# Introduzione agli algoritmi

Fondamenti di Informatica, AA 2022/23

Luca Cassano

luca.cassano@polimi.it

# Cos’è l’informatica?

*Studio sistematico degli **algoritmi** che descrivono e trasformano l’informazione:*

la loro teoria,

analisi,

progetto,

efficienza,

realizzazione,

applicazione.

[da Association for Computing Machinery (ACM)]

# Algoritmo

***Sequenza** di **istruzioni**, definite con **precisione**, che portano alla **realizzazione di un compito***

Le istruzioni devono:

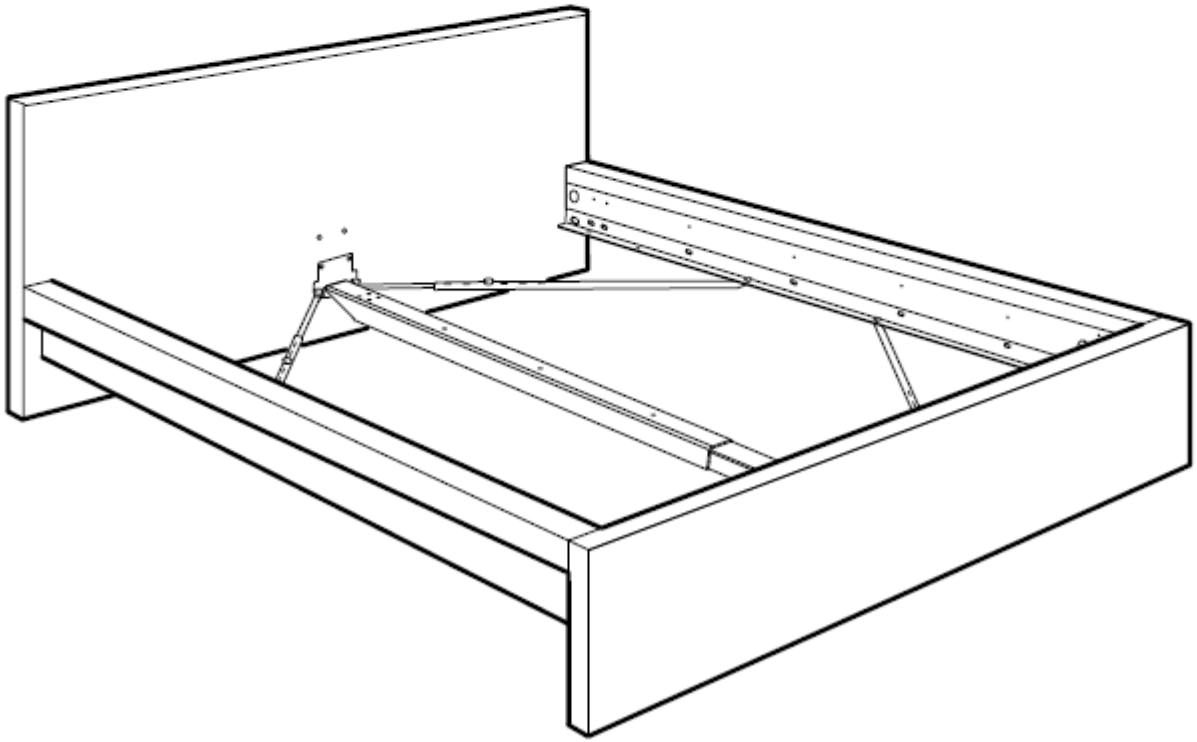
- essere **comprensibili** senza ambiguità
- essere **eseguibili** da uno strumento automatico: l'esecutore
- portare a realizzare un compito in **tempo finito** (devono contenere un numero finito di passi, ciascuno eseguibile in tempo finito)

Non è necessario un calcolatore per parlare di informatica!

# Algoritmo

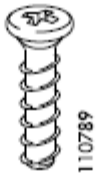
…….ora vedermo un esempio di algoritmo in cui vi sarà capitato di essere esecutori….

# MALM

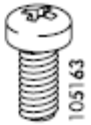


IKEA®
Design and Quality
IKEA of Sweden


12x



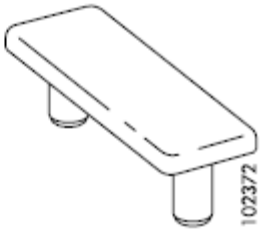

20x/22x




8x


12x


8x

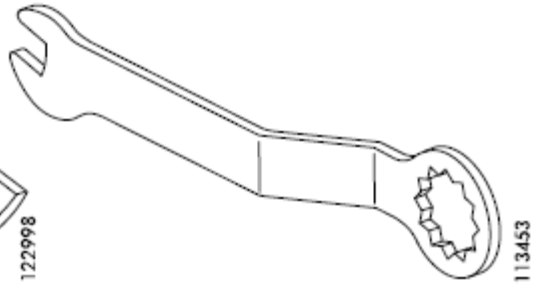



6x

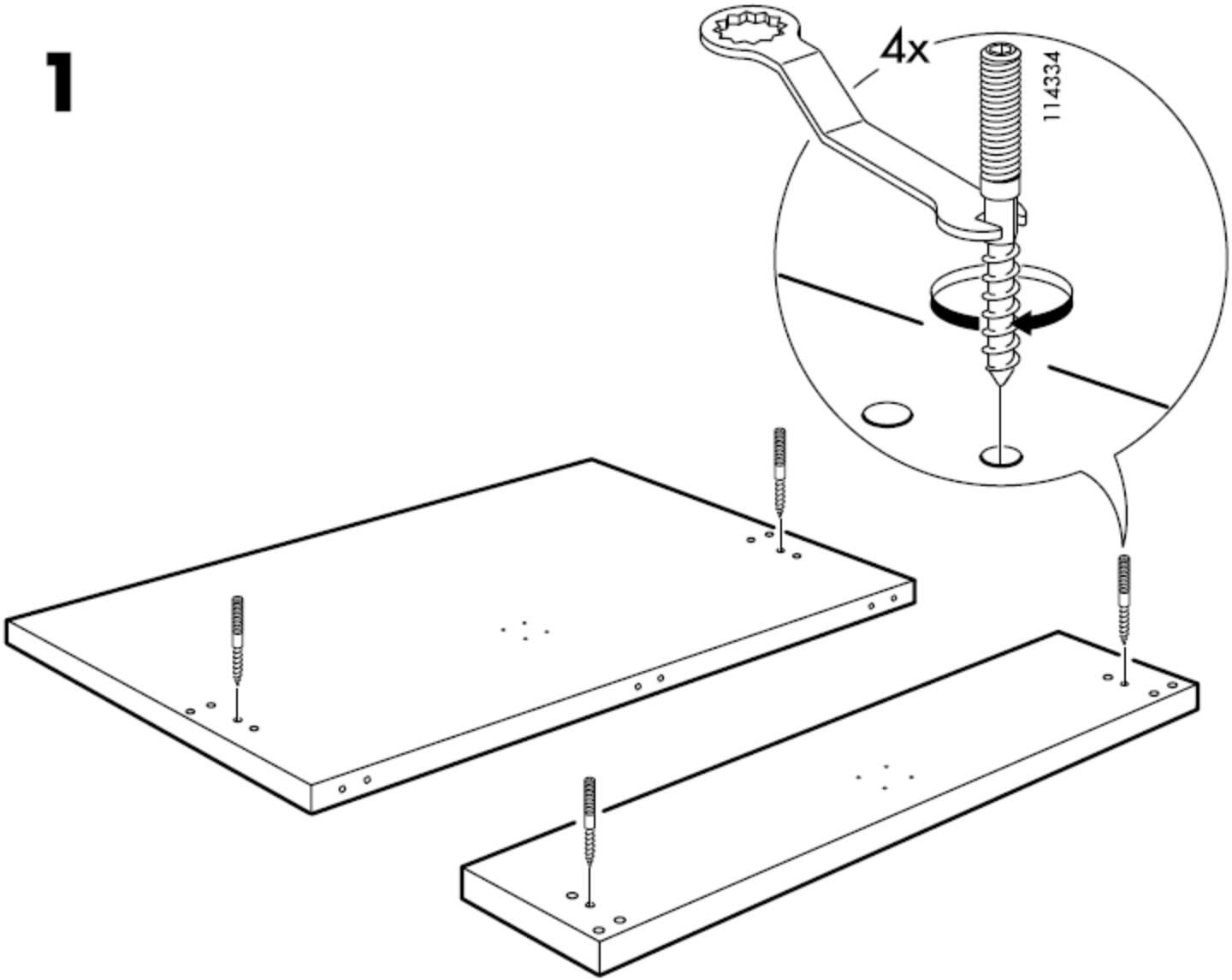

4x

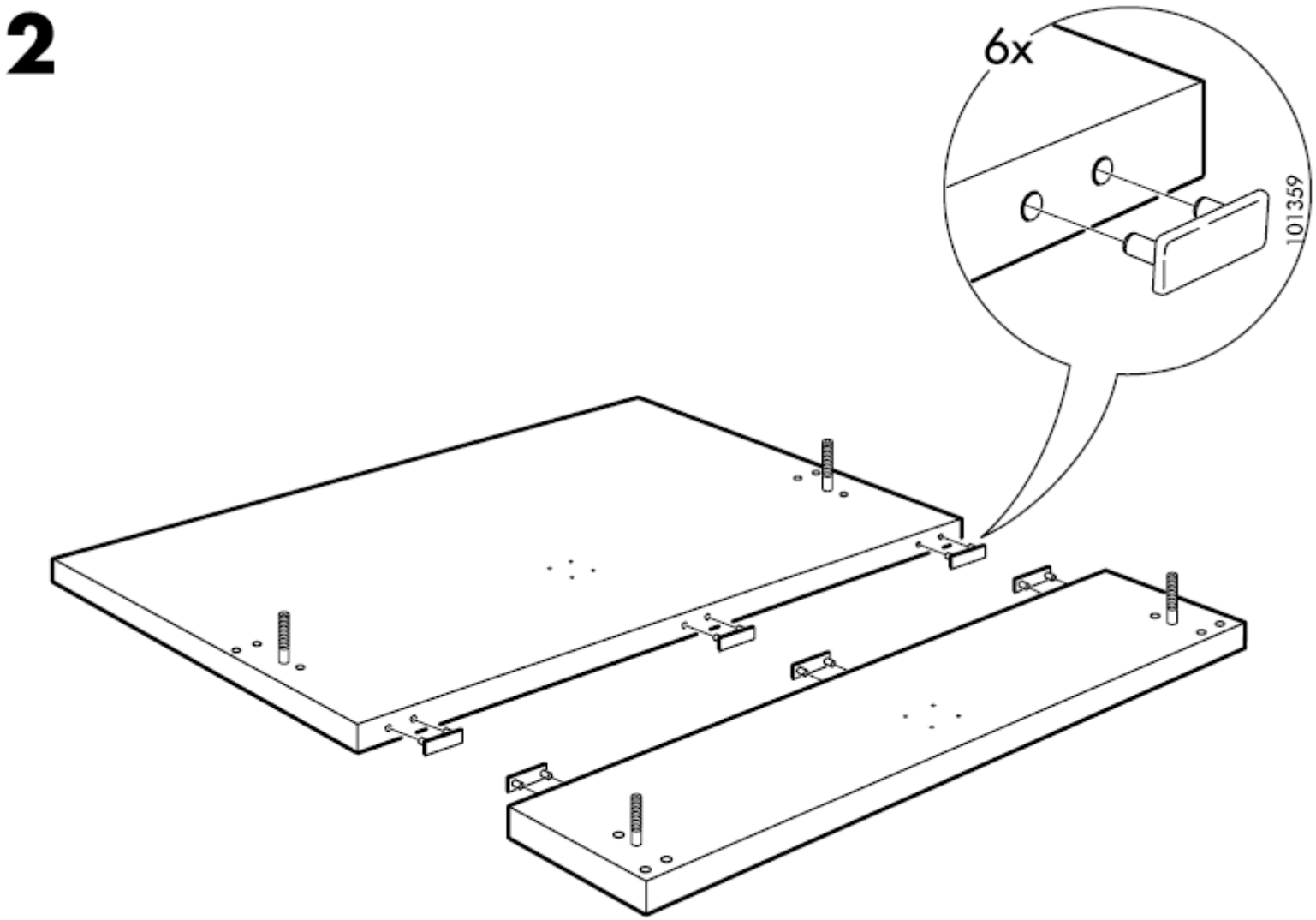

4x


4x



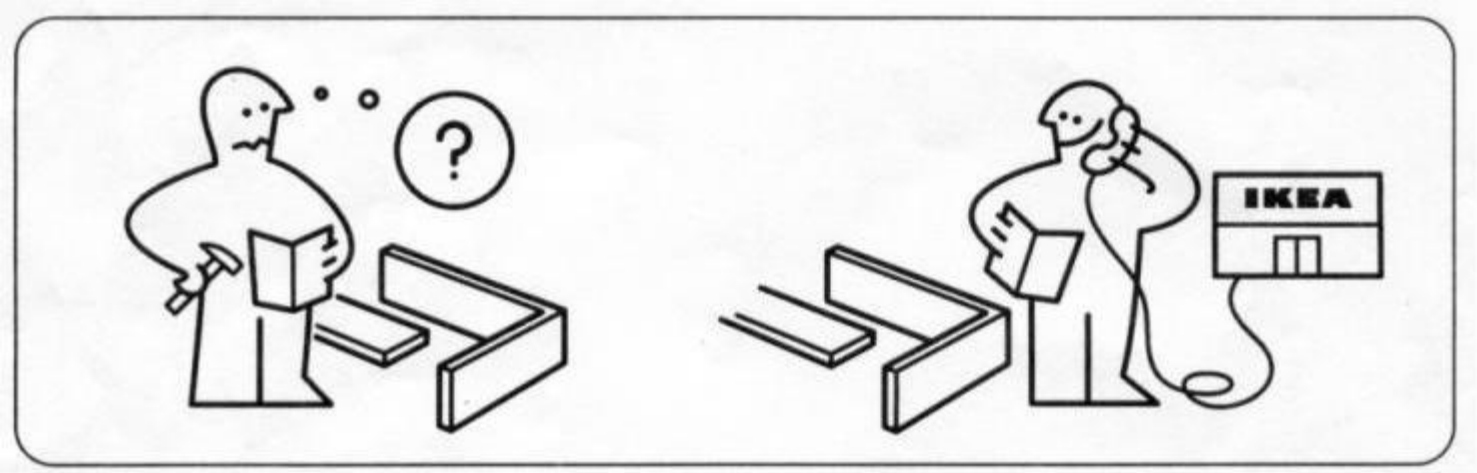

1x


1
4x
114334

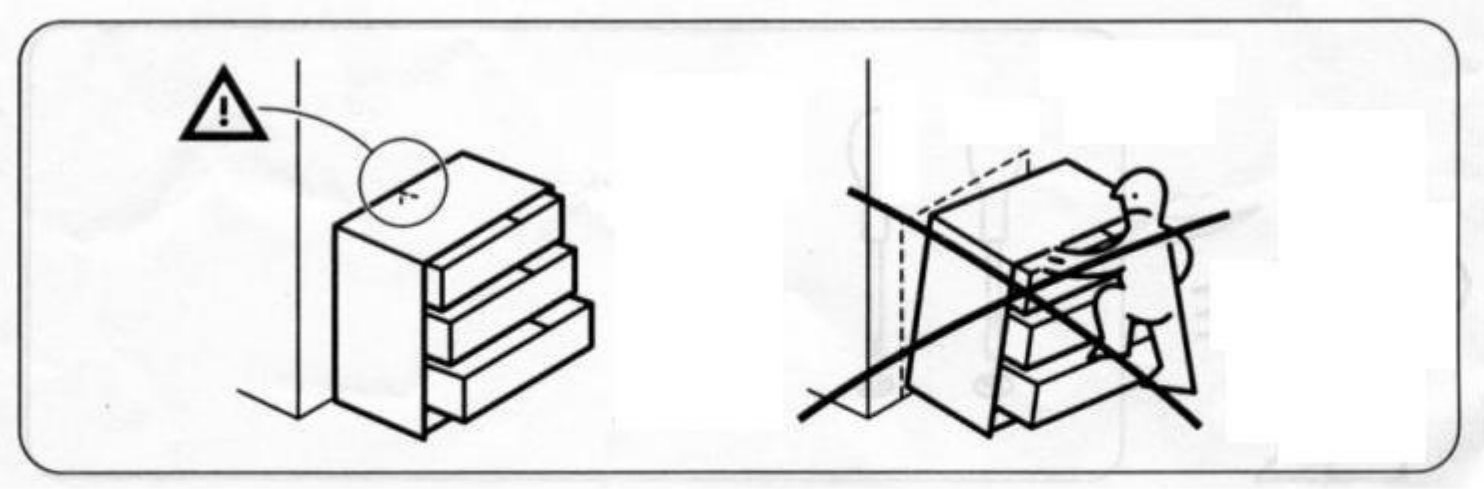

2
6x
101359

# Il Linguaggio «IKEA»

- Le istruzioni IKEA sono fatte per esecutori intelligenti (nello specifico…noi) l'interpretazione dei disegni richiede diverse capacità



# Il Linguaggio «IKEA»



- Quando l'esecutore è meno intelligente occorre esprimere le istruzioni in un linguaggio più preciso

# Definiamo un nostro linguaggio per gli algoritmi!

Svilupperemo i prossimi algoritmi in italiano (utilizzando un linguaggio molto essenziale).

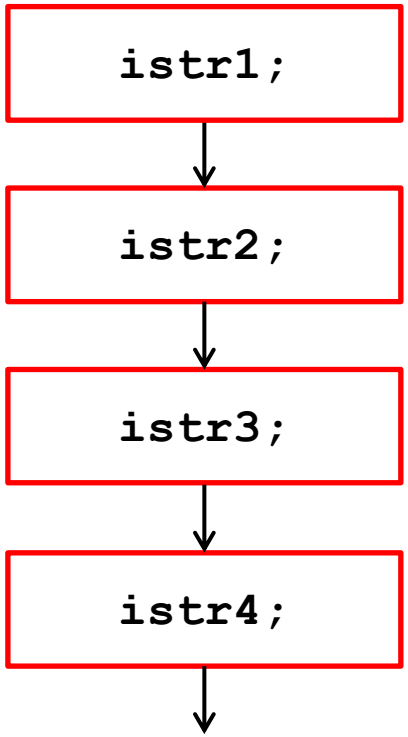
In particolare, il linguaggio sarà caratterizzato da:

- Sequenzialità delle istruzioni
- Costrutto condizionale
- Costrutto iterativo

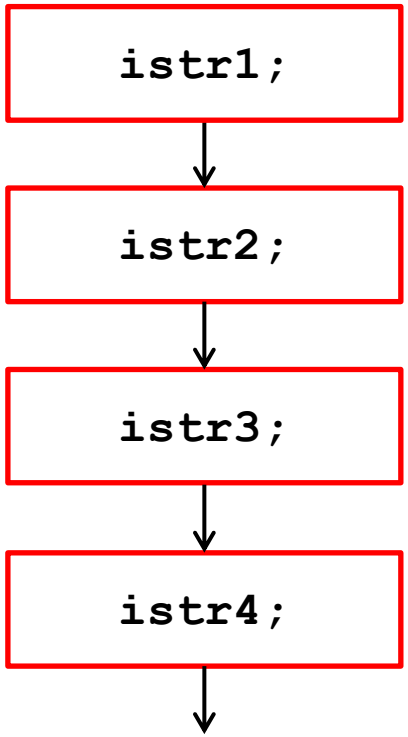
Inoltre, potremo utilizzare “foglietti” dove scrivere (registrare) alcuni valori

# La Sequenzialità

# La sequenzialità



# La sequenzialità




```
istr1;
istr2;
istr3;
istr4;
...
```

Le istruzioni vengono eseguite dalla prima all'ultima.

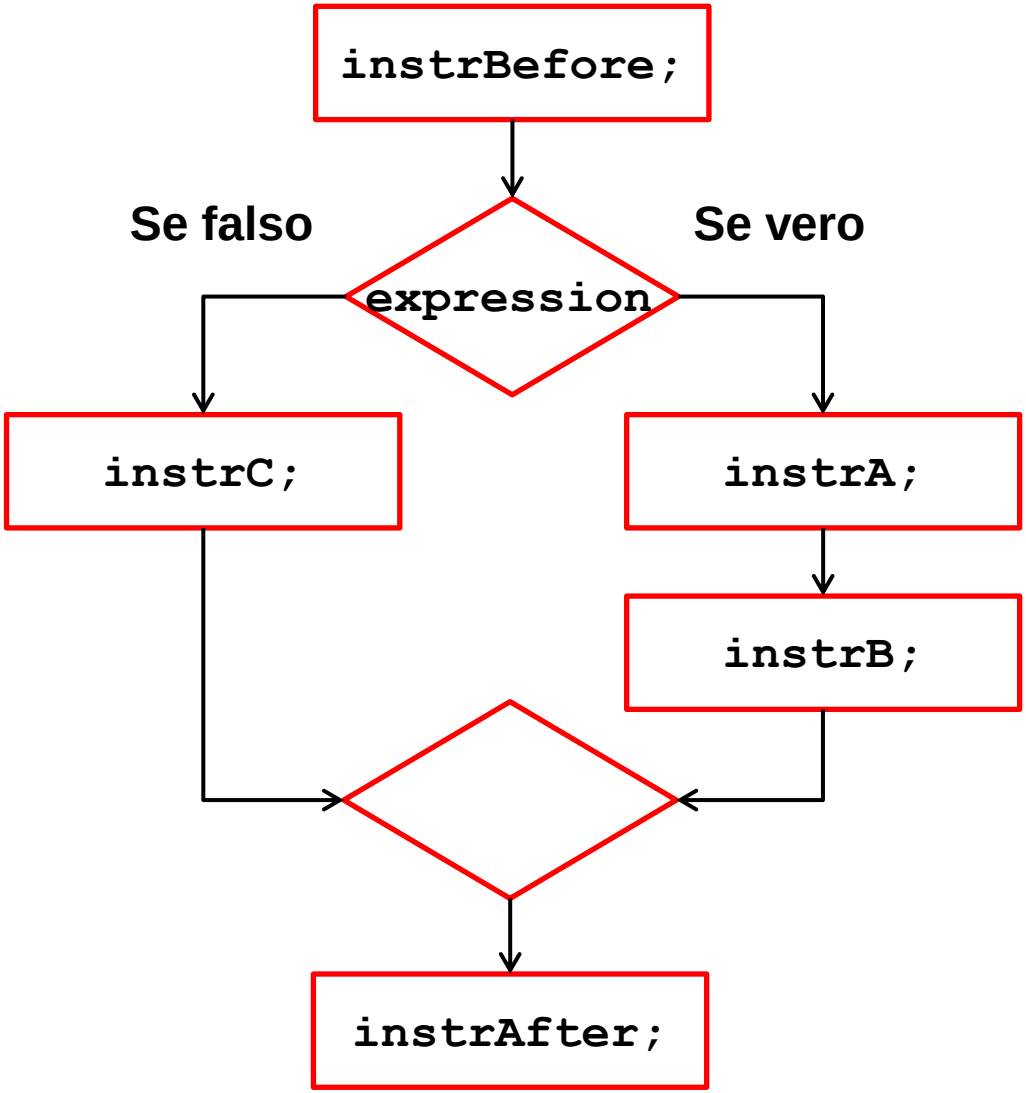
Terminata la *i-sima* istruzione, si esegue la *(i+1)-sima*

# Esempio: algoritmo per andare in università

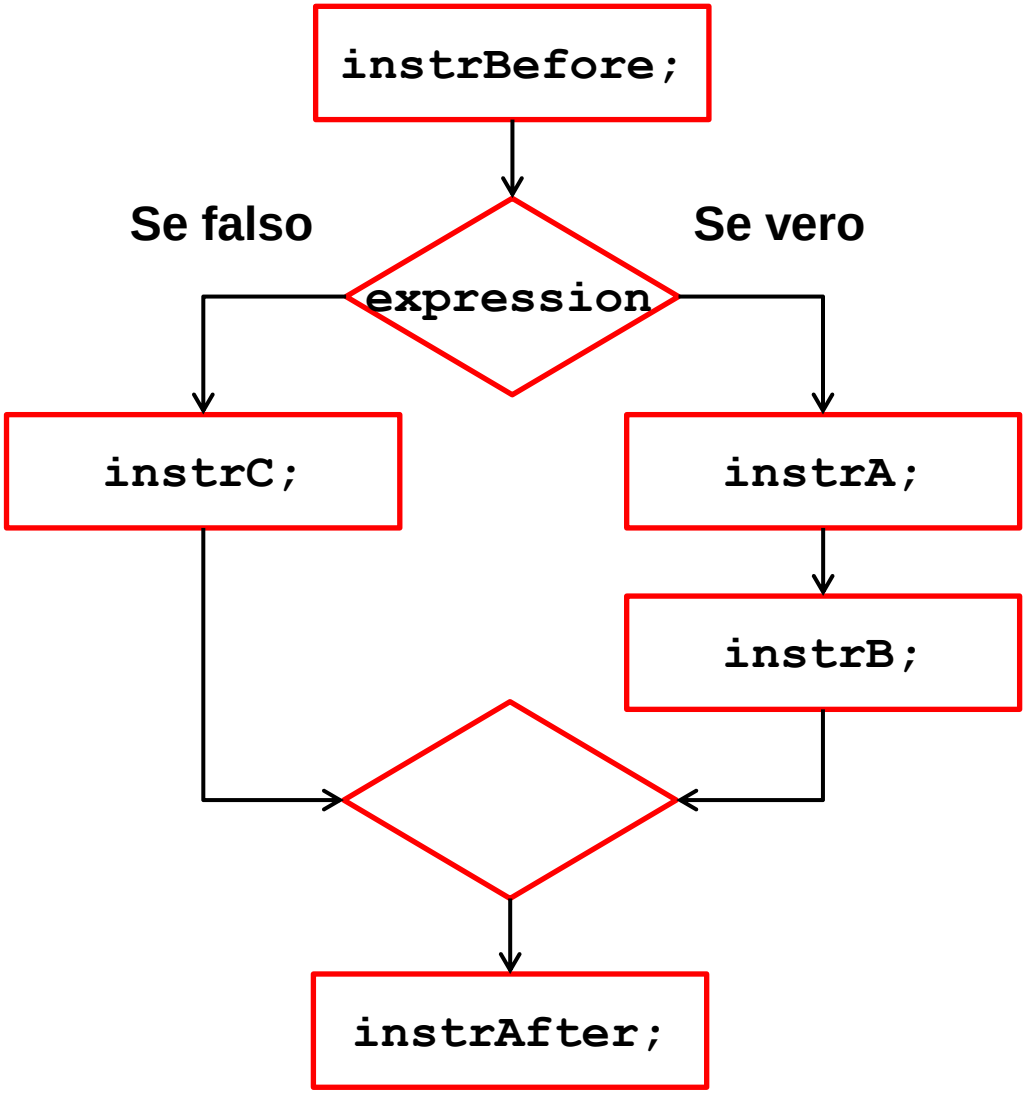
- Mi alzo quando suona la sveglia
- Mi dirigo verso la cucina
- Mangio un cornetto e bevo un caffè
- Mi lavo
- Mi vesto
- Esco
- Corro

# Costrutto Condizionale

# Costrutto Condizionale:



# Costrutto Condizionale:




Dopo aver eseguito **`instrBefore`** si valuta **`expression`**

Se **`expression`** è vera eseguo il ramo di istruzioni contenente **`instrA;`** e **`instrB`** **(ramo *then*)**

Altrimenti eseguo **`instrC`** (**ramo *else***)

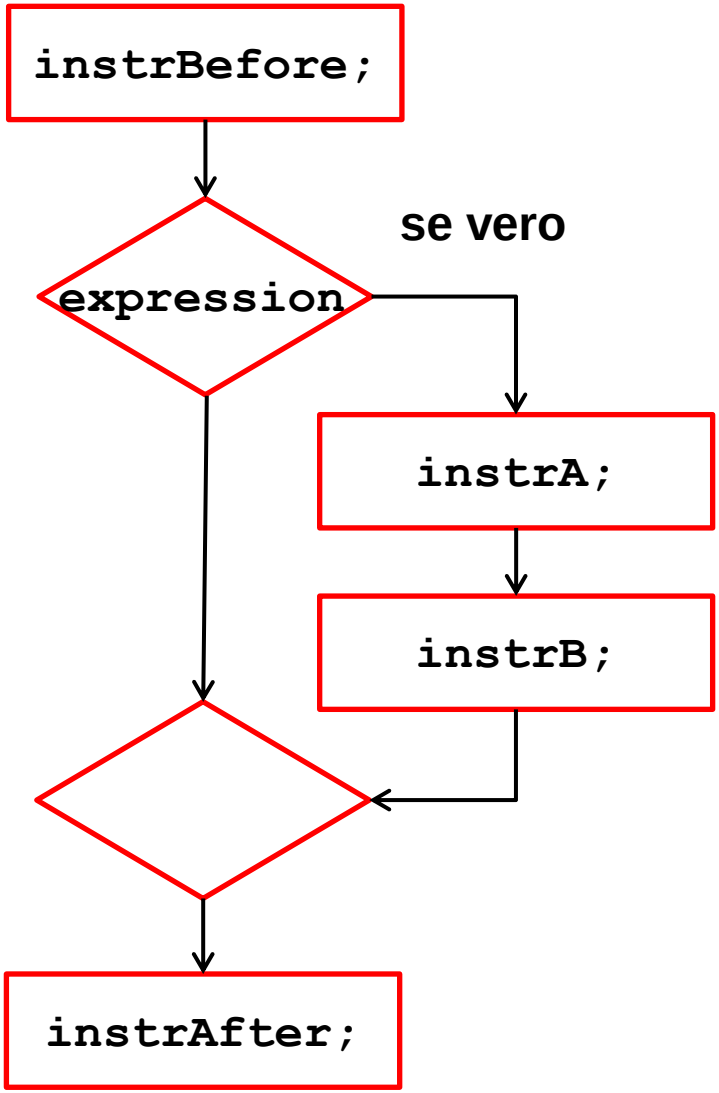
Poi eseguo **`instrAfter`**

# Esempio: algoritmo per andare in Università

- Mi alzo quando suona la sveglia
- Mi dirigo verso la cucina
- Mangio e bevo un caffè
- Mi lavo e mi vesto
- **Se** devo mangiare fuori
  - prendo il pranzo

  **Altrimenti**
  - prendo uno snack per metà mattina
- Esco
- Corro

# Costrutto Condizionale:



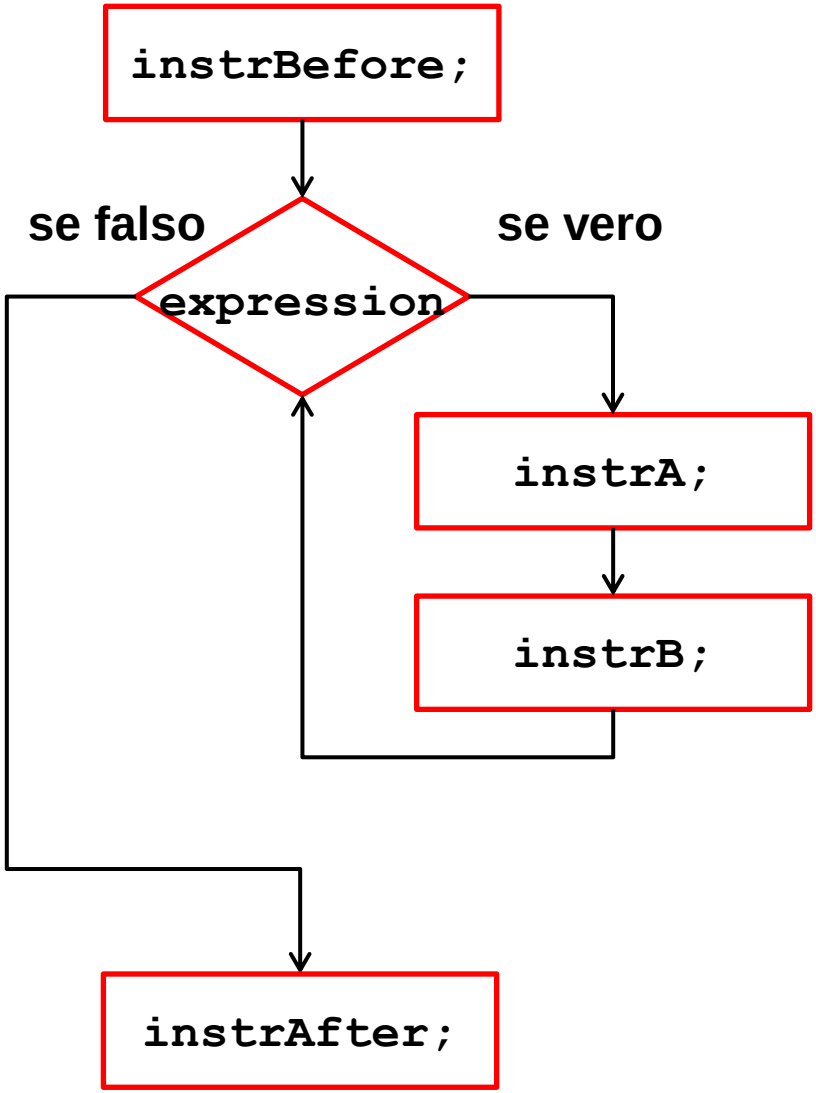

Non è obbligatorio prevedere azioni specifiche nel caso in cui **`expression`** fosse falsa

# Esempio: algoritmo per andare in Università

- Mi alzo quando suona la sveglia
- Mi dirigo verso la cucina
- Mangio e bevo un caffè
- Mi lavo e mi vesto
- **Se** c'è il laboratorio di informatica
  - Prendo il laptop
- Esco
- Corro

# Costrutto Iterativo

# Costrutto Iterativo




Se **`expression`** è vera eseguo il ramo di istruzioni contenente **`instrA;`** e **`instrB;`** (corpo del ciclo)

Al termine valuta nuovamente **`expression.`**

Se **`expression`** è falsa, prosegui oltre, altrimenti esegui le istruzioni nel corpo del ciclo

# Esempio: algoritmo per andare in Università

- Mi alzo quando suona la sveglia
- Mi dirigo verso la cucina
- Mangio e bevo un caffè
- Mi lavo e mi vesto
- **Se** c'è il laboratorio di informatica
  - Prendo il laptop
- Vado in stazione
- **Ripeti**: "guarda il treno in arrivo: non ferma a Lambrate?"
  - se vero, attendi il possimo treno
- Sali sul treno
- Arrivi a lezione, wow

# Esempi di Algoritmi

# Altri Esempi:

- Algoritmo per invertire il contenuto di due bicchieri A e B, supponendo di avere un terzo bicchiere C.

- Algoritmo per trovare il prodotto più economico nella corsia del supermercato, passando per la corsia una sola volta.

- Algoritmo per controllare che la maggioranza degli studenti abbia capito gli esercizi sopra.

# Esempio: invertire il contenuto di A e B

1. Prendi un terzo bicchiere C
2. Rovescia il contenuto del bicchiere A nel bicchiere C
3. Rovescia il contenuto di B in A
4. Rovescia il contenuto di C in B

# Esempio: invertire il contenuto di A e B

1. Prendi un terzo bicchiere C
2. Rovescia il contenuto del bicchiere A nel bicchiere C
3. Rovescia il contenuto di B in A
4. Rovescia il contenuto di C in B

**Algoritmo per scambiare i valori di due variabili A e B**

# Altri Esempi:

- Algoritmo per invertire il contenuto di due bicchieri, supponendo di avere un terzo bicchiere.

- Algoritmo per trovare il prodotto più economico nella corsia del supermercato, passando per la corsia una sola volta.

- Algoritmo per controllare che la maggioranza degli studenti abbia capito gli esercizi sopra.

# Esempio: ricerca del prodotto migliore

1. Prendi in mano il primo prodotto: assumi che sia il più economico
2. Procedi fino al prossimo prodotto
3. Confrontalo con quello che hai in mano
4. **Se** il prodotto davanti a te è più economico: abbandona il prodotto che hai in mano e prendi quello sullo scaffale
5. **Ripeti** i passi **2 - 4 fino a** raggiungere la fine della corsia
6. Hai in mano il prodotto più economico.

# Esempio: ricerca del prodotto migliore

1. Prendi in mano il primo prodotto: assumi che sia il più economico
2. Procedi fino al prossimo prodotto
3. Confrontalo con quello che hai in mano
4. **Se** il prodotto davanti a te è più economico: abbandona il prodotto che hai in mano e prendi quello sullo scaffale
5. **Ripeti** i passi **2 - 4 fino a** raggiungere la fine della corsia
6. Hai in mano il prodotto più economico.

**Algoritmo per trovare il minimo/massimo di una sequenza numerica**

# Altri Esempi:

- Algoritmo per invertire il contenuto di due bicchieri, supponendo di avere un terzo bicchiere.

- Algoritmo per trovare il prodotto più economico nella corsia del supermercato, passando per la corsia una sola volta.

- Algoritmo per controllare che la maggioranza degli studenti abbia capito gli esercizi sopra.

# Esempio: assicurarsi che il 50% abbia capito

1. Prendi due **fogli**, uno per contare chi non ha capito (N) ed uno per contare tutti gli studenti (T)
2. Avvicinati al primo studente
3. Chiedi se ha capito
4. **Se** risponde no, metti un segno su N
5. Metti un segno su T
6. Passa al prossimo studente
7. **Ripeti** i passi **3 – 6 fino** all'ultimo studente
8. Conta i segni su N e su T
9. **Se** il numero di N è maggiore della metà di T, la condizione è non verificata

# Esempio: assicurarsi che tutti abbiano capito

1. Prendi un solo **foglio**
2. Avvicinati al primo studente
3. Chiedi se ha capito
4. **Se** risponde no, metti un segno sul foglio
5. Passa al prossimo studente
6. **Ripeti** i passi **3 – 5 fino** all'ultimo studente
7. Se il foglio non ha segni, tutti hanno capito.

# Esempio: assicurarsi che tutti abbiano capito

1. Prendi un solo **foglio**
2. Avvicinati al primo studente
3. Chiedi se ha capito
4. **Se** risponde no, metti un segno sul foglio
5. Passa al prossimo studente
6. **Ripeti** i passi **3 – 5 fino** all'ultimo studente **o fino** a quando uno studente dice di non aver capito
7. Se il foglio non ha segni, tutti hanno capito.

Variante più efficiente

# Esempio: assicurarsi che tutti abbiano capito

1. Prendi un solo **foglio**
2. Avvicinati al primo studente
3. Chiedi se ha capito
4. **Se** risponde no, metti un segno sul foglio
5. Passa al prossimo studente
6. **Ripeti** i passi **3 – 5 fino** all'ultimo studente **o fino** a quando uno studente dice di non aver capito
7. Se il foglio non ha segni, tutti hanno capito.

**Algoritmo per verificare che una condizione sia soddisfatta da tutti gli elementi di un array**

# I Programmi

Dall’algoritmo al codice

# Il Calcolatore

Il **calcolatore** è un potente **esecutore di algoritmi**

- È rapido: permette di gestire quantità di informazioni altrimenti intrattabili
- È preciso: non commette mai errori
- Non ha spirito critico

I **programmi** sono **algoritmi codificati** in **linguaggi** comprensibili dal calcolatore.

# Il Programmatore (voi)

Compito dell'informatico (e di chiunque programmi) è:

1. **Ideare l'algoritmo**: conoscere la soluzione del problema e esprimerla in rigorosi passi
2. **Codificare l'algoritmo in un programma**: conoscere il linguaggio dell'esecutore (linguaggio di programmazione)

La parte più difficile è spesso la prima.

- Prima dobbiamo aver chiaro «cosa far fare alla macchina», poi traduciamolo correttamente.

# Proprietà essenziali dei programmi (algoritmi)

**Correttezza**: l'algoritmo risolve il compito senza errori o difetti.

**Efficienza**: l'algoritmo usa risorse in modo minimale (o almeno ragionevole)

- **Diversi criteri** quindi per definire l'efficienza a seconda delle risorse
  - tempo,
  - memoria,
  - numero di letture/scritture da disco

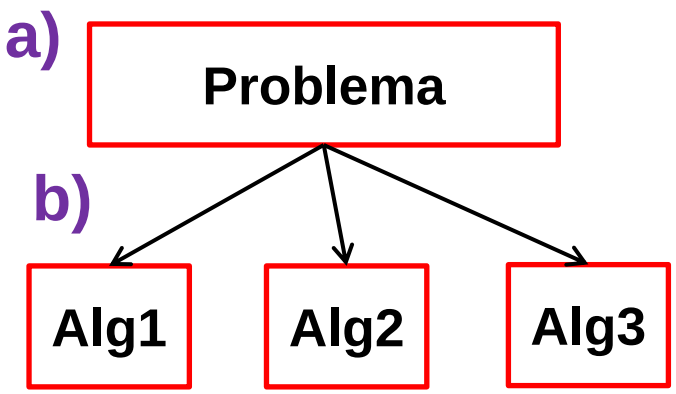
# Riepilogando: Come Procedere

a) Partirete dall'analisi di un problema

# Riepilogando: Come Procedere

a) Partirete dall'analisi di un problema

b) Individuerete uno o più algoritmi che risolvono il problema in modo più o meno efficiente

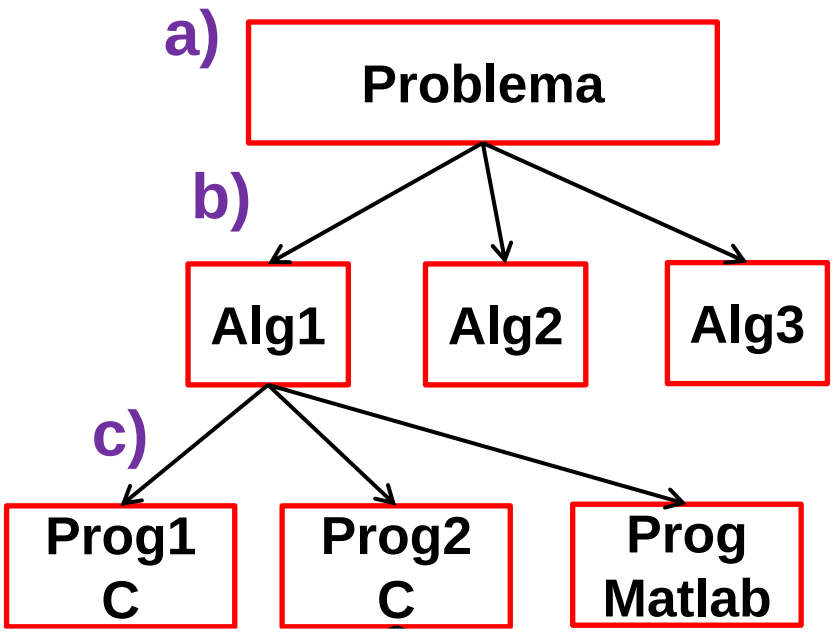


# Riepilogando: Come Procedere

a) Partirete dall'analisi di un problema

b) Individuerete uno o più algoritmi che risolvono il problema in modo più o meno efficiente

c) Codificherete un algoritmo in un linguaggio di programmazione,



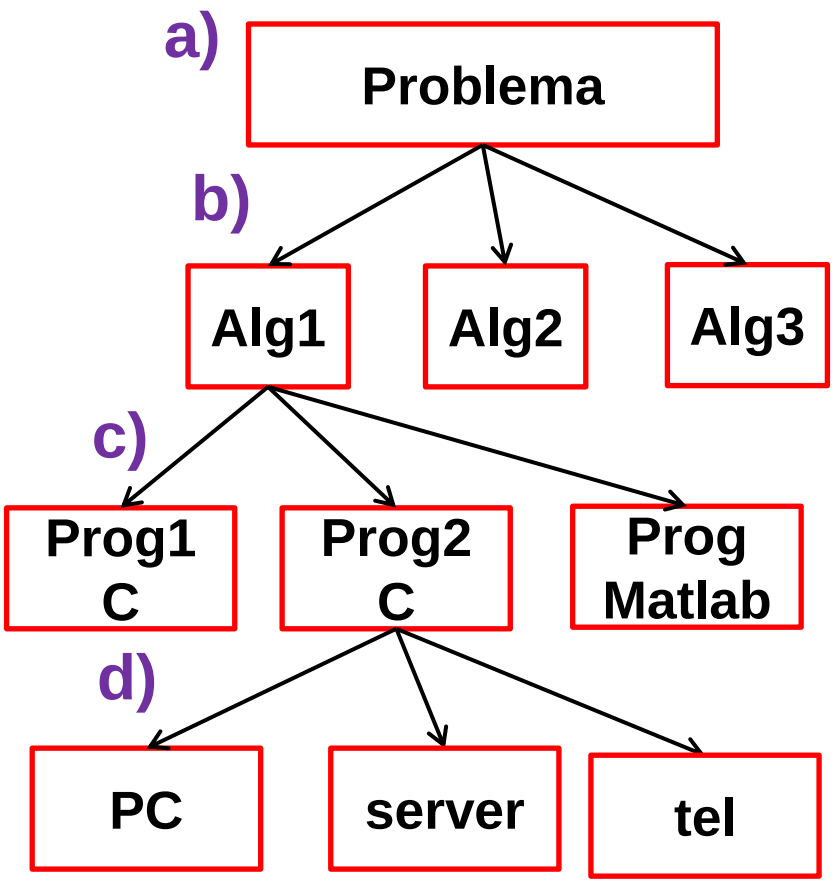
# Riepilogando: Come Procedere

a) Partirete dall'analisi di un problema

b) Individuerete uno o più algoritmi che risolvono il problema in modo più o meno efficiente

c) Codificherete un algoritmo in un linguaggio di programmazione

d) Il programma potrà essere «utilizzato» su diverse macchine



# Riepilogando: Cosa Fare

a) Dovrete (capire e) definire correttamente il problema

b) **Ricavare l'algoritmo** è la parte più **difficile**. Richiede, oltre a esperienza, competenze specifiche, creatività e anche rigore perché occorre specificarlo in modo formale

c) La codifica è più semplice ma il programma deve essere efficiente e corretto

d) All'ultimo step pensa la macchina… vedremo poi la compilazione