



Introduzione

Informatica B

Perchè studiare l'informatica?

- ❑ Perché ha a che fare con quasi tutto quello con cui abbiamo a che fare ogni giorno
- ❑ Perché è uno strumento fondamentale per progettare l'innovazione di domani



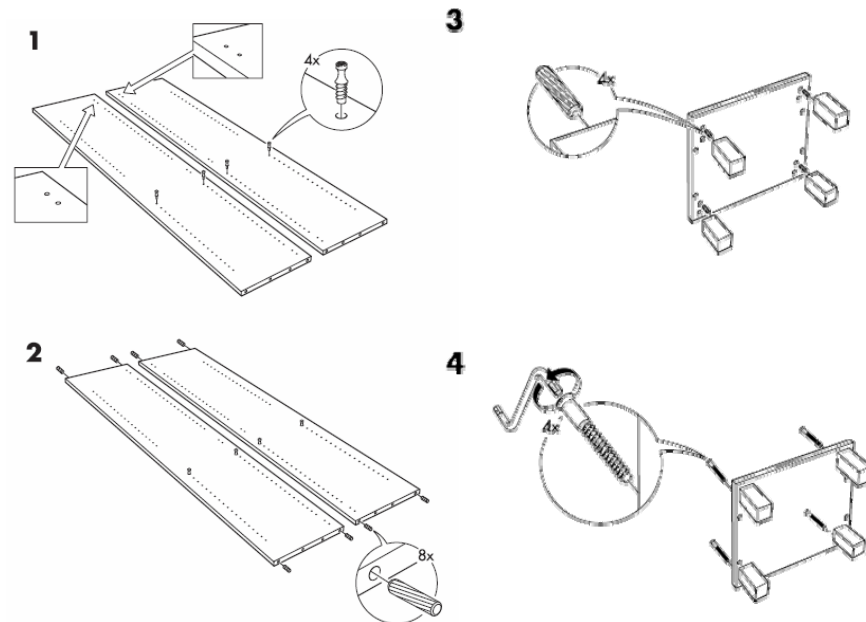
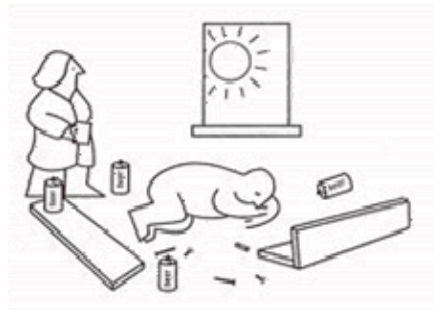
Cos'è l'informatica?

- ❑ È la scienza che si occupa della **rappresentazione** dell'**informazione** e della sua **elaborazione e gestione**
 - ▶ Si occupa dell'informazione, che fa parte di ogni attività umana, e non riguarda solo i calcolatori
 - ▶ Si occupa della rappresentazione, cioè di come modellare la realtà astruendo gli aspetti importanti da quelli trascurabili
 - ▶ Si occupa di elaborare e gestire l'informazione, cioè di trasformarla opportunamente per raggiungere lo scopo desiderato
- ❑ È lo studio sistematico degli **algoritmi** che descrivono e trasformano l'informazione: la loro teoria, analisi, progetto, efficienza, realizzazione e applicazione

Cos'è un algoritmo?

- Una sequenza finita di operazioni elementari tali che:
 - ▶ siano comprensibili ad uno specifico esecutore
 - ▶ possano essere eseguite senza ambiguità
 - ▶ permettano di risolvere uno specifico problema

- Es. istruzioni IKEA
 - ▶ passi elementari
 - ▶ senza ambiguità
 - ▶ raggiungono lo scopo



Esempio: gestione biblioteca

- ❑ Libri disposti sugli scaffali
- ❑ Ogni libro si trova in una precisa posizione definita da due coordinate: scaffale e posizione $\langle s,p \rangle$
- ❑ La biblioteca ha uno schedario, **ordinato** per autore/i e titolo

Autore/i: Atzeni, Paolo
Ceri, Stefano
Paraboschi, Stefano
Torlone, Riccardo
Titolo: Database Systems,
McGraw-Hill, 1999
Scaffale: 35
Posizione: 21

Algoritmo per la ricerca di un libro

1. Chiedi informazioni sul libro richiesto
2. **Preleva** il libro richiesto

Non è un passo elementare, comprensibile ed eseguibile dall'esecutore... deve essere dettagliato con un sottoalgoritmo

Questo procedimento incrementale, detto top-down, è spesso utile per progettare un buon algoritmo

Sotto-algoritmo Preleva

1. Cerca la scheda del libro richiesto
2. Segnati scaffale e posizione $\langle \mathbf{s}, \mathbf{p} \rangle$
3. Vai allo scaffale \mathbf{s}
4. Preleva il libro alla posizione \mathbf{p}
5. Compila la "scheda prestito"

Il passo Cerca non è ancora sufficientemente dettagliato

Sotto-algoritmo Cerca

1. Leggi la prima scheda nello schedario
2. **Se** titolo e autore/i corrispondono a quelli cercati **allora** concludi la ricerca con esito positivo, **altrimenti** passa la scheda successiva
3. **Se** le schede sono esaurite **allora** concludi la ricerca con esito negativo, **altrimenti ricomincia** dal punto 2

L'algoritmo può essere ulteriormente migliorato, rendendo la ricerca più furba!

Questo procedimento per raffinamenti incrementali è un altro approccio molto utile per la progettazione di buoni algoritmi.

Sotto-algoritmo Cerca (2)

1. Leggi la scheda centrale nello schedario
2. **Se** titolo e autore/i corrispondono a quelli cercati **allora** concludi la ricerca con esito positivo, **altrimenti**
 - I. **Se** la scheda cercata segue in ordine alfabetico quella prese in esame **allora ricomincia** la ricerca dal punto 1 nella seconda metà dello schedario
 - II. **Altrimenti ricomincia** la ricerca dal punto 1 nella prima metà dello schedario
3. **Se** le schede sono esaurite **allora** concludi la ricerca con esito negativo

Questo algoritmo sfrutta l'ordine alfabetico dello schedario per effettuare le ricerche più velocemente

Dal problema alla soluzione automatica

- ❑ Ci occuperemo di problemi che riguardano la gestione e l'elaborazione dell'informazione
- ❑ Vedremo come passare dalla **specifica** di un problema alla sua **soluzione automatica** attraverso l'uso di un calcolatore
 - ▶ La **specifica** è una descrizione semi-formale del problema
 - ▶ Affinché un **algoritmo**, che risolve il problema dato, sia eseguibile dal calcolatore dovrà essere definito in un **linguaggio comprensibile** al calcolatore stesso

Dalla specifica all'algoritmo

- ❑ Il processo che porta dalla specifica di un problema ad un algoritmo che lo risolve non è automatico e non è facile da formalizzare
 - ▶ La specifica spesso può essere poco chiara o ambigua
 - ▶ La scrittura di un algoritmo richiede uno sforzo creativo
- ❑ Come si impara a progettare un algoritmo?
 - ▶ Utilizzare un approccio incrementale
 - ▶ Realizzarlo per raffinamenti successivi
 - ▶ Fare molta pratica
- ❑ Qualità
 - ▶ Correttezza: risolve il problema e prende in considerazione tutti i casi possibili
 - ▶ Efficienza: usa con parsimonia le risorse (es. tempo)
- ❑ La correttezza è fondamentale ma difficile da verificare, l'efficienza è desiderabile e facile da misurare

Come si formalizza un algoritmo?

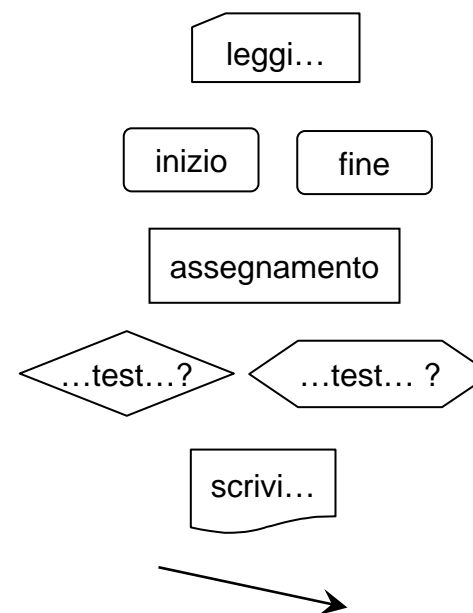
- ❑ Una buon processo di progettazione si conclude con la definizione precisa e concisa dell'algoritmo ideato
- ❑ Alcuni linguaggi semi-formali spesso usati

▶ Pseudo-codice

`se A > 0 allora A = A + 1 altrimenti A = 0`

▶ Diagrammi di flusso (o schemi a blocchi)

- Blocco di input dati
- Blocchi di inizio/fine dell'esecuzione
- Blocco esecutivo
- Blocco condizionale
- Blocco di output dati
- Flusso di controllo delle operazioni

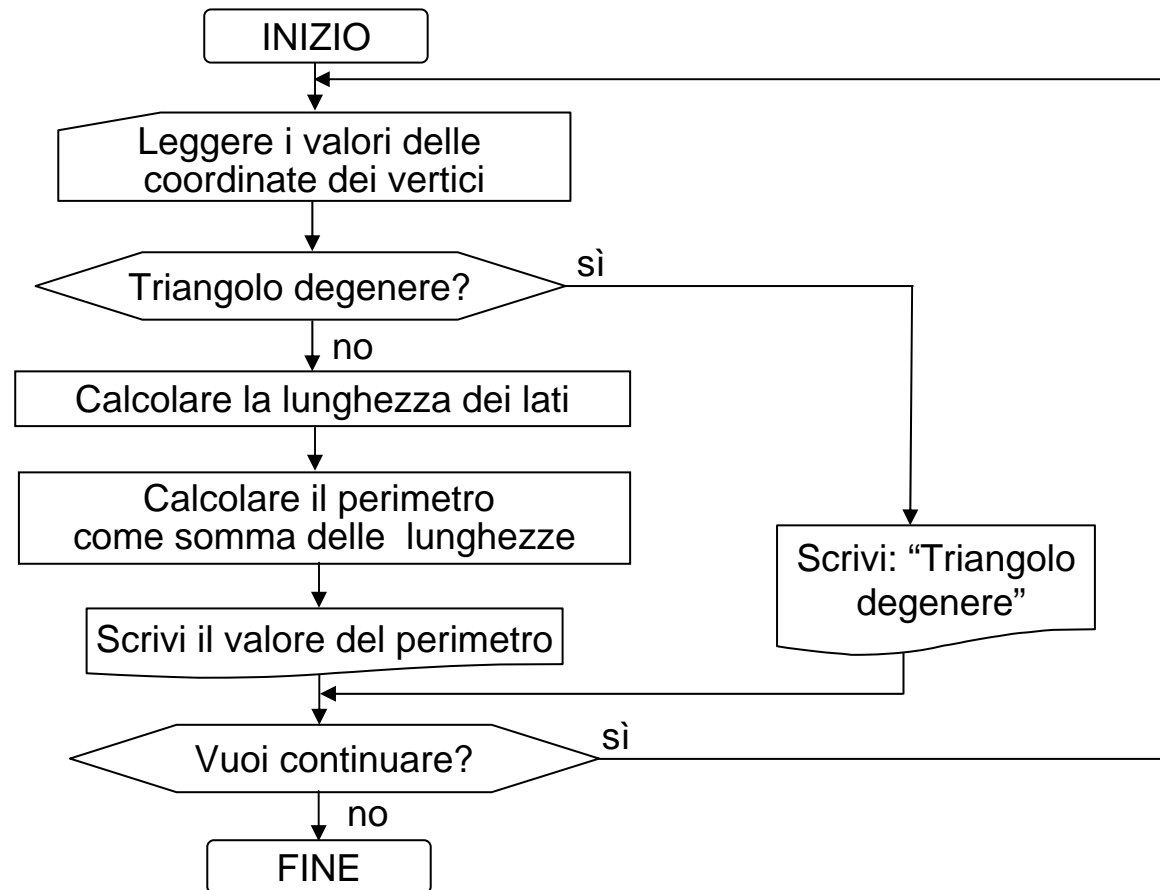


Esempio: M.C.D. di due naturali positivi

1. **Leggi** N ed M
2. MIN = il minimo tra N ed M
3. $X = 1$
4. $MCD = 1$
5. **Fintantoché** $X < MIN$
 - I. $X = X + 1$
 - II. **se** X divide sia N che M , **allora** $MCD = X$
6. **Scrivi** MCD

Esempio: perimetro di un triangolo

- Date le coordinate di tre punti, riconoscere se sono i vertici di un triangolo non degenere, e nel caso calcolarne il perimetro

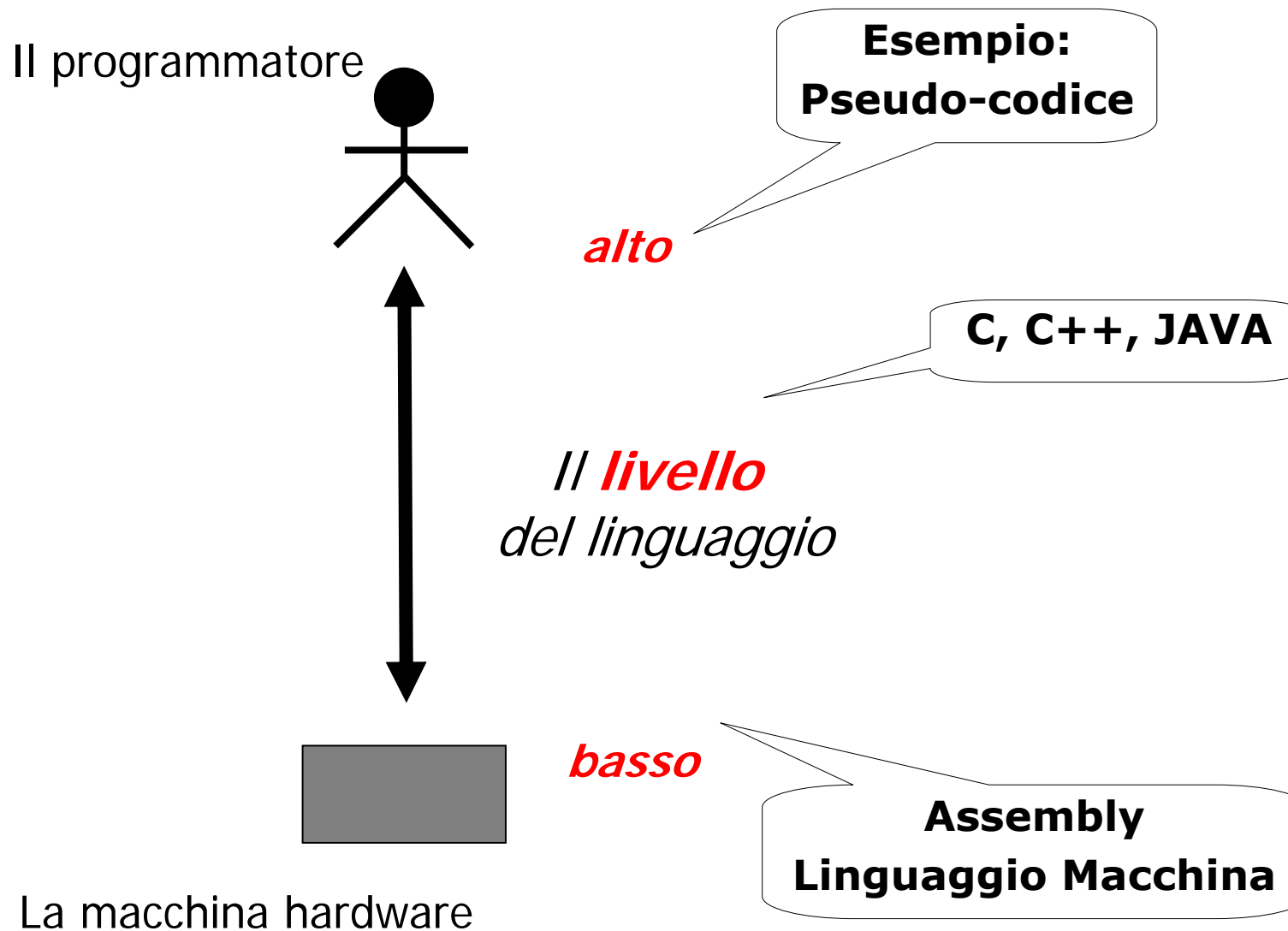


Dall'algoritmo alla soluzione

- ❑ Come deve essere formalizzato un algoritmo affinché sia comprensibile al calcolatore?
- ❑ Il calcolatore è in grado di eseguire algoritmi (**programmi**) definiti in **linguaggio macchina**

0	0100	00000000	(READ)	
1	0010	00100000	(STORE)	32
2	0001	00000000	(LOAD=)	0
3	0010	00100001	(STORE)	33
4	0000	00100000	(LOAD)	32
5	1100	00001101	(BEQ)	13
6	0100	00000000	(READ)	
7	0110	00100001	(ADD)	33
8	0010	00100001	(STORE)	33
9	0000	00100000	(LOAD)	32

Livello di un linguaggio di programmazione



Sintassi e Semantica

- ❑ La sintassi definisce come si scrive il programma (forma e struttura)
 - ▶ Esempio: `<variabile> = <espressione>`
- ❑ La semantica definisce come si interpretano le istruzioni contenute nel programma (significato)
 - ▶ Esempio:
`<variabile> = <espressione>`
"calcola il valore dell'espressione e assegna al contenuto della variabile il valore calcolato"
- ❑ Un programma sintatticamente corretto non è necessariamente corretto!
- ❑ La sintassi può essere verificata automaticamente dal compilatore, la semantica no!

Compilatori e Interpreti

- ❑ Come rendo “comprensibili” al calcolatore i programmi scritti in linguaggi formali di alto livello (ad es. C, C++, Java) ?
- ❑ Due sono gli approcci maggiormente usati:
 - ▶ Utilizzando un **compilatore**, un programma che traduce i programmi di alto livello in codice macchina
 - ▶ Utilizzando un **interprete**, un programma che interpreta direttamente le istruzioni di alto livello e le esegue
- ❑ Nei linguaggi compilati la **catena di programmazione** si compone di 5 fasi:
 - ▶ Scrittura
 - ▶ Traduzione
 - ▶ Collegamento
 - ▶ Caricamento
 - ▶ Esecuzione



1. Scrittura

- ❑ Il programma, costituito da una sequenza di caratteri, viene **composto e modificato** usando un qualsiasi editor
- ❑ Così otteniamo un **codice sorgente** memorizzato in memoria di massa in un file di testo (es. XYZ.c)

2. Traduzione

- ❑ Il compilatore si occupa della traduzione dal linguaggio di alto livello al linguaggio macchina
- ❑ Durante questa fase si riconoscono i simboli, le parole e i costrutti del linguaggio:
 - ▶ eventuali messaggi diagnostici segnalano errori di sintassi
- ❑ Viene generato il codice macchina in forma binaria : a partire dal **codice sorgente** si genera il **codice oggetto**, cioè in un file **binario**

3. Collegamento (linking)

- ❑ Il collegatore (linker) deve collegare fra loro il file oggetto ed altre librerie utilizzate (es. librerie di I/O)
- ❑ Si rendono globalmente coerenti i riferimenti agli indirizzi dei vari elementi collegati
- ❑ Si genera un **programma eseguibile**, un file binario che contiene il codice macchina del programma eseguibile completo, di nome XYZ.exe
- ❑ Messaggi di errore possono essere dovuti ad errori nel citare i nomi delle funzionalità di librerie esterne da collegare
- ❑ Il programma sarà effettivamente eseguibile solo dopo che il contenuto del file sarà stato caricato nella memoria di lavoro (centrale) del calcolatore

4. Caricamento (loading)

- ❑ Il caricatore (**loader**) individua una porzione libera della memoria di lavoro e vi copia il contenuto del programma eseguibile
 - ▶ Eventuali messaggi rivolti all'utente possono segnalare che non c'è abbastanza spazio in memoria

5. Esecuzione

- ❑ Per eseguire il programma occorre fornire in ingresso i dati richiesti e in uscita riceveremo i risultati (su video o file o stampante)
- ❑ Durante l'esecuzione possono verificarsi degli errori (detti "errori di run-time"), quali:
 - ▶ calcoli con risultati scorretti (per esempio un overflow)
 - ▶ calcoli impossibili (divisioni per zero, logaritmo di un numero negativo, radice quadrata di un numero negativo,....)
 - ▶ errori nella concezione dell'algoritmo (l'algoritmo non risolve il problema dato)
- ❑ Tutti gli esempi citati si riferiscono ai cosiddetti *errori semantici*