molecular Annealing per algoritmi biologici (Hamiltonian Path, 3Colorability..[8])
  - **Necessità di emulare** strutture presenti in altri linguaggi:
    Esempio. memoria dinamica per algoritmi su stack



[8] Jonoska N., C.C.Seeman, Computing by Molecular Self-Assembly, Interface Focus (2012) 2, doi:10.1098/rsfs.2011.0117

Linguaggi di Programmazione: Perché studiarli? /1

Rispondiamo in 2 slides e.... un corso di 9 crediti

Linguaggi di Programmazione: Perché studiarli? /1

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

- **Espressivitá**. Perchè 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-adeguamento...)
- **Metodologie** Supportano *Metodologie di Programmazione* diverse e/o usabili in modo maggiore o minore
- **Acquisire Competenza** (sintassi, semantica, implementazione, uso e metodologie supportare)

Linguaggi di Programmazione: Perché studiarli? /2

- **Acquisire Competenza** (sintassi, semantica, implementazione, uso e metodologie supportare)
  - 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#, F#, ...
- **Scripting**: Perl, Python, PHP, JavaScript...
- **Concurret**: Lucid, OCCAM, C-Linda, PrologLinda, SPARK, Parlog, Java, C#, ...
- **Dataflow**: Lucid, C-Linda, PrologLinda,...
- **Multi-paradigms**: (i più recenti) FF#, Ruby, ...

50 anni di Linguaggi di Programmazione 1955-2015

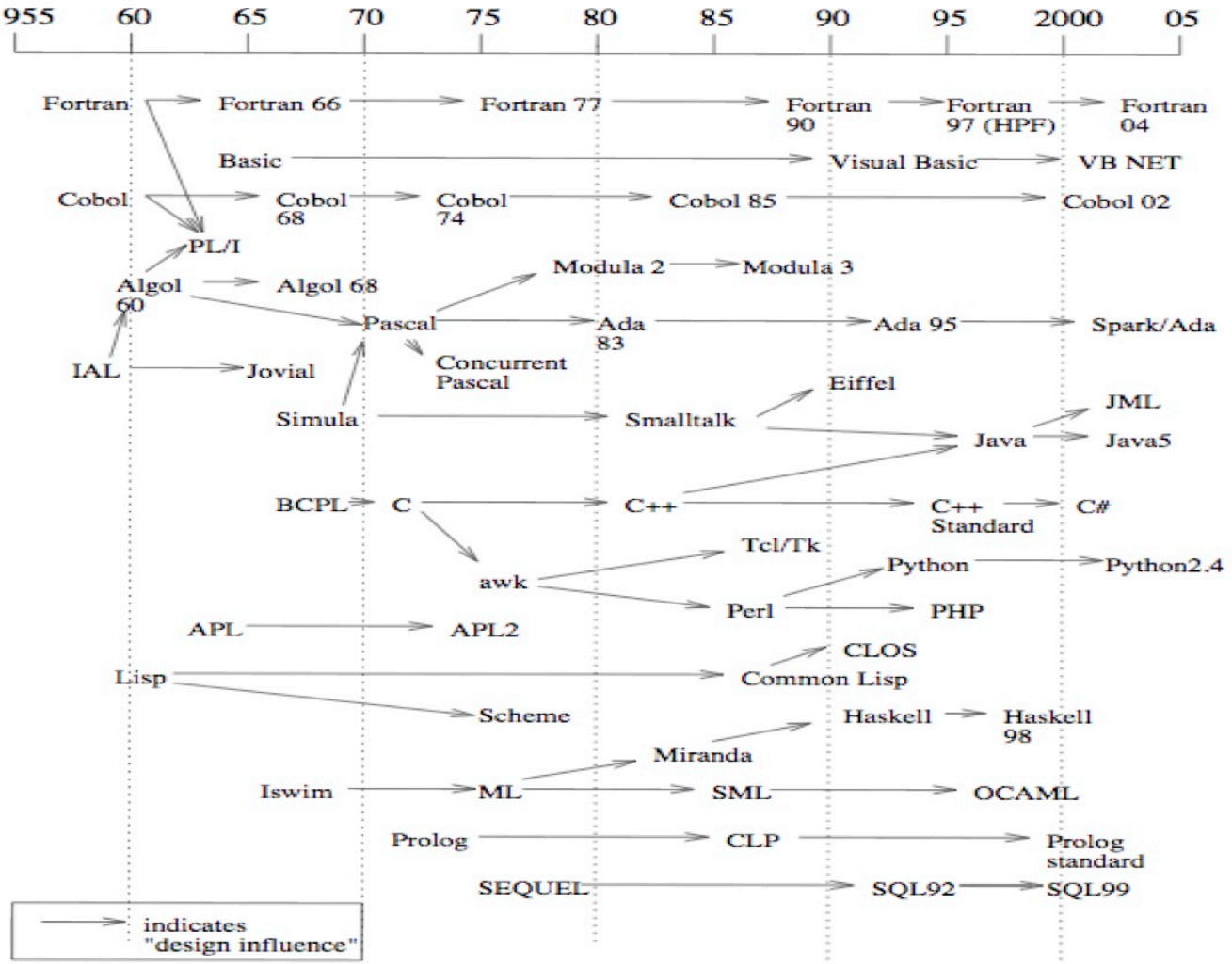
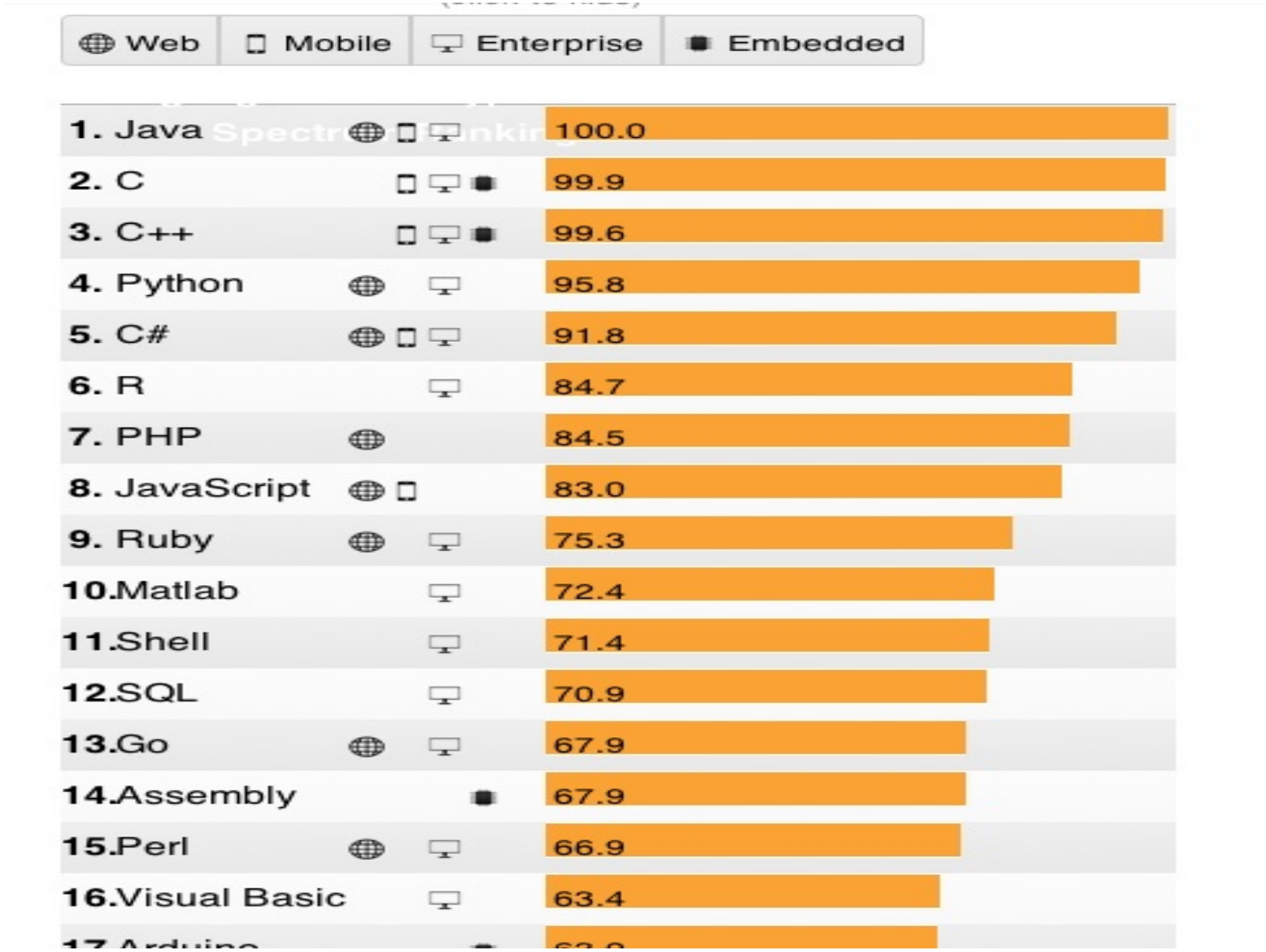


Figure 1.2: A Snapshot of Programming Language History

I Linguaggi pi usati nel 2015 e loro Spettro di Applicazioni[9]



[9] fonte IEEE: http://spectrum.ieee.org/static/interactive-the-top-programming-languages-2015