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

#defi

doub

int m
  int i,
  double S = 0.;
  double a, b, delta;
  printf("Insert a, b: ");
  scanf("%lf %lf", &a, &b);
  delta = b - a;
  for (i = 0; i < N; i++) {
    x = a + delta * rand() / RAN
    S += f(x) * delta;
  }
  S /= N;
  printf("I = %f\n", S);
  return 0;
}
```

# Programmazione Scientifica

Luciano M. Barone
Enzo Marinari
Giovanni Organtini
Federico Ricci-Tersenghi

```
#include <stdio.h>
#include <math.h>
#incl

#def

doub

int
  in
  do
  do
  pr
  sc
  de
  fo
    i
    {
  }
  s
  pr
  re
}
```



- Inizio progetto: ottobre 2004
- I edizione: marzo 2006
- 650 pagine

```
#include <stdio.h>
#include <math.h>
#incl

#def

doub

int
  in
  do
  do
  pr
  sc
  de
  fo
    }
  S
  pr
  re
}
```


# La Sapienza investe sul calcolo

- La storia: dalla laurea quadriennale al "3+2"
  - finalmente il calcolo per i Fisici
- I corsi
  - Laboratorio di Calcolo (primo anno)
  - Laboratorio di Fisica Computazionale I (secondo anno)
  - Laboratorio di Fisica Computazionale II (terzo anno)
  - e molti altri (Fisica Computazionale, C++, Java, etc.)
  - Insomma, alla Sapienza abbiamo fatto la scelta di investire sul calcolo (per esempio Programmazione++, concepito in modo del tutto innovativo da Giorgio Parisi)

```
#include <stdio.h>
#include <math.h>
#incl

#def

doub

int
  in
  do
  do
  pr
  sc
  de
  fo
    }
  S
  pr
  re
}
```


# Vogliamo insegnare un linguaggio

- Dare informazione che serva a lavorare nella scienza. Un po' come i corsi di matematica a fisica, alla Smirnov… qualche teorema in meno (ma rigore comunque) e molto "saper fare"
- Abbiamo scelto il linguaggio C: "equazioni differenziali"… procedurali piuttosto che oggetti (è diverso per programmi algebrici o gestione dati complessi, ma viene dopo)
- Esempi e  metodi che crescono con lo studente (Cv scientifico, Scienze MFN e Ingegneria)

```
#include <stdio.h>
#include <math.h>
#incl

#def

doub

int
  in
  do
  do
  pr
  sc
  de
  fo
    }
  S
  pr
  re
}
```


# La ricerca dei testi

- Qualcosa cosí non esisteva
  - Ad esempio: Gould Tobochnik ha molti problemi ben scelti ma non insegna il  linguaggio
  - Manuali
  - Testi per specialisti hardware
  - Libri di analisi numerica (avanzati)
- Valutazione dei corsi da parte degli studenti:
  - positiva sui contenuti
  - negativa sui libri di testo

```
#include <stdio.h>
#include <math.h>
#incl

#def

doub

int
  in
  do
  do
  pr
  sc
  de
  fo
    }
    S
  pr
  re
}
```


# Quindi: serviva un libro

- (peccato… è un lavoro duro)
- (bene… è una fortuna insegnare un corso appena nato)
- Insegniamo il C, da a+b a dettagli di ottimizzazione e di gestione di liste complesse
- Lo facciamo insegnando a trattare problemi scientifici
- E dicendo come funziona un calcolatore
- fattoriale, soluzione di equazioni, ricerca di primi, arrotondamento, ordinamento, ricerca, sistemi di equazioni, chi quadro, interpolazione, integrazione, Newton, numeri pseudocasuali, cammini aleatori, cluster connessi, percolazione, automi cellulari, grafi, strutture dati avanzate, problemi di ottimizzazione, metodi stocastici, …

```
#include <stdio.h>
#include <math.h>
#incl

#def

doub

int
  in
  do
  do
  pr
  sc
  de
  fo
    }
  S
  pr
  re
}
```


# La divisione in parti

- Analisi del curriculum
  - tre anni ⇒ tre parti
  - target: ingegneria e scienze
- Parte 1: il computer come strumento, sintassi di base del linguaggio C, derivazione e integrazione numerica, interpolazione
- Parte 2: tecniche avanzate di programmazione, allocazione dinamica della memoria, le **`struct`**, soluzione delle equazioni della fisica, studio della percolazione, metodi stocastici
- Parte 3: algoritmi, metodi Monte Carlo, problemi di ottimizzazione e di ricerca, ordinamento, grafi
- Contiene, spiega e discute algoritmi avanzati utili…

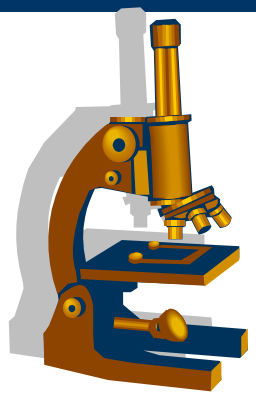
```
#include <stdio.h>
#include <math.h>
#incl

#def

doub

int
  in
  do
  do
  pr
  sc
  de
  fo
    }
  }
  S
  pr
  re
}
```

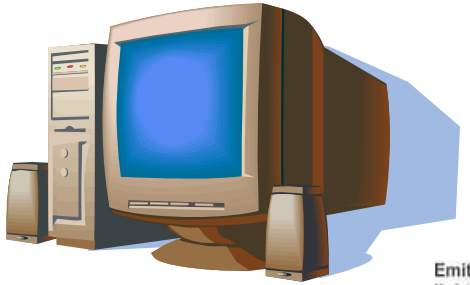

# La scrittura del testo

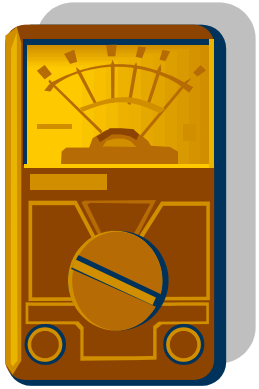
- Coerenza delle parti
- I codici in C
- La veste grafica
- Il sito web
  http://www.programmazionescientifica.org
- Il futuro: traduzione in altre lingue

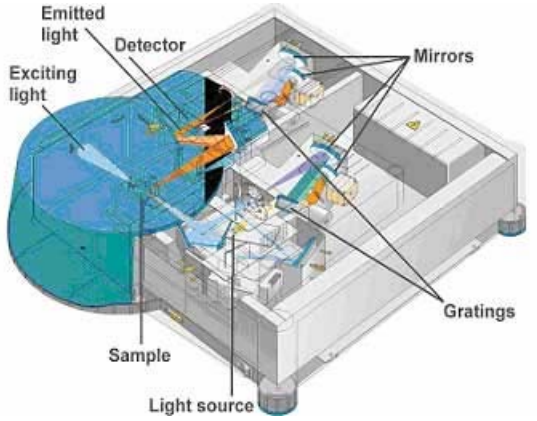

```
#include <stdio.h>
#include <math.h>
#incl

#def

doub

int
   in
   do
   do
   pr
   sc
   de
   fo
      }
   S
   pr
   re
}
```

# Il computer come strumento



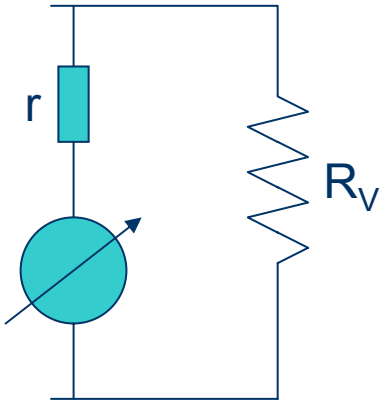






```
#include <stdio.h>
#include <math.h>
#incl

#def

doub

int
  in
  do
  do
  do
  pr
  sc
  de
  fo
    }
  S
  pr
  re
}
```


# Conoscere gli strumenti

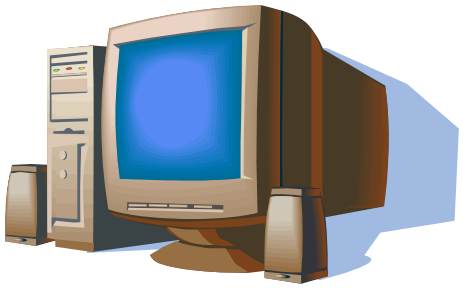


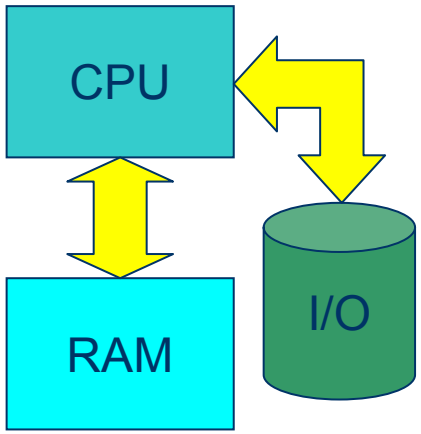

$$V_{mis} = V \frac{r}{R_v + r}$$

L'uso di ogni strumento può perturbare il sistema. I risultati devono saper essere interpretati. La presenza dello strumento, non correttamente trattata, può produrre effetti sulla misura.

```
#include <stdio.h>
#include <math.h>
#incl

#def

doub

int
  in
  do
  do
  pr
  sc
  de
  fo
    }
  S
  pr
  re
}
```


# Conoscere lo strumento computer





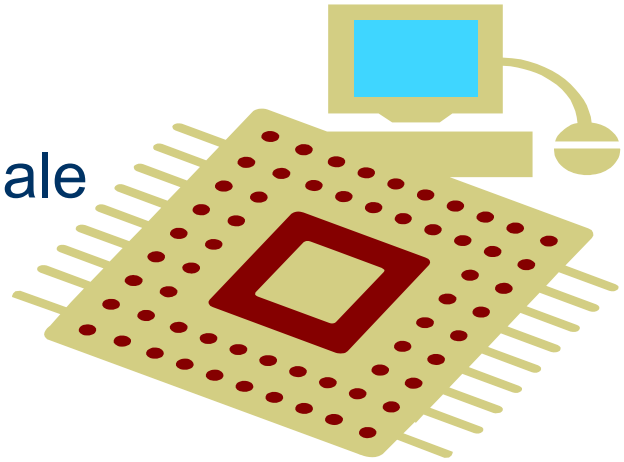

$$R = f(dati, arch.)$$

L'uso del computer per acquisire e analizzare dati o per calcolare può produrre risultati inaspettati se non si sa tenere conto degli effetti della sua presenza.

```
#include <stdio.h>
#include <math.h>
#incl

#def

doub

int
  in
  do
  do
  pr
  sc
  de
  fo
    }
  S
  pr
  re
}
```


# Le tecnologie

- Evoluzione rapidissima
  - inutile entrare in dettagli architetturali
- Principî generali stabili
  - Toy CPU
  - Linguaggio Macchina artificiale semplificato



```
#include <stdio.h>
#include <math.h>
#incl

#def

doub

int
  in
  do
  do
  pr
  sc
  de
  fo
    }
  }
  S
  pr
  re
}
```


*È vero che il software non potrebbe esercitare i poteri della sua leggerezza se non mediante la pesantezza dell'hardware; ma è il software che comanda, che agisce sul mondo esterno e sulle macchine, le quali esistono solo in funzione del software*

Italo Calvino, Lezioni Americane

```
#include <stdio.h>
#include <math.h>
#incl

#def

doub

int
  in
  do
  do
  pr
  sc
  de
  fo
    }
  S
  pr
  re
}
```


# Introduzione al linguaggio

- Il linguaggio rispetta una sintassi, noiosa ma necessaria
- Esempi *ad hoc* per chi si occupa di scienza, per evidenziare i problemi che derivano dall'uso "cieco" dello strumento
- Un caso esemplare: introduzione ai cicli iterativi ➔ il costrutto **`for`**
- Un programma apparentemente innocuo che può produrre risultati inaspettati: la media di molti numeri


# Il programma

```
#include <stdio.h>
#include <math.h>
#incl

#def

doub

int
  in
  do
  do
  pr
  sc
  de
  fo
    }
  S
  pr
  re
}
```

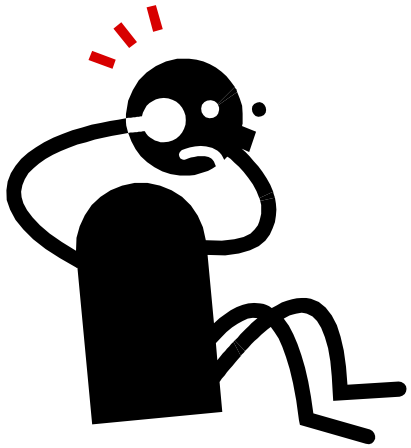
```
float x = 7., S = 0.;
for (i = 0; i < N; i++) {
   S += x;
}
printf("S = %f\n", S);
```

Per N = $10^7$ ci aspettiamo S = 70 000 000


# Il programma

```
#include <stdio.h>
#include <math.h>
#incl
#def
doub
int
  in
  do
  do
  pr
  sc
  de
  fo
    }
  S
  pr
  re
}
```

```
float x = 7., S = 0.;
for (i = 0; i < N; i++) {
   S += x;
}
printf("S = %f\n", S);
```



Il risultato è invece S = 77 603 248

```
#include <stdio.h>
#include <math.h>
#incl

#def

doub

int
  in
  do
  do
  pr
  sc
  de
  fo
    }
  S
  pr
  re
}
```


# Cosa avviene e come si cura

$S = 16777222 = 2^{24}(1+2^{-22}+2^{-23})$

$x = 7 = 2^{2}(1+2^{-1}+2^{-2})$

$S' = 2^{24}(1+2^{-22}+2^{-23}+2^{-22}+2^{-23}+2^{-24})$

```
#include <stdio.h>
#include <math.h>
#incl

#def

doub

int
  in
  do
  do
  pr
  sc
  de
  fo
    x
    {
  }
  S
  pr
  re
}
```


# Cosa avviene e come si cura

$S = 16777222 = 2^{24}(1+2^{-22}+2^{-23})$

$x = 7 = 2^{2}(1+2^{-1}+2^{-2})$

$S' = 2^{24}(1+2^{-22}+2^{-23}+2^{-22}+2^{-23}+2^{-24}) =$

$= 16777228$

```
#include <stdio.h>
#include <math.h>
#incl
#def
doub
int
  in
  do
  do
  pr
  sc
  de
  fo
  }
  S
  pr
  re
}
```


# L’algoritmo di Kahan

- Il troncamento produce effetti di ordine $2^{-23}$
- La propagazione di questo errore produce effetti disastrosi
- Per spiegare perché il numero aumenta si devono conoscere i dettagli dell’architettura
- Soluzione: algoritmo di Kahan (p.104)

```
#include <stdio.h>
#include <math.h>
#incl

#def

doub

int
  in
  do
  do
  pr
  sc
  de
  fo
    }
  S
  pr
  re
}
```


# L'algoritmo di Kahan

- Invece di sommare **x**, sommo **x′ = x+**δ, con δ**=0** inizialmente
- Calcolo la somma **s′ = s + x′**
- Calcolo la differenza tra **s** e **s′**, e aggiungo **x′**. Assegno il valore ottenuto a δ
- Pongo **s = s′**

- Algebricamente non ho fatto nulla, ma l'algebra dei computer è diversa…

```
#include <stdio.h>
#include <math.h>
#incl

#def

doub

int
  in
  do
  do
  pr
  sc
  de
  fo
    }
  S
  pr
  re
}
```


# Linguaggio e algoritmi

- Quali strumenti servono per realizzare un dato algoritmo
- Come posso modellizzare un problema con il linguaggio di cui dispongo
- Usi creativi di strumenti sintattici

```
#include <stdio.h>
#include <math.h>
#incl

#def

doub

int
  in
  do
  do
  pr
  sc
  de
  fo
    }
  }
  S
  pr
  re
}
```


# Alcuni esempi

- Calcolo della derivata
  - tool: puntatori a funzione
  - warning: precisione numerica
- Integrale con il metodo MC
  - warning: statistica
- Automi cellulari
  - tool:  operatori di singolo bit

```
#include <stdio.h>
#include <math.h>
#incl

#def

doub

int
  in
  do
  do
  pr
  sc
  de
  fo
    }
  S
  pr
  re
}
```


# Calcolo della derivata

- Si calcola il rapporto incrementale iterando per $\varepsilon$ sempre più piccolo
- ...ma, attenzione ! Non sempre piccolo è bello
- La funzione derivanda è generica


# Il codice

```
double derivative(double (*f), double x,
                  double epsilon) {
  return ( f(x+epsilon)-f(x))/epsilon;
}
```

Chiamata in un ciclo in cui la variabile **epsilon** parte da 1.0 ed è via via divisa per 10.

Applichiamo a $\sqrt{x}$ per x=1, la cui derivata vale 0.5

```
#include <stdio.h>
#include <math.h>
#incl

#def

doub

int
   in
   do
   do
   pr
   sc
   de
   fo
      }
   S
   pr
   re
}
```


# Il risultato

```
1.000000e+00  0.414214
1.000000e-01  0.488088
1.000000e-02  0.498756
1.000000e-03  0.499875
1.000000e-04  0.499988
1.000000e-05  0.499999
1.000000e-06  0.500000
1.000000e-07  0.500000
1.000000e-08  0.500000
1.000000e-09  0.500000
1.000000e-10  0.500000
1.000000e-11  0.500000
1.000000e-12  0.500044
1.000000e-13  0.499600
1.000000e-14  0.488498
1.000000e-15  0.444089
1.000000e-16  0.000000
1.000000e-17  0.000000
1.000000e-18  0.000000
```

**`epsilon`** troppo grande produce un risultato approssimato

**`epsilon`** troppo piccolo produce un risultato sbagliato o addirittura paradossale

```
#include <stdio.h>
#include <math.h>
#incl

#def

doub

int
  in
  do
  do
  pr
  sc
  de
  fo
    }
  S
  pr
  re
}
```

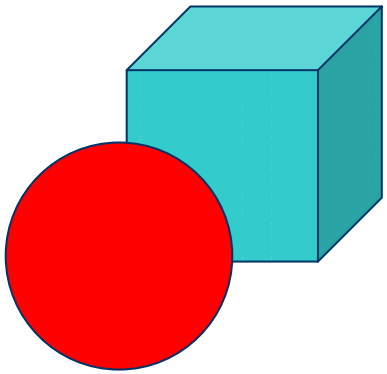

# Integrare con il metodo MC

- Calcolo numerico di integrali
- Integrali multipli di area o di volume
- Come per la derivata, porre attenzione alle limitazioni numeriche
  - somma di molti termini
  - statistica “giusta”
  - fermarsi in tempo

```
#include <stdio.h>
#include <math.h>
#incl
#def
doub
int
  in
  do
  do
  pr
  sc
  de
  fo
    }
  S
  pr
  re
}
```


# Un esempio di integrale

Una sfera centrata in (0,0,0) R=1, intersecata con un cubo di lato 2 centrato in (1,1,1).
Calcoliamo il volume dell'intersezione.



```
#include <stdio.h>
#include <math.h>
#incl

#def

doub

int
  in
  do
  do
  pr
  sc
  de
  fo
    }
  }
  S
  pr
  re
}
```


# Calcolo numerico

- Basta generare in modo aleatorio N punti (x,y,z) interni al cubo e contare i k punti interni alla sfera.

  $V_{s \cap c} = V_c \, k / N$

```
#include <stdio.h>
#include <math.h>
#incl

#def

doub

int
  in
  do
  do
  pr
  sc
  de
  fo
    }
  S
  pr
  re
}
```


# Uso creativo del linguaggio

- Operatori di singolo bit
  &(AND), |(OR), ^(XOR), ~(NOT), >>,<<
  - componenti molto tecniche e noiose da studiare
  - non appare a prima vista l'utilità
- ma con essi si possono creare applicazioni molto interessanti
  - Bit calculators
  - Automi cellulari

```
#include <stdio.h>
#include <math.h>
#incl
#def
doub
int
  in
  do
  do
  pr
  sc
  de
  fo
  }
  S
  pr
  re
}
```


# Automi cellulari

Si definisce una configurazione di partenza e una regola di evoluzione. Es.:

| | | |
|---|---|---|
| 1 | 1 | $1 \rightarrow 0$ |
| 1 | 1 | $0 \rightarrow 1$ |
| 1 | 0 | $1 \rightarrow 0$ |
| 1 | 0 | $0 \rightarrow 1$ |
| 0 | 1 | $1 \rightarrow 1$ |
| 0 | 1 | $0 \rightarrow 0$ |
| 0 | 0 | $1 \rightarrow 1$ |
| 0 | 0 | $0 \rightarrow 0$ |

Questa è la cosiddetta Regola 90 di Wolfram
Dal numero
01011010

```
#include <stdio.h>
#include <math.h>
#incl

#def

doub

int
  in
  do
  do
  pr
  sc
  de
  fo
    }
  }
  S
  pr
  re
}
```

# Il codice


La parte di codice rilevante è molto semplice

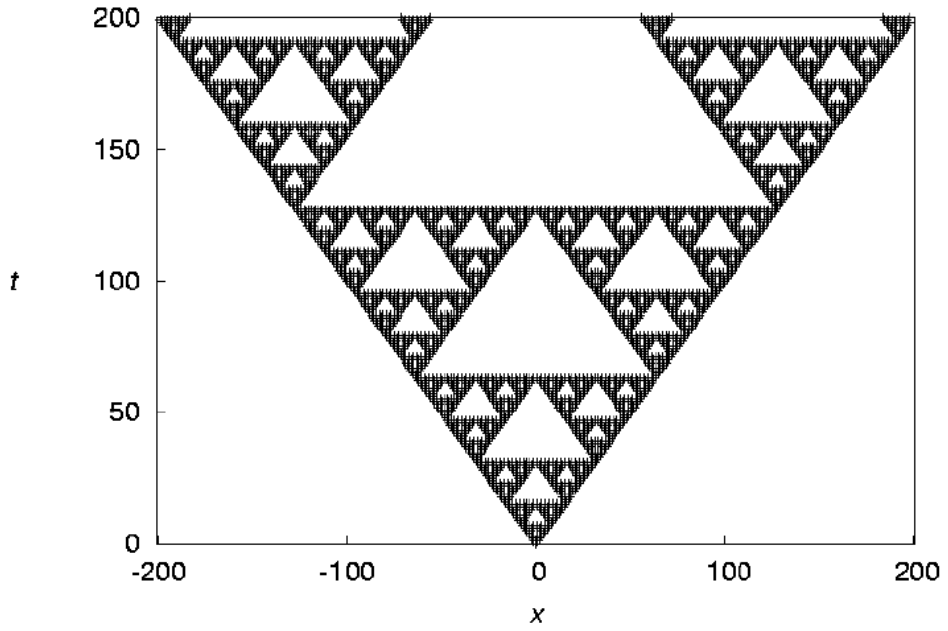
```
CP[site] = C[site-1] ^ C[site+1];
```

dove **site** rappresenta il sito in evoluzione, e i vettori **C** e **CP** contengono gli stati prima e dopo ogni passo evolutivo.

Con questa semplice regola si ottiene...

```
#include <stdio.h>
#include <math.h>
#incl

#def

doub

int
  in
  do
  do
  pr
  sc
  de
  fo
    }
  S
  pr
  re
}
```


# La regola 90 di Wolfram



```
#include <stdio.h>
#include <math.h>
#incl
#def
doub
int
  in
  do
  do
  pr
  sc
  de
  fo
  }
  S
  pr
  re
}
```

# Algoritmi...
# ...e come programmarli


Dato un problema scientifico

➔ scelta di un buon algoritmo

➔ come programmarlo in modo efficiente

```
#include <stdio.h>
#include <math.h>
#incl

#def

doub

int
  in
  do
  do
  pr
  sc
  de
  fo
    }
  }
  s
  pr
  re
}
```


# Scelta dell'algoritmo

- Complessità computazionale
  dipendenza delle risorse di calcolo dalla taglia del problema
- Livello di approssimazione
  errore = $O(\varepsilon^{\alpha})$

compromesso

sempre in mente il problema scientifico!

```
#include <stdio.h>
#include <math.h>
#incl
#def
doub
int
  in
  do
  do
  pr
  sc
  de
  fo
  }
  S
  pr
  re
}
```

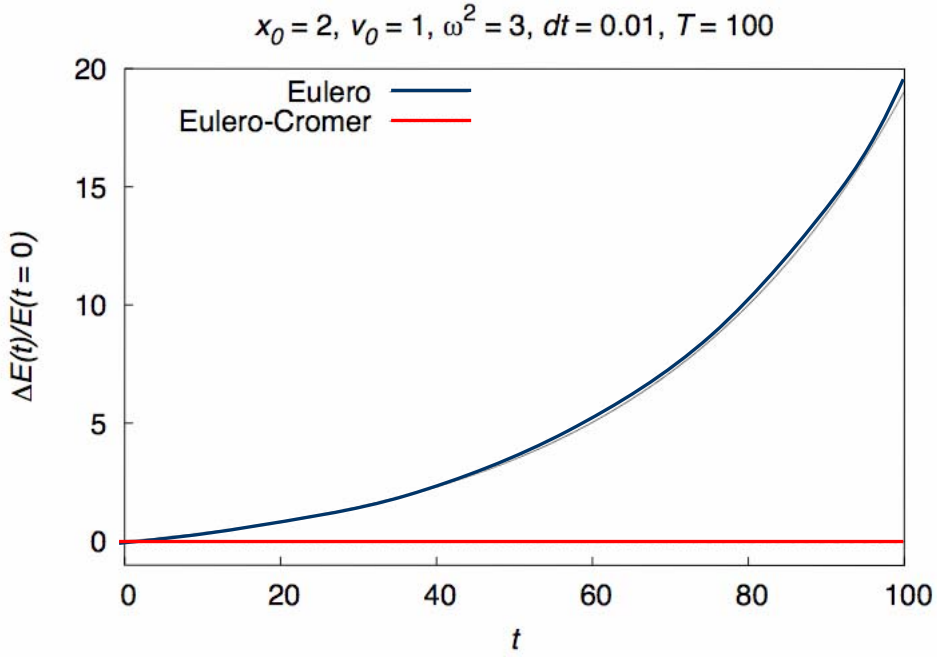
# Un esempio: integrazione equazioni differenziali


df(t)/dt ⟶ [f(t+Δt) - f(t)] / Δt

$O(\Delta t)$: Eulero, Eulero-Cromer ⟶ stabilità

$O(\Delta t^2)$: Verlet

$O(\Delta t^{\geq 2})$: Runge-Kutta



```
#include <stdio.h>
#include <math.h>
#incl
#def
doub
int
  in
  do
  do
  pr
  sc
  de
  fo
  }
  S
  pr
  re
}
```

# Ampio panorama di algoritmi...


- integraz. di equaz. differenziali
- interpolaz. e integraz. di funzioni
- ordinamento di sequenze (quicksort, heapsort,...)
- strutture di dati (heap, albero binario,...)
- algoritmi su grafo (ricerca componenti conesse, MST)
- Monte Carlo
- ottimizzazione (ricerca di un minimo):
  - metodi del gradiente, programmazione lineare
  - algoritmi genetici, simulated annealing

```
#include <stdio.h>
#include <math.h>
#incl
#def
doub
int
  in
  do
  do
  pr
  sc
  de
  fo
  }
  S
  pr
  re
}
```


# Ampio panorama di algoritmi...
# ...e delle teorie su cui si basano

- integraz. di equaz. differenziali } sviluppo di Taylor
- interpolaz. e integraz. di funzioni } basi di polinomi
- ordinamento di sequenze } ricorsione
- strutture di dati
- generatori numeri pseudocasuali — teoria dei numeri
- Monte Carlo — catene Markov
- ottimizzazione (ricerca di un minimo):
  - metodi del gradiente, programmazione lineare
  - algoritmi genetici, simulated annealing

```
#include <stdio.h>
#include <math.h>
#incl

#def

doub

int
  in
  do
  do
  pr
  sc
  de
  fo
    }
  }
  S
  pr
  re
}
```


# Non è un ‘Numerical Recipes’ !

Per ogni algoritmo:

- ne spieghiamo la teoria di base;
- lo applichiamo subito a modelli di interesse scientifico.

Niente studio astratto degli algoritmi

Partiamo sempre da un problema scientifico (il modello che vogliamo studiare)

Interesse nel risultato stimola gli studenti


# I modelli... come un laboratorio

- Oscillatore armonico
- Percolazione
- Grafi aleatori
- Modello di Ising

Risultato noto ⟶ test del programma

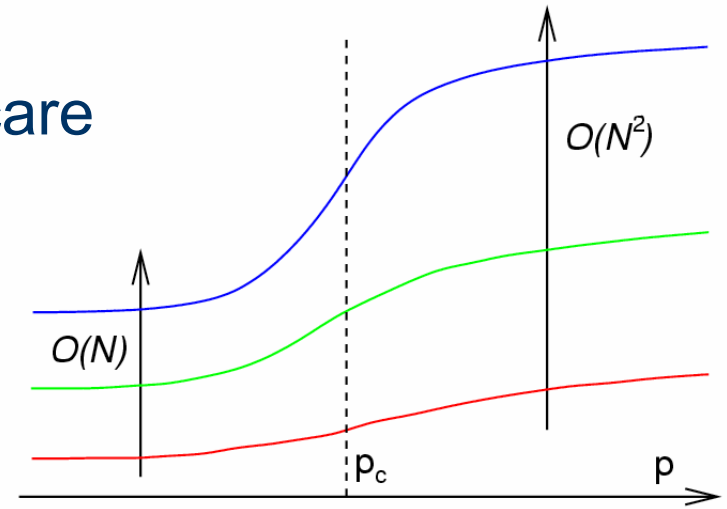
Risultato ignoto ⟶ curiosità, partecipazione


# Connessioni tra algoritmi e modelli

Percolazione: transizione di fase, cluster connessi, raggio di girazione (lungh. correlaz.)

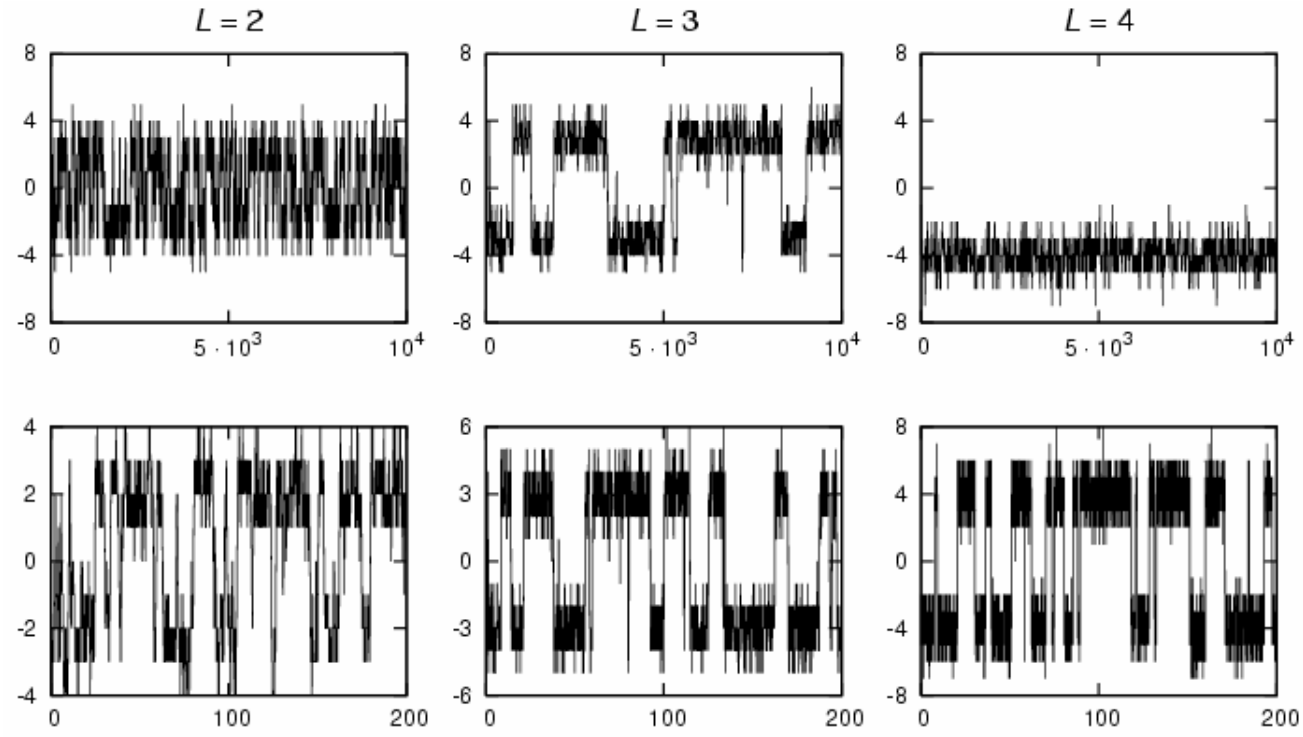
Algoritmo semplice: cluster ⟷ lista concatenata

il tempo per identificare tutti i cluster cresce molto intorno al punto critico




# Ancora sulle connessioni tra algoritmi e modelli

Transizione di fase ⟷ rottura dell'ergodicità



```
#include <stdio.h>
#include <math.h>
#incl

#def

doub

int
  in
  do
  do
  pr
  sc
  de
  fo
    }
    }
  }
  s
  pr
  re
}
```

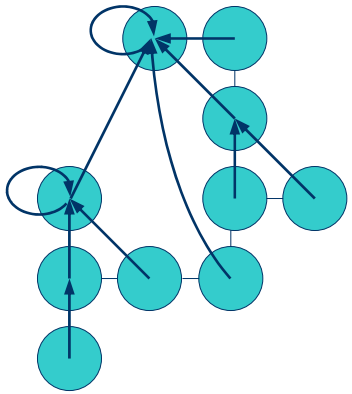

# Gli algoritmi migliorano...

Dopo le strutture di dati ad albero mostriamo un algoritmo migliore per contare i cluster (par. 17.5)

cluster ⟷ albero (di bassa profondità)

costa poco
unire due cluster

risolto il problema del rallentamento critico!



```
#include <stdio.h>
#include <math.h>
#incl
#def
doub
int
  in
  do
  do
  pr
  sc
  de
  fo
    }
  S
  pr
  re
}
```


# Dall'algoritmo al programma

**Consigli** sui migliori elementi di programmazione per realizzare l'algoritmo (ad es. la struttura dati piú efficiente), ma anche quei **"trucchi"** (che non si trovano né nei manuali del linguaggio né nei testi di algoritmi):

- puntatori piuttosto che indici;
- uso delle look-up table;
- ottimizzazione accessi in memoria;
- ottimizzazione della sequenza di istruzioni (*pipeline*);
- . . . .

```
#include <stdio.h>
#include <math.h>
#incl
#def
doub
int
  in
  do
  do
  pr
  sc
  de
  fo
  }
  S
  pr
  re
}
```

# Altri utensili utili nella programmazione scientifica


- **`gnuplot`** (per fare i grafici)
- analisi statistica
  - test del $\chi^2$ (e relativa tabella)
  - test di Kolmogorov-Smirnov
  - media a blocchi
  - jackknife
- analisi transizioni di fase
  - teoria del riscalamento al punto critico

**Un libro da tenere a portata di mano!**