L'Informatica come scienza autonoma

Appunti per una storia

Simone Martini

Corso di Storia dell'Informatica
Prof. G. Casadei

12 aprile, 2012

Outline

Outline

Complicato...

Presupporrebbe:

- una definizione epistemologicamente precisa;
- l'accesso a fonti primarie.

Solo alcuni appunti di lavoro *per una storia*

La speciazione (dalla zoologia!)

Una specie è rappresentata da quegli individui che incrociandosi tra loro generano potenzialmente una prole illimitatamente feconda.

[Dobzhansky, Mayr]

- Riproduzione attaverso specifici corsi di studi
- Ipotesi di lavoro:
  informatica *qua scienza autonoma* = dipartimenti di CS

- Alcune date certe
- Molti eventi anticipatori precedenti
  "Siamo di una specie diversa"

La speciazione (dalla zoologia!)

Una specie è rappresentata da quegli individui che incrociandosi tra loro generano potenzialmente una prole illimitatamente feconda.
[Dobzhansky, Mayr]

- Riproduzione attaverso specifici corsi di studi
- Ipotesi di lavoro:
  informatica *qua scienza autonoma* = dipartimenti di CS

- Alcune date certe
- Molti eventi anticipatori precedenti
  "Siamo di una specie diversa"

Anni '50: corsi di programmazione

IBM aveva donato 100 calcolatori "gratis", purché venissero tenuti corsi di programmazione.

But these early stages hardly represented computer science as it is understood today, nor did many people regard it as the germ of a genuine discipline worthy of study on a par with other subjects. I myself was a graduate student in mathematics who enjoyed programming as a hobby; I had written two compilers, but I had no idea that I would someday be teaching about data structures and relating all this to mathematics. A few people, like George Forsythe and Alan Perlis and Richard Hamming, had no such mental blocks.

[D. Knuth, 1972; CACM 15(8), 721-727. Forsythe's obituary]

Consapevolezza

Enough is known already of the diverse applications of computing for us to recognize the birth of a coherent body of technique, which I call computer science.
Whether computers are used for engineering design, medical data processing, composing music or other purposes, the structure of computing is much the same....

[G. Forsythe, 1961
Educational implications of the computer revolution. *Applications of Digital Computers, W. F. Freiberger and William Prager (eds.), Ginn, Boston, 1963, pp. 166-178.]*

Resistenze

Most scientists thought that using a computer was simply programming — that it didn't involve any deep scientific thought and that anyone could learn to program. So why have a degree? They thought computers were vocational vs. scientific in nature.

[Conte, Computerworld magazines, 1999]

Computational thinking, ante litteram

The most valuable acquisitions in a scientific or technical education are the general-purpose mental tools which remain serviceable for a lifetime. I rate natural language and mathematics as the most important of these tools, and computer science as a third... The learning of mathematics and computer science together has pedagogical advantages, for the basic concepts of each reinforce the learning of the other.

[G. Forsythe. What to do till the computer scientist comes. Amer. Math. Monthly 75 (1968), 454-462.]

The purpose of computing is insight, not numbers

[Richard Hamming; in Numerical Methods for Scientists and Engineers, McGraw-Hill, 1962]

Qualche data USA e Italia

1962 Purdue University (West Lafayette, IN): primo dpt di CS; Samuel D. Conte (Perlis: 1951-1956@computation center)

1965 Stanford University (Palo Alto, CA); George Forsythe (Herriot, McCarthy, Feigenbaum, Wirth, Knuth(poi)) Dal 1961 era una "division" di Matematica

1965 Carnegie Mellon University (Pittsburg, PA); Alan J. Perlis (Allen, Simon)

1965 Primo PhD *da un Dpt in CS*: Richard Wexelblat @ University of Pennsylvania (ENIAC!)

1971 Yale (New Haven, CT); Perlis

1969 Pisa: Laurea in Scienze dell'Informazione; A. Faedo (Grasselli, Caracciolo, Gerace, Preparata)

1970 Bari

Poi: Salerno e Torino; Udine (1979); . . . ; Milano (1986); . . . ; Bologna (1990); . . .

Outline

Errori, Metodi

- Errori di arrotondamento: Jim Wilkinson (1950s)
- Metodi iterativi
- Costruzione di un *corpus* organico

- Necessità di affrancarsene. . .

Errori, Metodi

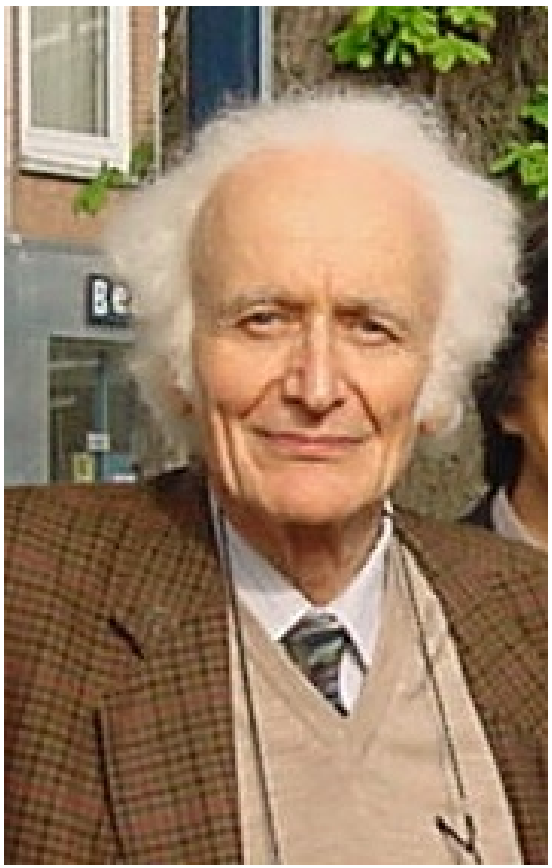
- Errori di arrotondamento: Jim Wilkinson (1950s)
- Metodi iterativi
- Costruzione di un *corpus* organico

- Necessità di affrancarsene...

Outline

Compilatori meta-circolari

Un compilatore per un linguaggio, scritto in quello stesso linguaggio: $\mathcal{C}_{\mathcal{L}\to\mathcal{L}'}$, scritto in $\mathcal{L}$



Corrado Böhm,
Calculatrices digitales:
Du déchiffrage de formules logico-mathématiques par la machine même dans la conception du programme
Ann. di Mat. Pura ed Applicata, vol. 37(4), pp. 175-217, 1954.

Algol 60

- J. Backus, P. Naur, T. Hoare, N. Wirth, J. McCarthy, A. Perlis, A. van Wijngaarden
- Chiara percezione di fare una cosa nuova. Definire:
  - sintassi: Backus-Naur Form (à la Chomsky)
  - semantica: la regola di copia
  - modello di macchina: ambienti locali a stack

LISP

- John McCarthy
- circa 1958, MIT
- ispirato al λ-calcolo
- usa idee di *Information Processing Language*, assembly per list processing (Newell, Shaw, Simon)
- strutture *simboliche*

APL

- Kenneth Iverson (1920 – 2004)
- Linguaggio orientato al calcolo matriciale
- Estremamente terso e sintetico (contra: Cobol!)
  - La somma degli elementi di un vettore: $(+/\omega)$
  - La media degli elementi di un vettore: $(+/\omega) \div \rho\omega$
  - I numeri primi da 1 a R: $(\sim R \in R \circ.\times R)/R \leftarrow 1 \downarrow \iota R$

APL

- Kenneth Iverson (1920 – 2004)
- Linguaggio orientato al calcolo matriciale
- Estremamente terso e sintetico (contra: Cobol!)
  - La somma degli elementi di un vettore: $(+/\omega)$
  - La media degli elementi di un vettore: $(+/\omega) \div \rho\omega$
  - I numeri primi da 1 a R: $(\sim R \in R \circ.\times R)/R \leftarrow 1 \downarrow \iota R$

Outline

Linguaggi e automi

Dalla matematica (Kleene, 1950) all'informatica:

- Origine in Chomsky 1955 (ma è un linguista!)
- Unicità DFA minimo (Myhill & Nerode, 1958)
- NFA (Rabin & Scott, 1959)
- LR (Knuth, 1965)
- LL (Rosenkrantz & Stearns, 1969)

Complessità

- Linear bounded automata: Myhill (1960)
- La definizione di classe: Hartmanis & Stearns (1965); Cobham (1965).
- Definizione assiomatica: Blum (1967)
- Riduzioni polinomiali, problemi completi per NP: Cook, Levine (1971).

Outline

Algoritmi e programmazione

- Pubblicazione di algoritmi per CACM (eg, Forsythe editor)
- Ford & Fulkerson (1956): MaxFlow
- Hoare (1960): QuickSort
- Knuth: The Art of Computer Programming
- Dijkstra: Correttezza, anche concorrente
- Floyd: Correttezza

Algorithm 64

CACM 4(7), 321 (1961)

ALGORITHM 64
QUICKSORT
C. A. R. Hoare
Elliott Brothers Ltd., Borehamwood, Hertfordshire, Eng.

```
procedure  quicksort (A,M,N); value M,N;
             array A; integer M,N;
comment  Quicksort is a very fast and convenient method of
sorting an array in the random-access store of a computer. The
entire contents of the store may be sorted, since no extra space is
required. The average number of comparisons made is 2(M−N) ln
(N−M), and the average number of exchanges is one sixth this
amount. Suitable refinements of this method will be desirable for
its implementation on any actual computer;
begin      integer I,J;
           if M < N then begin partition (A,M,N,I,J);
                               quicksort (A,M,J);
                               quicksort (A, I, N)
                         end
end        quicksort
```

Algorithm 63

ALGORITHM 63
PARTITION
C. A. R. HOARE
Elliott Brothers Ltd., Borehamwood, Hertfordshire, Eng.

```
procedure  partition (A,M,N,I,J);  value M,N;
             array A;  integer M,N,I,J;
```

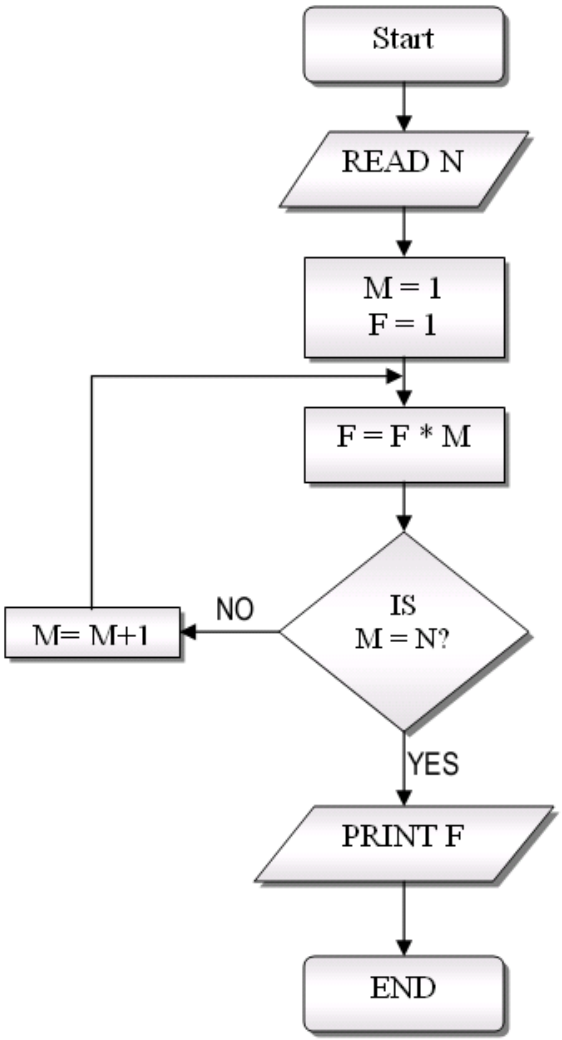
comment I and J are output variables, and A is the array (with subscript bounds M:N) which is operated upon by this procedure. Partition takes the value X of a random element of the array A, and rearranges the values of the elements of the array in such a way that there exist integers I and J with the following properties:

$M \leqq J < I \leqq N$ provided $M < N$
$A[R] \leqq X$ for $M \leqq R \leqq J$
$A[R] = X$ for $J < R < I$
$A[R] \geqq X$ for $I \leqq R \leqq N$

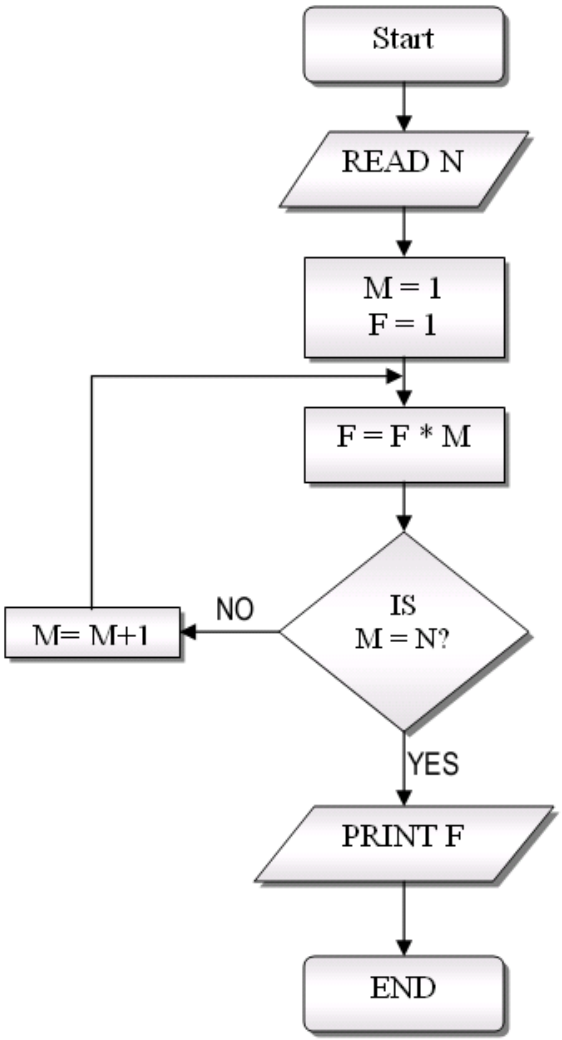
The procedure uses an integer procedure random (M,N) which chooses equiprobably a random integer F between M and N, and also a procedure exchange, which exchanges the values of its two parameters;

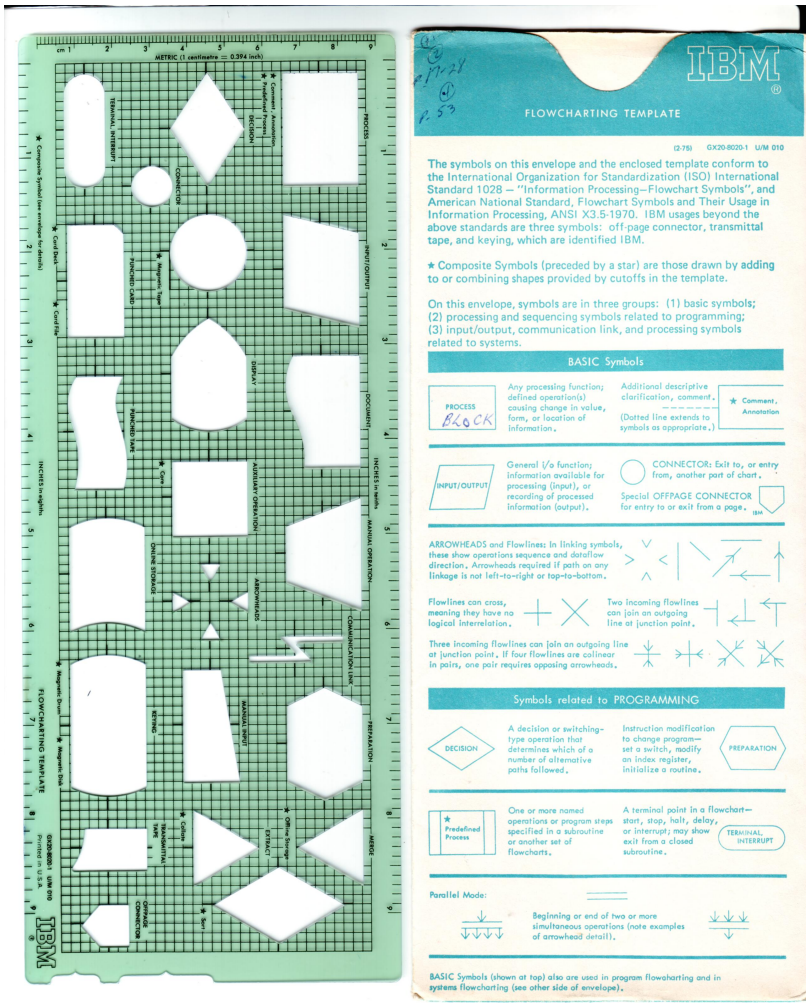
```
begin    real X;  integer F;
         F := random (M,N);  X := A[F];
         I := M;  J := N;
up:      for I := I step 1 until N do
                   if X < A [I] then go to down;
         I := N;
down:    for J := J  step −1 until M do
                   if A[J]<X then go to change;
         J := M;
change:  if I < J then begin exchange (A[I], A[J]);
                              I := I + 1; J := J − 1;
                              go to up
                         end
         else    if I < F then begin exchange (A[I], A[F]);
                              I := I + 1
                         end
         else    if F < J then  begin exchange (A[F], A[J]);
                              J := J − 1
                         end;
end      partition
```

Diagrammi di flusso



Diagrammi di flusso





The Art of Computer Programming

Donald E. Knuth (n. 1938)

1962 Contratto con Addison-Wesley: libro su compilatori

1963 Primo capitolo: sorting

He read many technical articles, and noticed the spotty and sometimes unreliable nature of the literature in the, then new, field of computer science. He saw the need for someone to write a book which organized and reliably presented what was then known about the field. Knuth was a good writer and had an instinct for trying to organize things, so he decided to tackle it. He used a quantitative rather than qualitative approach, and emphasized aesthetics—the creation of programs that are beautiful.

1965 Prima bozza (12 capitoli)

1966 Il progetto editoriale: 7 volumi...

The Art of Computer Programming, 2

Vol 1 "Fundamental Algorithms": Basic Concepts; Information Structures, 1968.

Vol 2 "Seminumerical Algorithms": Random Numbers; Arithmetic, 1969.

Vol 3 "Sorting and Searching" : Sorting; Searching, 1973.

Vol 4A "Combinatorial Algorithms Part 1"

Vol 4B "Combinatorial Algorithms Part 2"

Voll 5-7 : Scanning, Parsing, Context-free grammars, Compiler construction

Vol 4A a fascicoli dal 2005 in poi. In volume: 2011.

Nel mezzo: TEX (e molto altro...)

The Art of Computer Programming, 2

Vol 1 "Fundamental Algorithms": Basic Concepts; Information Structures, 1968.

Vol 2 "Seminumerical Algorithms": Random Numbers; Arithmetic, 1969.

Vol 3 "Sorting and Searching" : Sorting; Searching, 1973.

Vol 4A "Combinatorial Algorithms Part 1"

Vol 4B "Combinatorial Algorithms Part 2"

Voll 5-7 : Scanning, Parsing, Context-free grammars, Compiler construction

Vol 4A a fascicoli dal 2005 in poi. In volume: 2011.

Nel mezzo: TEX (e molto altro...)

Outline

AI

- Artificial Intelligence: John McCarthy 1956
- Il convegno a Dartmouth College: Newell, McCarthy, Simon, Minsky

- Automi neurali come rappresentazione fedele del cervello
- Manipolazioni simbolico fondamento della mente
- Grande (e non giustificato) ottimismo:
  in vent'anni i calcolatori sapranno fare quello che sa fare l'uomo (Simon, Minsky)

Outline

Premi Turing: 1-15

1966	Perlis	programming techniques, compilers
1967	Wilkes	EDSAC, program libraries
1968	Hamming	error-detecting codes
1969	Minsky	AI
1970	Wilkinson	numerical analysis
1971	McCarthy	AI, LISP
1972	Dijkstra	Algol, programmazione, algoritmi
1973	Bachman	primo DBMS, SE
1974	Knuth	analysis of algorithms, TAOCP
1975	Newell & Simon	AI
1976	Rabin & Scott	NFA
1977	Backus	FORTRAN e LP
1978	Floyd	Parsing, semantica, correttezza
1979	Iverson	APL
1980	Hoare	linguaggi di programmazione

Premi Turing: 16-30

1981	Codd	Database relazionali
1982	Cook	Complessità computazionale
1983	Thompson & Ritchie	Unix
1984	Wirth	LP
1985	Karp	Algoritmi, complessità
1986	Hopcroft & Tarjan	algorithms and data structures
1987	Cocke	Compilatori, RISC
1988	Sutherland	Grafica
1989	Kahan	Analisi numerica (IEEE 754)
1990	Corbató	Sistemi operativi (CTSS, Multics)
1991	Milner	LCF, ML, CCS
1992	Lampson	Personal workstations (@Xerox)
1993	Hartmanis & Stearns	Complessità computazionale
1994	Feigenbaum & Reddy	AI
1995	Blum	Complessità computazionale

Premi Turing: 31-45

1996	Pnueli	Logica temporale, verifica
1997	Engelbart	Interactive computing (mouse)
1998	Gray	database, transaction processing
1999	Brooks	Architettura, SE
2000	Yao	Complessità computazionale
2001	Dahl & Nygaard	PL: object-oriented (Simula)
2002	Rivest & Shamir & Adleman	public-key cryptography
2003	Kay	Smalltalk, personal computing
2004	Cerf & Kahn	TCP/IP
2005	Naur	PL: ALGOL 60
2006	Allen	Compilatori ottimizzanti
2007	Clarke & Emerson & Sifakis	Model checking
2008	Liskov	PL: data abstraction
2009	Thacker	Xerox Alto, Ethernet
2010	Valiant	probabilistic learning

Premi Turing: 46-

2011	Pearl	AI: causal reasoning
2012