

Programmazione (imperativa)

Corso di Laurea in Informatica

Roberto Cordone

DI - Università degli Studi di Milano



Lezioni: **Lunedì 12.00 - 13.00 e 14.00 - 16.00** **Mercoledì 14.00 - 17.00**
Laboratorio: **Giovedì 12.00 - 13.00 e 14.00 - 17.00**
Ricevimento: **su appuntamento**
Tel.: **02 503 16235**
E-mail: **roberto.cordone@unimi.it**
Web page: **<http://homes.di.unimi.it/~cordone/courses/2015-prog/2015-prog.html>**

Obiettivi del modulo

- acquisire uno **strumento per risolvere problemi** pratici in modo meccanico, e quindi

- ① con una **garanzia di correttezza**
- ② in un **tempo inferiore**
- ③ anche se di **grandi dimensioni**

dopo averli ricondotti a problemi matematici

- acquisire una **forma mentis adeguata a risolvere problemi**
- acquisire uno **stile nella creazione di oggetti funzionali**

Perché uno stile?

Per fare un favore a se stessi...



Perché risolvere problemi e creare oggetti?

Per riconoscere sfide...



Perché risolvere problemi e creare oggetti?

... superare ostacoli ...

Congratulations. You've made it to level 2. Go to **www.Linux.org** and enter *Bobsyouruncle* as the login and the answer to this equation as the password.

$$f(1)= 7182818284$$

$$f(2)= 8182845904$$

$$f(3)= 8747135266$$

$$f(4)= 7427466391$$

$$f(5)= \underline{\hspace{2cm}}$$

Perché risolvere problemi e creare oggetti?

... e cogliere occasioni. ...



Congratulations.

Nice work. Well done. Mazel tov. You've made it to Google Labs and we're glad you're here.

One thing we learned while building Google is that it's easier to find what you're looking for if it comes looking for you. What we're looking for are the best engineers in the world. And here you are.

As you can imagine, we get many, many resumes every day, so we developed this little process to increase the signal to noise ratio. We apologize for taking so much of your time just to ask you to consider working with us. We hope you'll feel it was worthwhile when you look at some of the interesting projects we're developing right now. You'll find links to more information about our efforts below, but before you get immersed in machine learning and genetic algorithms, please send your resume to us at problem-solver@google.com.

We're tackling a lot of engineering challenges that may not actually be solvable. If they are, they'll change a lot of things. If they're not, well, it will be fun to try anyway. We could use your big, magnificent brain to help us find out.

Some information about our current projects:

- [Why you should work at Google](#)
- [Looking for interesting work that matters to millions of people?](#)
- <http://labs.google.com>

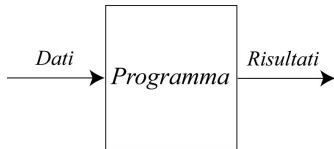
©2004 Google

... come la campagna virale di Google del 2004

1. Programma = Macchina

Un programma è una macchina che trasforma dati in risultati attraverso una sequenza di operazioni

- ① determinate meccanicamente dai dati
- ② elementari (perfettamente definite e di durata limitata)



2. Programma \neq Codice

Un programma non è il codice che lo rappresenta

In passato, realizzare un programma era

- collegare opportunamente ingranaggi, alberi a camme, ...
- collegare opportunamente cavi, relais, ...
- perforare opportunamente schede

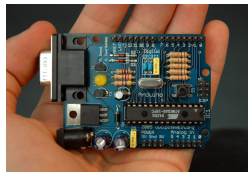
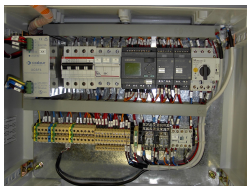
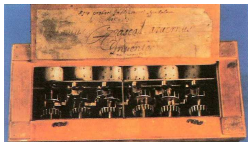
Oggi, realizzare un programma è

- scrivere codice in linguaggi di programmazione (es., C)
- collegare opportunamente registri e porte logiche

In futuro?

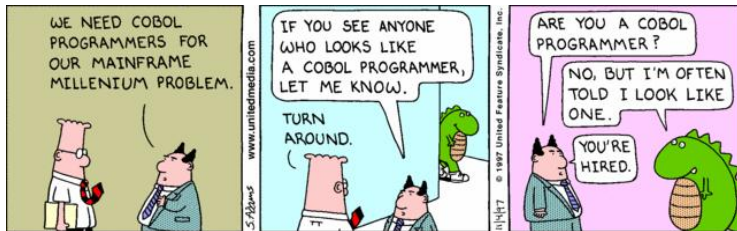
La tecnologia cambia, il programma rimane lo stesso

2. Programma \neq Codice



2. Programma \neq Codice

Programmare non significa imparare un linguaggio di programmazione

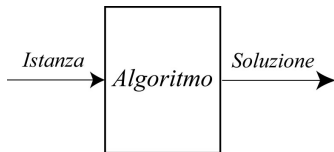


- i linguaggi si estinguono: meglio non estinguersi con loro
- le abilità rare valgono: conoscere più linguaggi è un'abilità rara

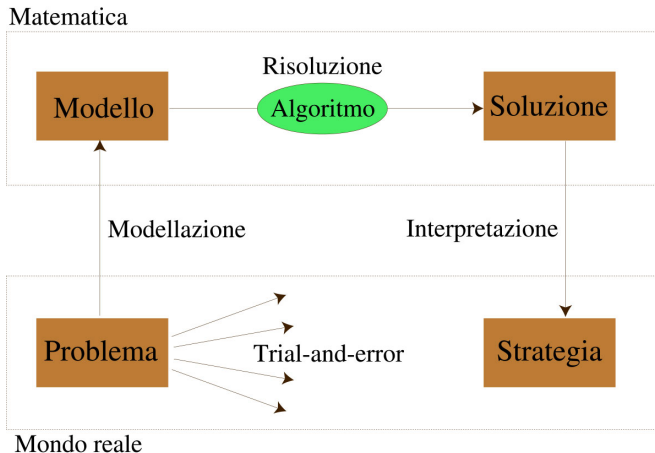
3. Algoritmo = Programma utile

Algoritmo risolvete: trasforma un'istanza di un problema nella soluzione attraverso una sequenza di operazioni

- ① determinate meccanicamente dall'istanza
- ② elementari (perfettamente definite e di durata limitata)
- ③ di durata complessiva limitata



4. $P \rightarrow M \xrightarrow{A} Sol \rightarrow Str$



Modellazione e interpretazione sono le fasi più delicate e creative

Esempi di problema (1)

Alcuni esempi tipici di problemi affrontati con la programmazione

① Crittografia/decifrazione

- Istanza: testo in chiaro / testo cifrato
- Soluzione: testo cifrato / testo in chiaro

② Correzione automatica di testi

- Istanza: testo corrotto
- Soluzione: testo ricostruito

③ Interpretazione di immagini

- Istanza: immagine in formato bitmap
- Soluzione: elenco di oggetti riconosciuti

Esempi di problema (2)

Alcuni esempi tipici di problemi affrontati con la programmazione

④ Navigazione su strada

- Istanza: rete stradale, origine e destinazione, preferenze
- Soluzione: percorso ottimo

⑤ Pianificazione della produzione

- Istanza: sistema di (dis)equazioni che descrivono un impianto e la relazione fra produzione e profitto
- Soluzione: livelli di produzione ottimali

⑥ Battaglia navale

- Istanza: griglia, posizioni colpite, navi mancanti, navi colpite, navi affondate
- Soluzione: prossimo bersaglio

Il modulo di “Programmazione imperativa” si compone di

- 12 lezioni di teoria da 3 ore
- 6 laboratori da 4 ore (comprese in 3 per motivi logistici)
 - i primi 4 applicano i concetti descritti in teoria
 - gli ultimi 2 sono di preparazione all'esame

Nelle lezioni teoriche vedrete girare esempi di codice
(se avete un portatile e li scaricate, potete lanciaarli durante la lezione)

Qualsiasi ambiente standard va bene

- per macchine Linux gcc è installato in ogni distribuzione (ne esistono molte trascrizioni per macchine Windows)
- in laboratorio useremo Dev-C++ (versione 4), che include
 - compilatore gcc
 - ambiente grafico (sconsigliato se volete imparare)
 - debugger (forse a fine corso)

(<http://www.bloodshed.net/devcpp.html>)

- per Windows 8 esiste una versione aggiornata (<http://sourceforge.net/projects/orwelldevcpp>)
- per Windows 10 fatemi sapere se funziona la versione Win8

Per imparare, sono meglio gli ambienti semplici

Gli ambienti avanzati spesso non sono standard

L'esame ha un voto solo, ma si compone di due moduli indipendenti

- **dovete superare entrambi i moduli**
- **potete ripeterli** illimitatamente, anche se avete già un voto
- **il voto complessivo è la media** dei voti dei due moduli
- **ciascun singolo voto scade dopo un anno**
(es.: il voto di giugno 2015 vale fino a giugno 2016 compreso)
- **per registrare** dovete
 - 1 **iscrivervi a un appello**
 - 2 **contattare i due docenti**
 - 3 **ricordare loro le date in cui avete sostenuto le due parti**

- 1 venire a lezione (i lucidi sono superficiali)
- 2 prendere appunti completi, non glossare i lucidi
(chi non sa scrivere quel che ha sentito, non lo ha capito)
- 3 interrompere per chiedere chiarimenti o conferme e segnalare incomprensioni
- 4 fare esercizi a casa (36 + 24 ore non sono sufficienti)
- 5 studiare sugli appunti o su un libro
(il lucido induce a credere di aver capito, ma si sta solo ripetendo)
- 6 studiare un po' per volta, ritornando sugli argomenti più vecchi
(si scopriranno cose che erano sfuggite la prima volta e nuove relazioni fra concetti)