# Introduzione al Corso

Informatica, AA 2021/2022

Francesco Trovò

https://trovo.faculty.polimi.it/

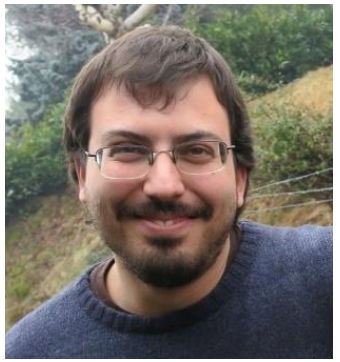
francesco1.trovo@polimi.it

# Ci presentiamo

# Chi siamo

**Francesco Trovò** (docente)

- Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano
- Homepage: https://trovo.faculty.polimi.it/
- E-mail francesco1.trovo@polimi.it
- Webex Room: https://politecnicomilano.webex.com/meet/francesco1.trovo
- Ricevimento studenti: online, su **appuntamento mandando un'e-mail**
- Ufficio: Edificio 21, Primo Piano, Ufficio 013, via Ponzio 34/5, Milano
- Tel 02 2399 3491



# Chi siamo

**Andrea  Mazzoleni** (esercitatore e responsabile di laboratorio)

- email: andrea1.mazzoleni@polimi.it

I materiali di esercitazioni e laboratorio saranno caricati sulla pagina del corso

Sito del corso

https://trovo.faculty.polimi.it/info_ica_2021_2022.html

# Il Corso

# Organizzazione

**Lezioni**: 28 ore

**Esercitazioni:** 24 ore

**Laboratori:** 16 ore

Lezioni in presenza: il Martedì dalle 9.00 alle 11:00 in aula B.1.5.

Lezioni online: il Venerdì dalle 14.30 alle 17:00

**Controllate sempre** il **calendario** del corso:

https://calendar.google.com/calendar/u/0/embed?src=nmpthod7eeo2pfnmi3s5b4vmqs@group.calendar.google.com&ctz=Europe/Rome

Dove troverete anche indicazioni su chi terrà ogni lezione ed esercitazione

# La Pagina del Corso

La pagina del corso è

https://trovo.faculty.polimi.it/info_ica_2021_2022.html

Troverete:

- Materiale didattico usato a lezione (queste slides sono da considerare **un supporto** allo studio)
- Alcuni temi d'esame
- Calendario del corso (lezioni, esercitazioni, laboratori)
- Avvisi, esiti esami/prove intermedie

# I Laboratori

Nei laboratori vi sarà richiesto di **sviluppare autonomamente** dei programmi

**Il laboratorio è molto utile per**

- prendere familiarità con l'ambiente di sviluppo
- consolidare la conoscenza dei linguaggi, dei  metodi e degli strumenti introdotti a lezione

A laboratorio potrete usare i PC dell'aula informatizzata

Vi saranno comunque date informazioni per installare Matlab sul vostro PC

# Lezioni ed Esercitazioni

How to…

# Modalità di Insegnamento

**Due consigli:**

- sfruttate al massimo la presenza (anche virtuale) di docenti ed esercitatori
- riguardate regolarmente almeno gli esercizi svolti

È molto gradita l'interazione docente-studente:

- farò domande: rispondete (sia da casa che dall'aula)!
- chiedete le cose che non vi sono chiare!

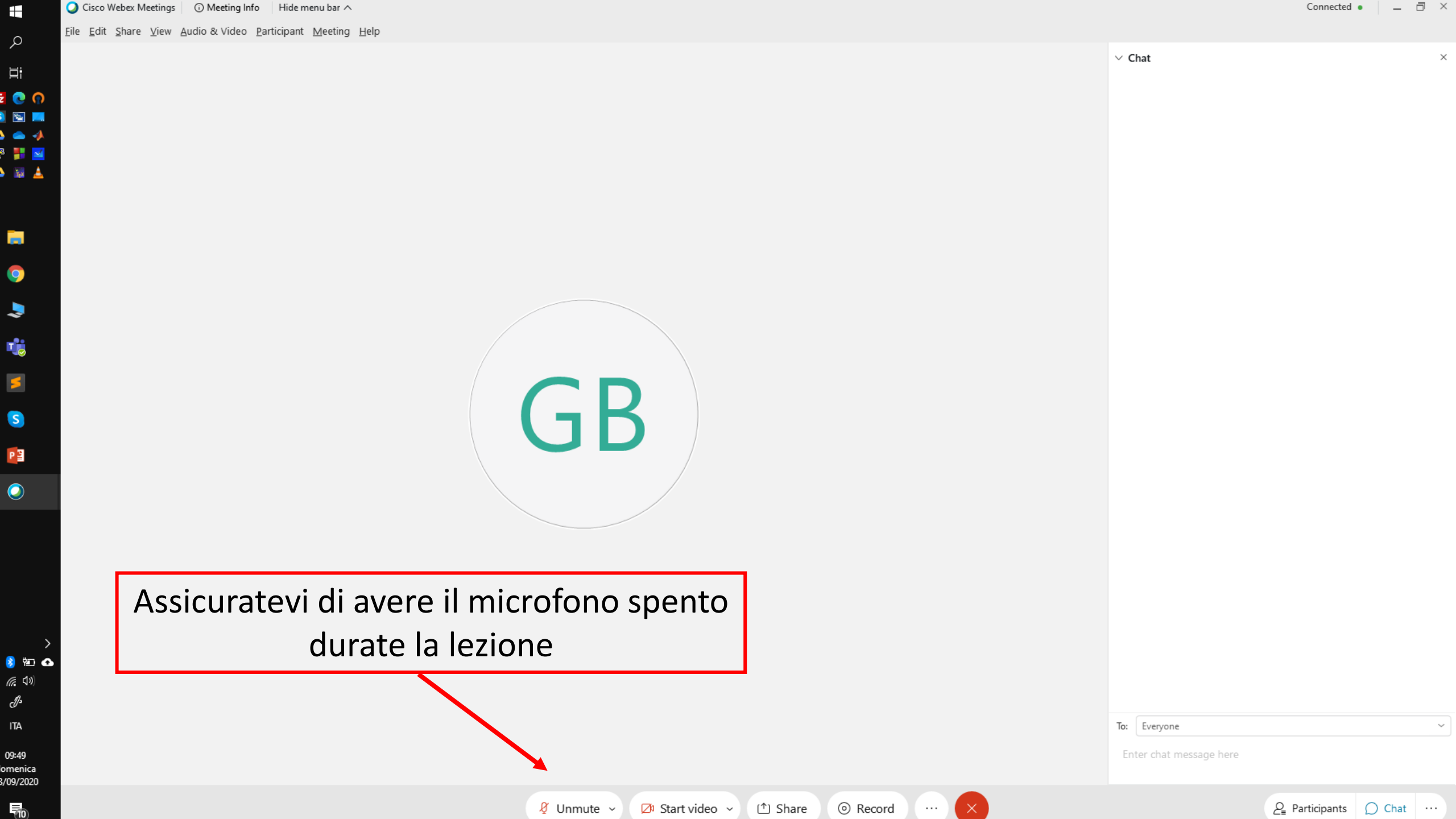
## Esercizi a lezione:

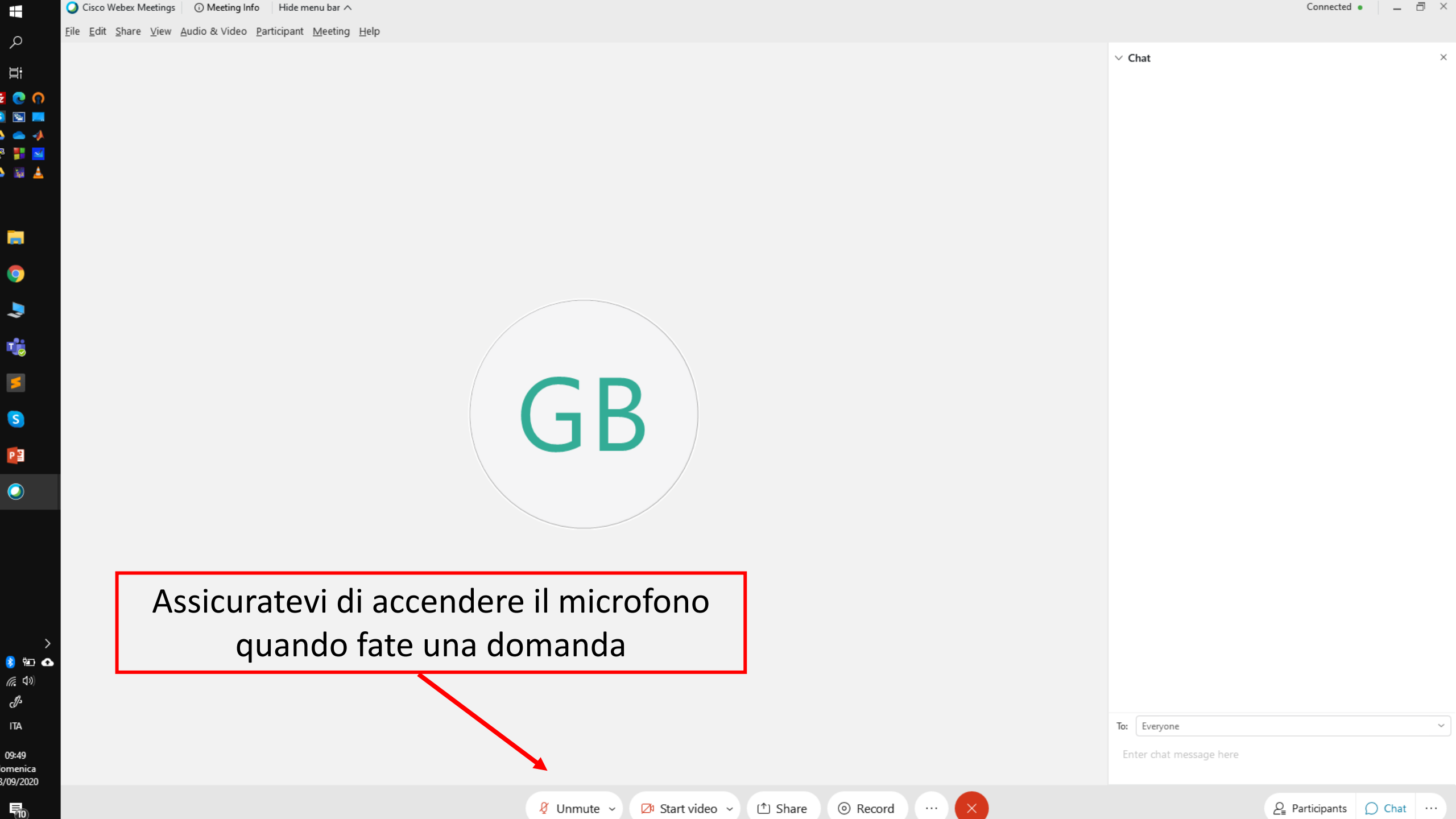
- Farò **molti esercizi** durante la lezione, **spesso svolti** al PC
- Caricherò i programmi in versione finale sul sito del corso
- Consiglio di **non programmare mentre seguite le lezioni**
- **Prendete appunti! Non fate affidamento sulle registrazioni** (non potete certo riguardarle tutte) o sulle slides (potrebbero contenere solo parte delle cose che spiego)

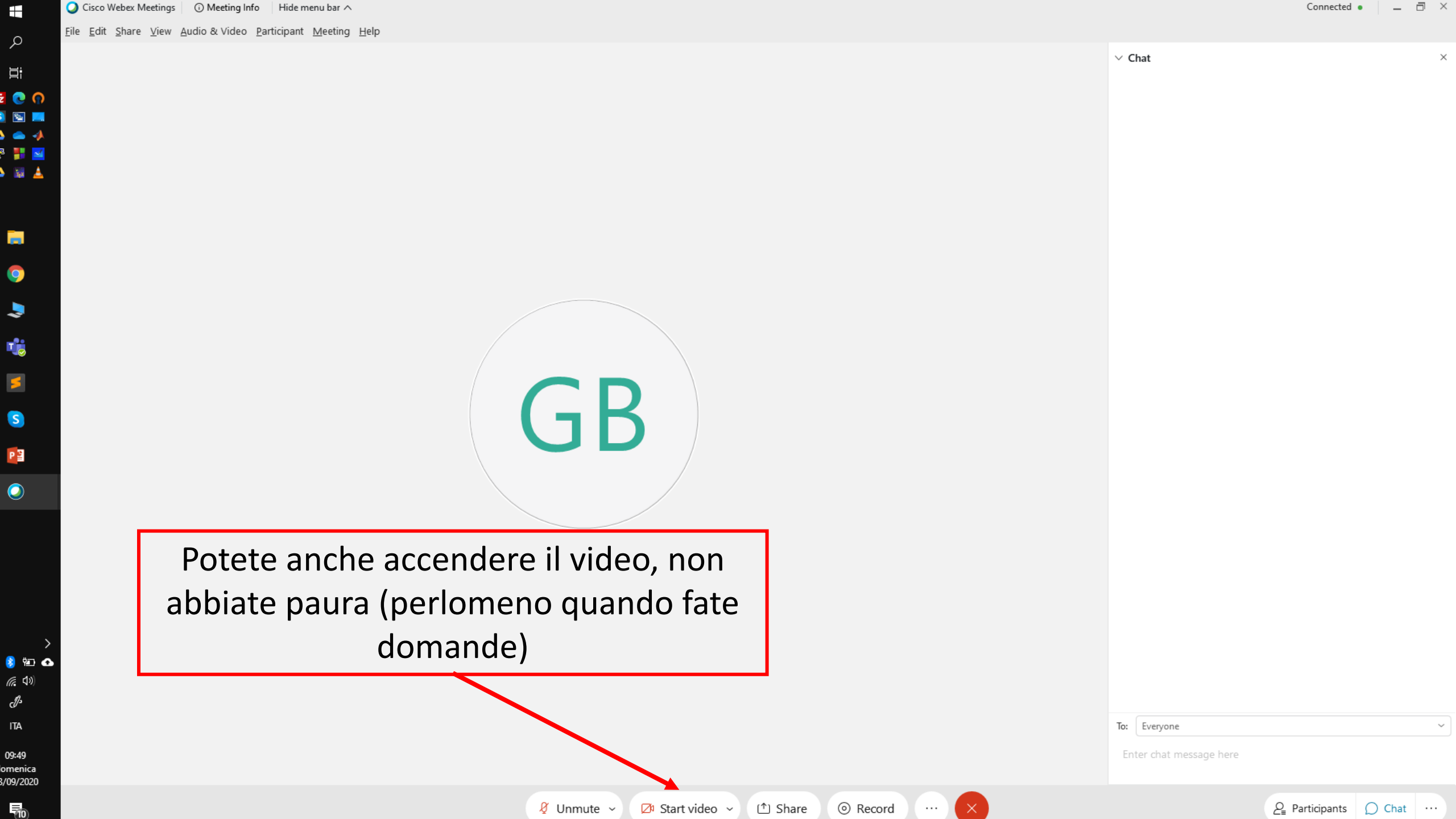
# Partecipazione alle Lezioni

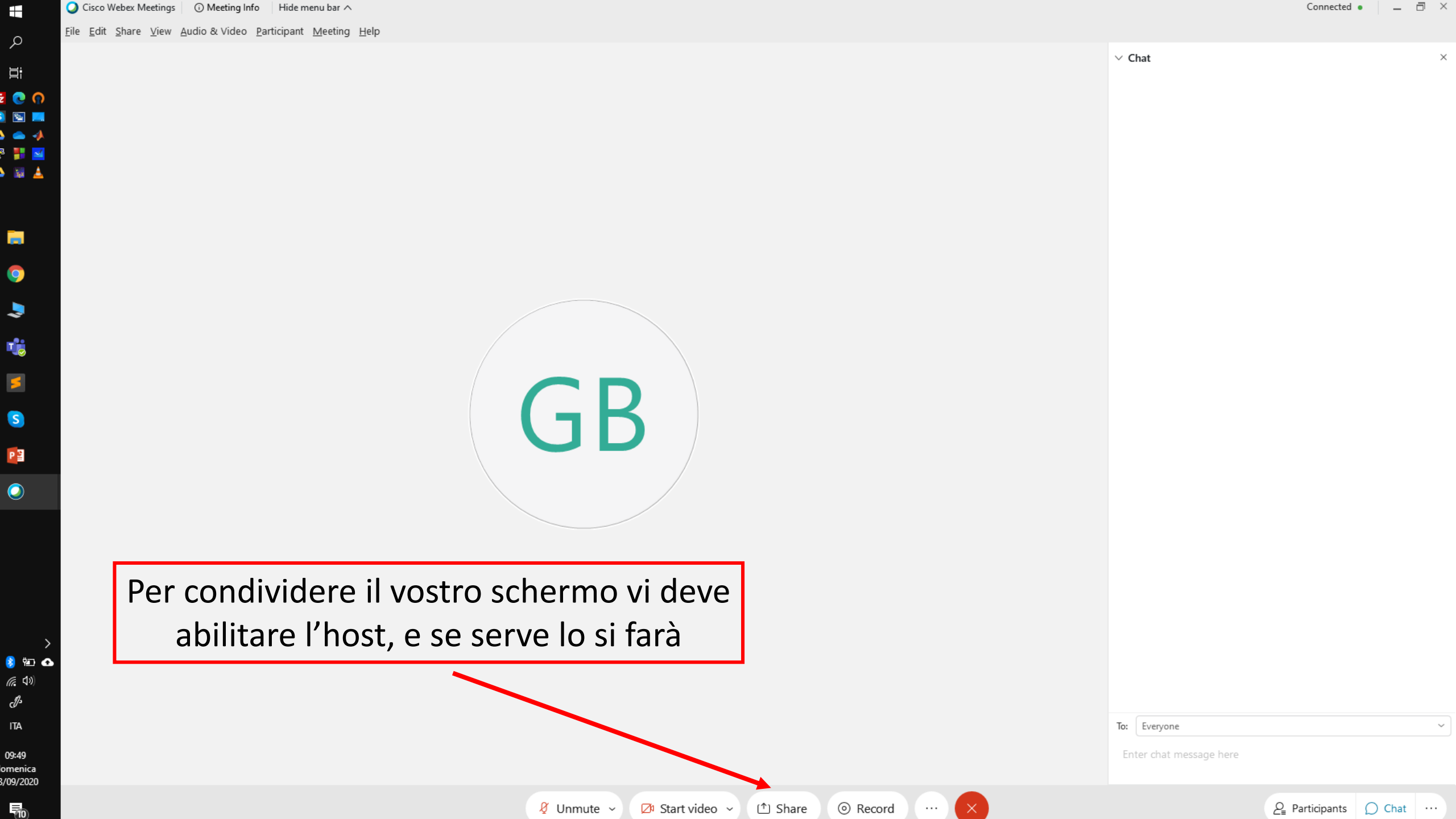
**Mi auspico la massima interazione durante lezioni/esercitazioni online/blended!**

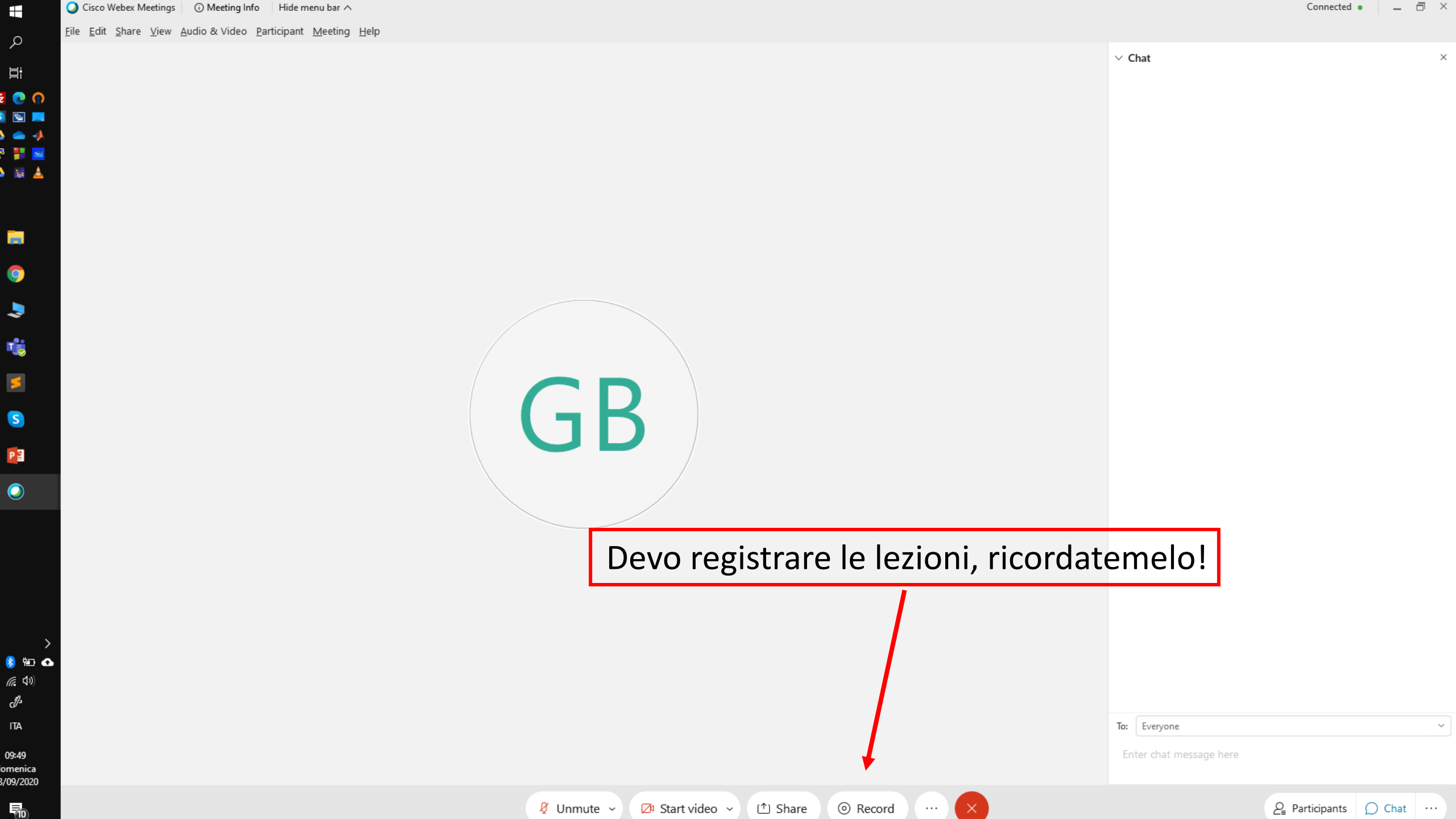
- Potete comunicare
- Togliendo il muto e ponendo educatamente una domanda (mi aspetto sappiate scegliere il momento opportuno)
- Scrivendo nella chat a tutta la classe (o al docente)

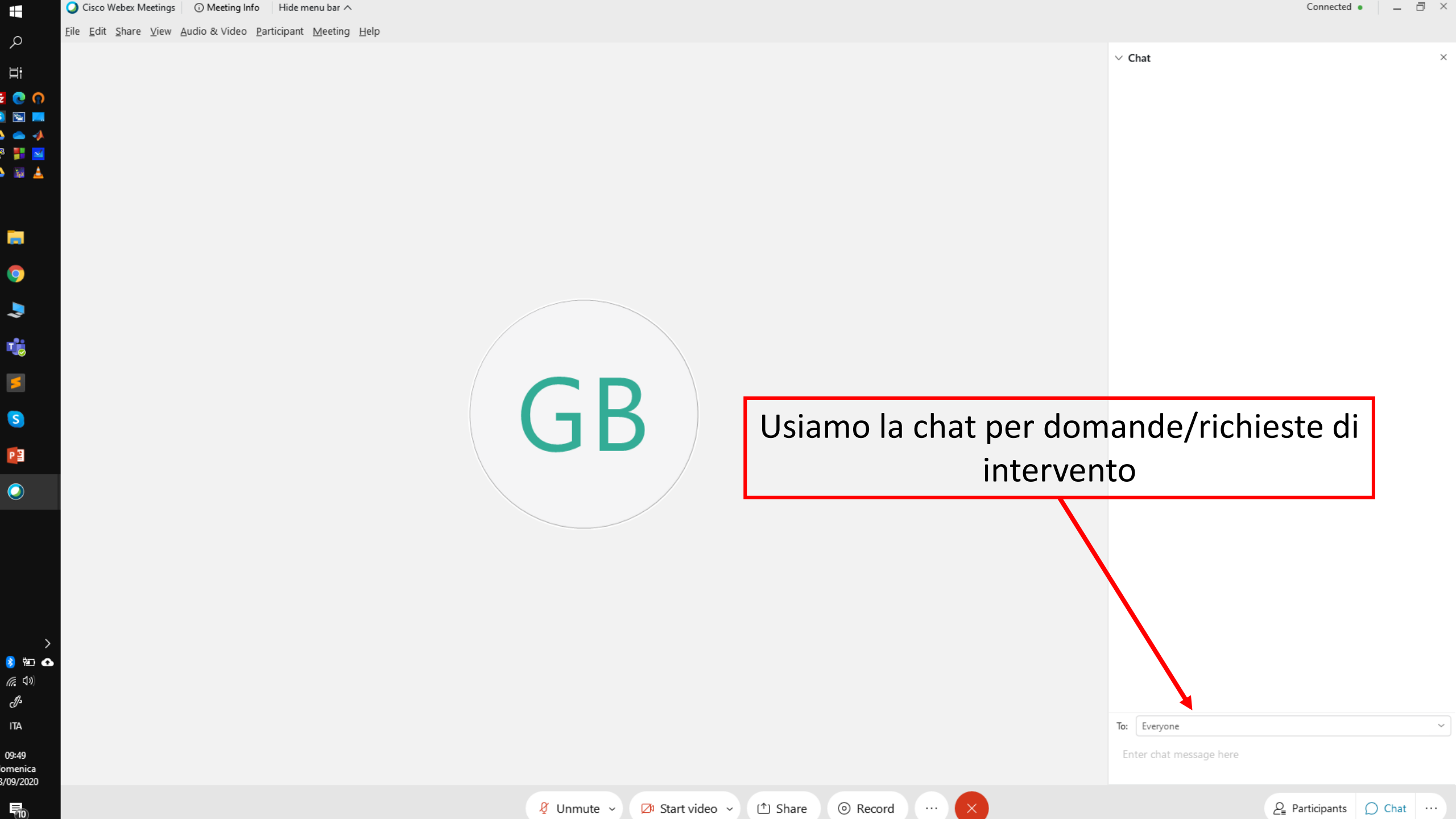

Cisco Webex Meetings
Meeting Info
Hide menu bar
Connected
File Edit Share View Audio & Video Participant Meeting Help
Chat
GB
Assicuratevi di avere il microfono spento durate la lezione
To: Everyone
Enter chat message here
Unmute
Start video
Share
Record
Participants
Chat
ITA
09:49
3/09/2020


Cisco Webex Meetings
Meeting Info
Hide menu bar
Connected
File Edit Share View Audio & Video Participant Meeting Help
Chat
GB
Assicuratevi di accendere il microfono quando fate una domanda
To: Everyone
Enter chat message here
Unmute
Start video
Share
Record
Participants
Chat
ITA
09:49
3/09/2020


Cisco Webex Meetings
Meeting Info
Hide menu bar
Connected
File Edit Share View Audio & Video Participant Meeting Help
Chat
GB
Potete anche accendere il video, non abbiate paura (perlomeno quando fate domande)
To: Everyone
Enter chat message here
Unmute
Start video
Share
Record
Participants
Chat
ITA
09:49
omenica
3/09/2020


Cisco Webex Meetings
Meeting Info
Hide menu bar
Connected
File Edit Share View Audio & Video Participant Meeting Help
Chat
GB
Per condividere il vostro schermo vi deve abilitare l'host, e se serve lo si farà
To: Everyone
Enter chat message here
Unmute
Start video
Share
Record
Participants
Chat
ITA
09:49
omenica
3/09/2020


Cisco Webex Meetings
Meeting Info
Hide menu bar
Connected
File Edit Share View Audio & Video Participant Meeting Help
Chat
GB
Devo registrare le lezioni, ricordatemelo!
To: Everyone
Enter chat message here
Unmute
Start video
Share
Record
Participants
Chat
ITA
09:49
3/09/2020


Cisco Webex Meetings
Meeting Info
Hide menu bar
Connected
File Edit Share View Audio & Video Participant Meeting Help
Chat
GB
Usiamo la chat per domande/richieste di intervento
To: Everyone
Enter chat message here
Unmute
Start video
Share
Record
Participants
Chat
ITA
09:49
omenica
3/09/2020

# Richieste di Chiarimento

# Richieste di Chiarimento

È bene che le lezioni siano quanto più interattive possibile

Domande e richieste di chiarimenti sono sempre ben accette:

- Domande via chat
- Domandando direttamente a voce/video

Potete anche rivolgere domande via mail

- Però, per facilitare il lavoro di tutti, è bene evitare richieste vaghe e del tipo:

«è corretto così?»

Secure | https://mail.google.com/mail/u/0/#all/159e546f3c53b2df

Google

Gmail

COMPOSE

Inbox (12)
Starred
Important
Sent Mail
Drafts (5)
All Mail
Spam (3)
Bin
Categories
[Gipi] (361)
[Lista PD] (450)
[matlab notifications...
Gilardoni
polimi.it (4,547)
smafsoft.it (1,184)
master-information
More

Move to Inbox | More

# Domanda tde

Inviata da Polimi x | polimi.it x

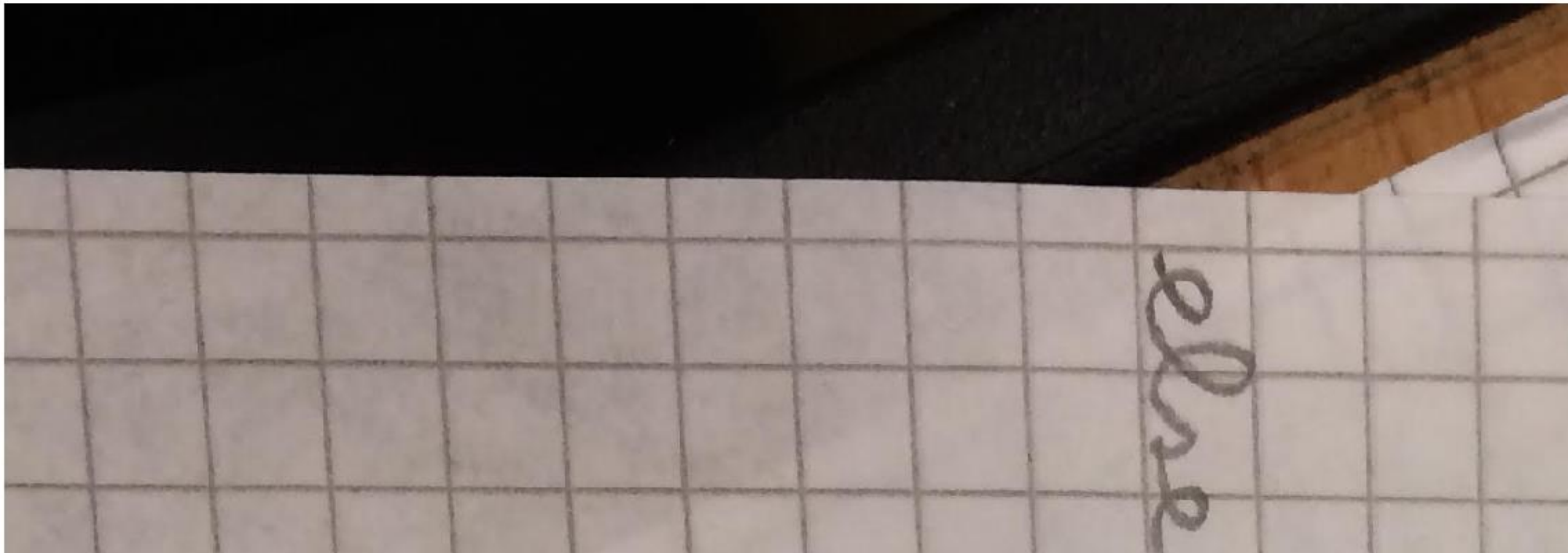
[redacted]
to Francesco, Giacomo

Buona sera,

È corretta questa soluzione per il punto 2 dell'esercizio 2 del tde 28/01/13?

Cordiali saluti

[redacted]



## Esercizio 3 (4 punti)

Dati i seguenti due numeri in codifica IEEE 754 (virgola mobile, il bit più a sinistra è ovviamente il bit di segno)

```
A = S:0 E:01111111 M:00100000000000000000000
B = S:1 E:01111100 M:00100000000000000000000
```

Calcolare a quanto equivale la divisione A/B (in decimale).

### Soluzione

I numeri A e B si differenziano solo per esponente e segno. Si può quindi calcolare la divisione A/B prendendo in considerazione solo gli esponenti:

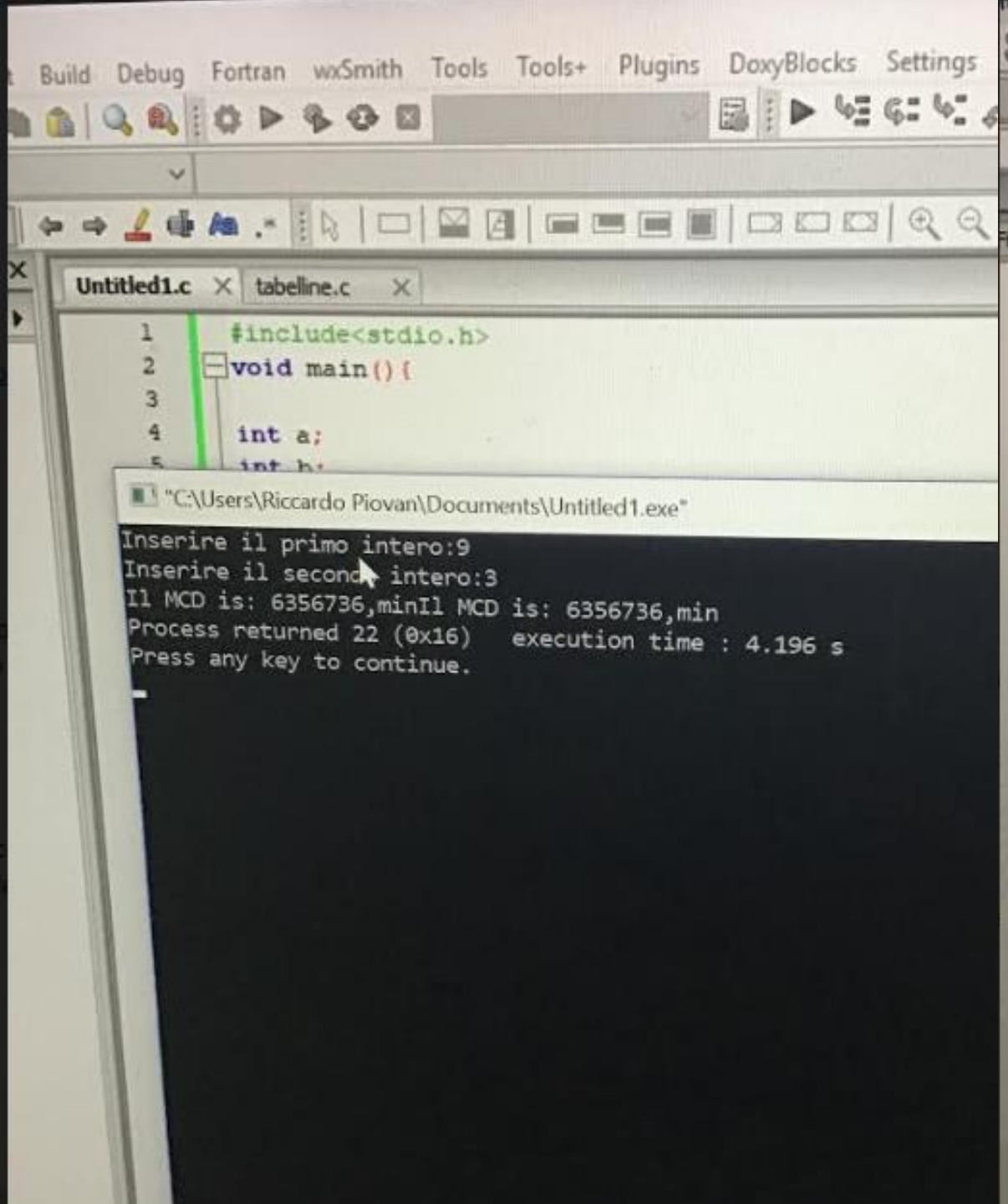
```
A = - B * 2^3
```

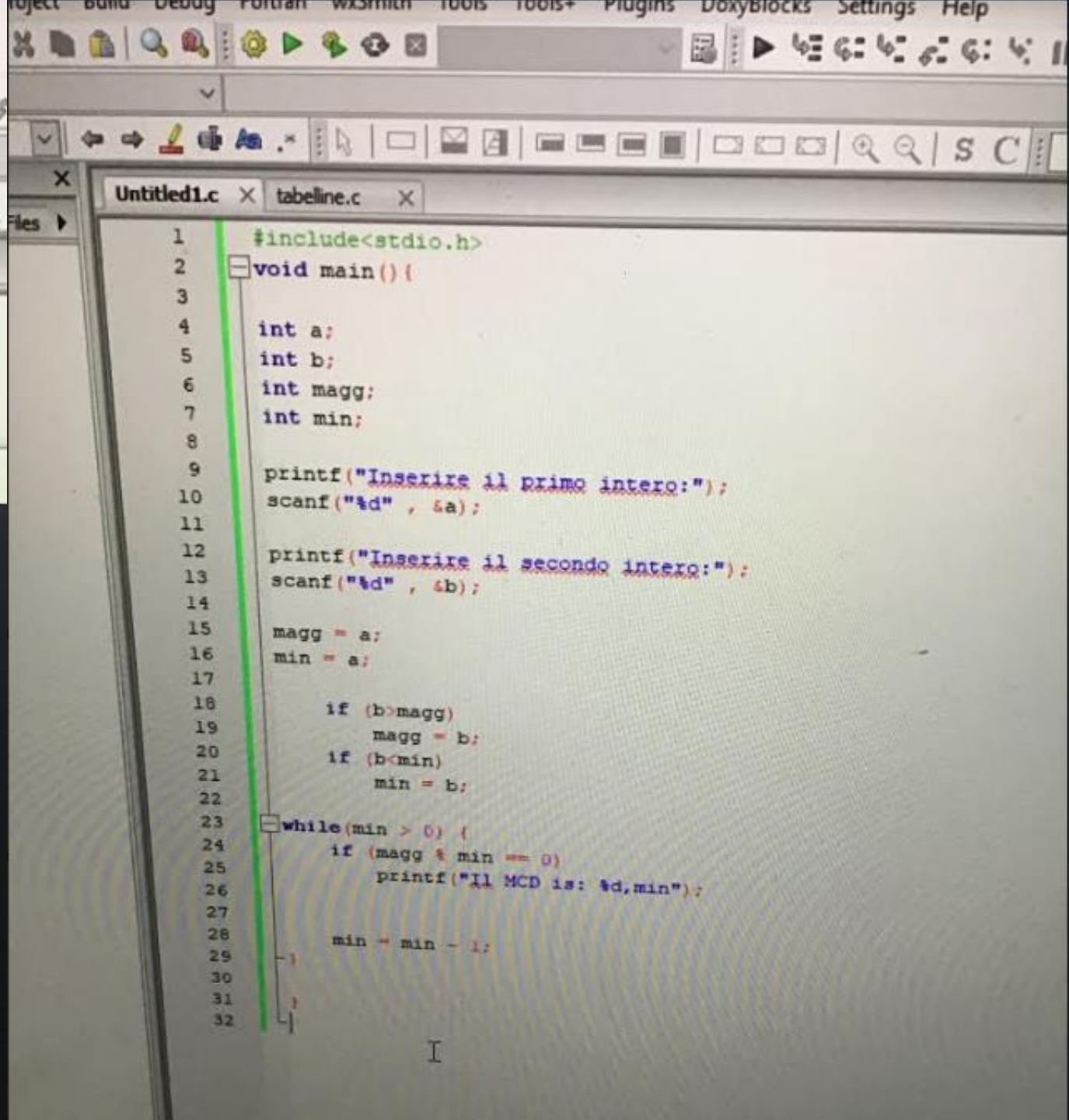
Quindi:

```
A/B = - 8
```

Si può anche notare che:

```
A =       (1 + 2^(-3))           =   1.125
B = -1 * (1 + 2^(-3)) * 2^(-3)   = - 0.140625
```


Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings
Untitled1.c
tabelline.c
1 #include<stdio.h>
2 void main(){
3
4 int a;
"C:\Users\Riccardo Piovan\Documents\Untitled1.exe"
Inserire il primo intero:9
Inserire il secondo intero:3
Il MCD is: 6356736,minIl MCD is: 6356736,min
Process returned 22 (0x16) execution time : 4.196 s
Press any key to continue.


Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Untitled1.c
tabelline.c
1 #include<stdio.h>
2 void main(){
3
4 int a;
5 int b;
6 int magg;
7 int min;
8
9 printf("Inserire il primo intero:");
10 scanf("%d" , &a);
11
12 printf("Inserire il secondo intero:");
13 scanf("%d" , &b);
14
15 magg = a;
16 min = a;
17
18 if (b>magg)
19 magg = b;
20 if (b<min)
21 min = b;
22
23 while(min > 0) {
24 if (magg % min == 0)
25 printf("Il MCD is: %d,min");
26
27
28 min = min - 1;
29 }
30
31 }
32

# Richieste via mail

Quando scrivete una mail per una richiesta di chiarimento:

- dite di che corso siete (tengo un altro corso di informatica!)
- allegate Il codice sorgente, ripulito e commentato
- allegate il testo dell’esercizio (no foto)
- un breve commento che spiega cosa non funziona e i tentativi che avete fatto

# L’Esame

# Modalità di Verifica

Solo appelli regolari (niente prove intermedie):

- **ci sono 5 appelli regolari, con date disponibili nel calendario del facoltà/sistema online.**
- nessun appello oltre a questi
- il laboratorio non sarà valutato
- vi verrà richiesto **di risolvere dei programmi in Matlab al computer**

**L'esame orale non è previsto** se non a discrezione del docente

**Studenti con OFA in matematica (o totale) non possono sostenere** le prove prima di aver passato l'OFA

- **È necessario iscriversi all'esame prima della chiusura**
- Non è possibile verbalizzare l'esame se non siete iscritti

# Punti Bonus

Gli studenti potranno ricevere **punti bonus** per:

- la loro partecipazione attiva a lezioni/esercitazioni/laboratori
- partecipazione a contest presentati nel corso delle lezioni

I punti bonus sono a discrezione del docente/esercitatore

Verranno conteggiati direttamente sul punteggio dell’esame

Esame online

Press F11 to exit full screen

# Informatica A 28/8/2020

La fase 1 si compone di DUE ESERCIZI C E DUE QUERY SQL.

- Ogni campo risposta puo' contenere AL MASSIMO 140 RIGHE.
Controllate che il vostro codice sia in questo limite e, nel caso non lo fosse, compattate il codice. In casi estremi potete non includere le funzioni ausiliarie che vi abbiamo fornito (esercizio delle liste) nelle soluzioni che caricate.

- potete sottomettere UNA SOLA RISPOSTA.

- e' necessario inserire almeno un carattere nel campo "risposta" di ogni domanda per poter sottomettere il form.

- TEMPO A DISPOSIZIONE 1H20

...

Hi Giacomo, when you submit this form, the owner will be able to see your name and email address.

* Required

1. Selezionare la TERZULTIMA (cioè la SESTA) cifra del vostro codice persona 10XXXXXX *

- ○ 0 oppure 1 oppure 3
- ○ 2 oppure 4
- ◉ 5 oppure 6 oppure 9
- ○ 7 oppure 8

2. Si sviluppi una funzione colonneGrandi che prende in ingresso una matrice quadrata di interi B che ha r righe e c colonne (r e c possono essere inferiori alla dimensione con cui B viene dichiarata), e restituisce al programma chiamante un vettore v di c - 1 elementi che contiene, nella posizione i-sima il numero di elementi della colonna i-sima che sono maggiori della media

# Contenuti del Corso

# Argomenti Trattati

- Introduzione all’informatica
- Algoritmi
- Aritmetica Binaria
- Programmazione in linguaggio Matlab
- Argomenti avanzati di programmazione

# Bibliografia

“*Materiale su sistemi informatici e i principi di programmazione in C per il corso di Informatica B* “, Editore: Mc Graw Hill, Anno edizione: 2016, ISBN: 9781308911731 (*)

A. Campi, E. Di Nitto, D. Loiacono, A. Morzenti, B. Spoletini, “*Introduzione alla programmazione in Matlab*”, seconda edizione.

# Bibliografia (nota importante)


Questo testo contiene i capitoli del *Informatica Arte e Mestiere* che sono rilevanti per il corso di Informatica B

Se siete in possesso di *Informatica Arte e Mestiere*, non dovete acquistare anche questo

## Cosa imparerete:

- Gli **elementi fondamentali** ed i **principi** che regolano il funzionamento di un **sistema informatico**
- Alcune nozioni di base sulla **codifica binaria** e l'algebra di Boole
- Come **sviluppare algoritmi** per risolvere problemi
- Come **codificare** tali **algoritmi** in programmi che ne permettano l'automatizzazione
- Le basi della **programmazione**
- L'utilizzo e la scrittura di programmi nel linguaggio **Matlab**

# Cos’è l’informatica?

*«Scienza della rappresentazione e dell’elaborazione dell’informazione.»*

# Cos’è l’Informatica?

*Scienza della rappresentazione e dell’elaborazione dell’informazione.*

[da «Informatica Arte e Mestiere»]

- ***Scienza:*** ovvero una **conoscenza sistematica** e **rigorosa** di tecniche e metodi
- ***Informazione:*** l’oggetto dell’investigazione scientifica (informazione intesa come entità astratta e come tecnologie per la sua gestione)
- ***Rappresentazione:*** il modo in cui l’informazione viene strutturata e trasformata in dati fruibili da macchine
- ***Elaborazione:*** uso e trasformazione dell’informazione per un dato scopo. L’elaborazione deve poter essere eseguita da macchine che processano dati

# Cos’è l’Informatica?

*«Studio sistematico degli **algoritmi** che descrivono e trasformano l’informazione»*
[da Association for Computing Machinery (ACM)]

- la loro teoria
- analisi
- progetto
- efficienza
- realizzazione
- applicazione

# Gli Algoritmi

# Algoritmo

***Sequenza*** *precisa di* ***operazioni****, definiti con* ***precisione****, che portano alla* ***realizzazione di un compito***

Le operazioni devono:

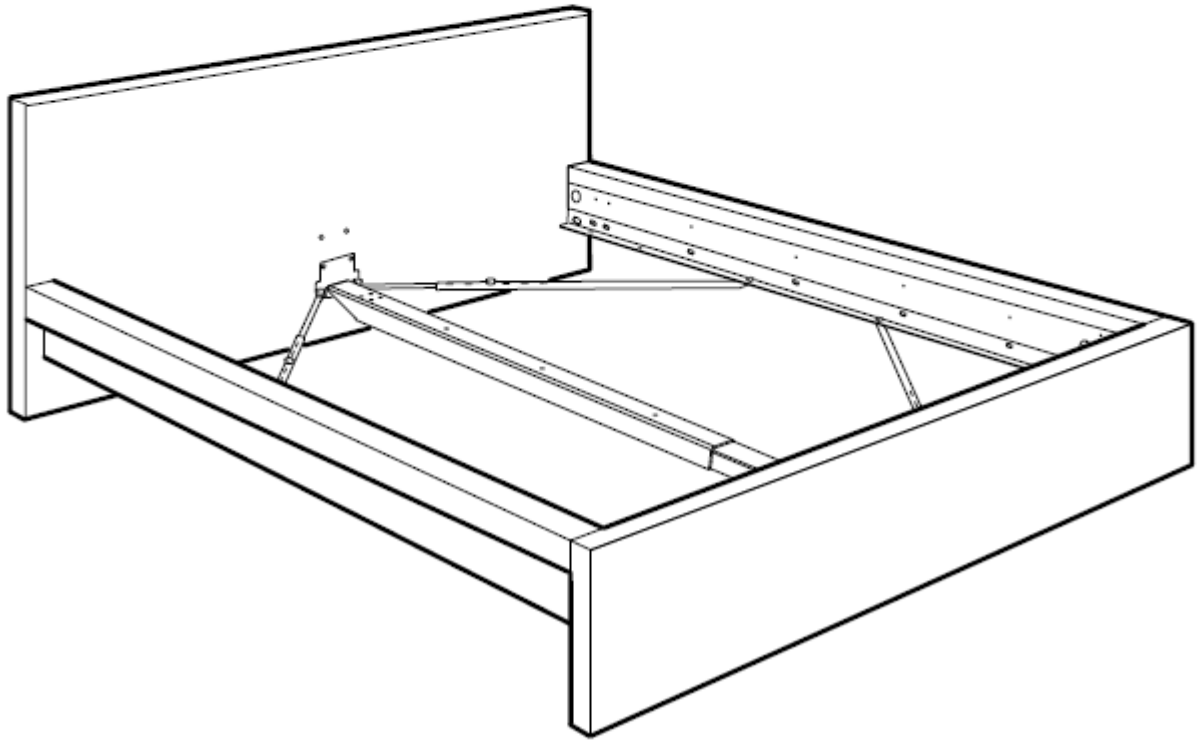
essere **comprensibili** senza ambiguità

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)

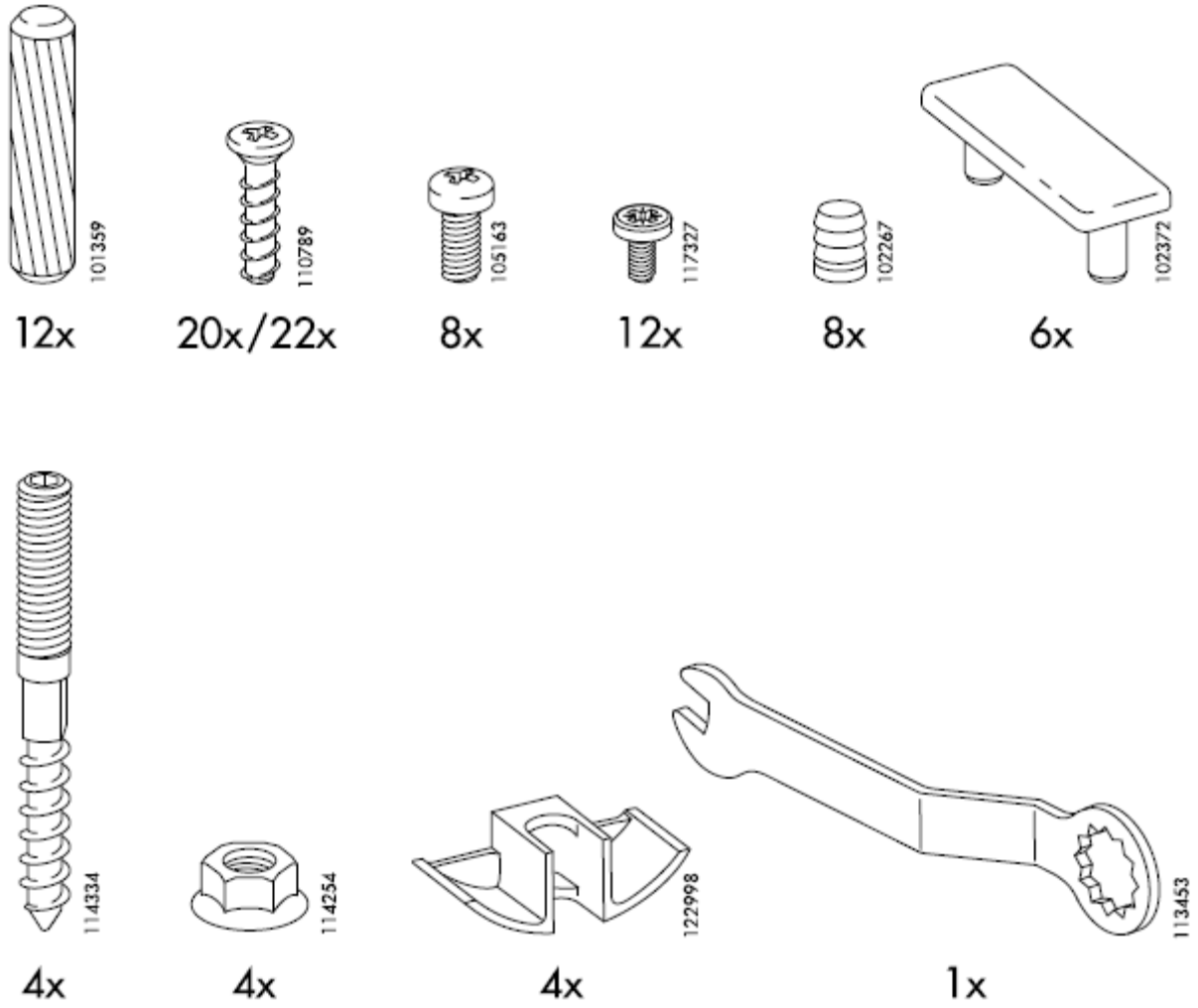
…ora vedermo un esempio di algoritmo in cui vi sarà capitato di essere esecutori

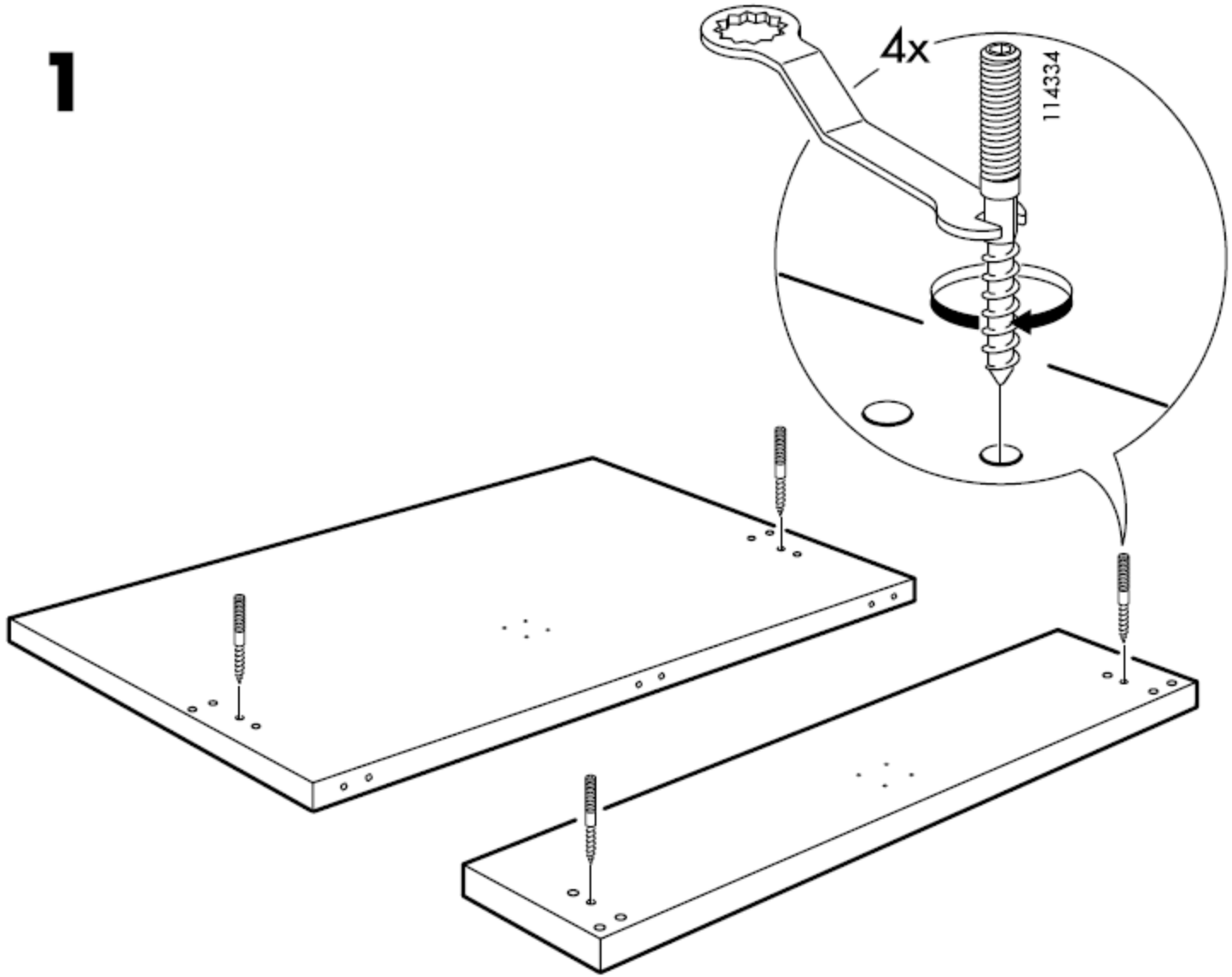
# MALM

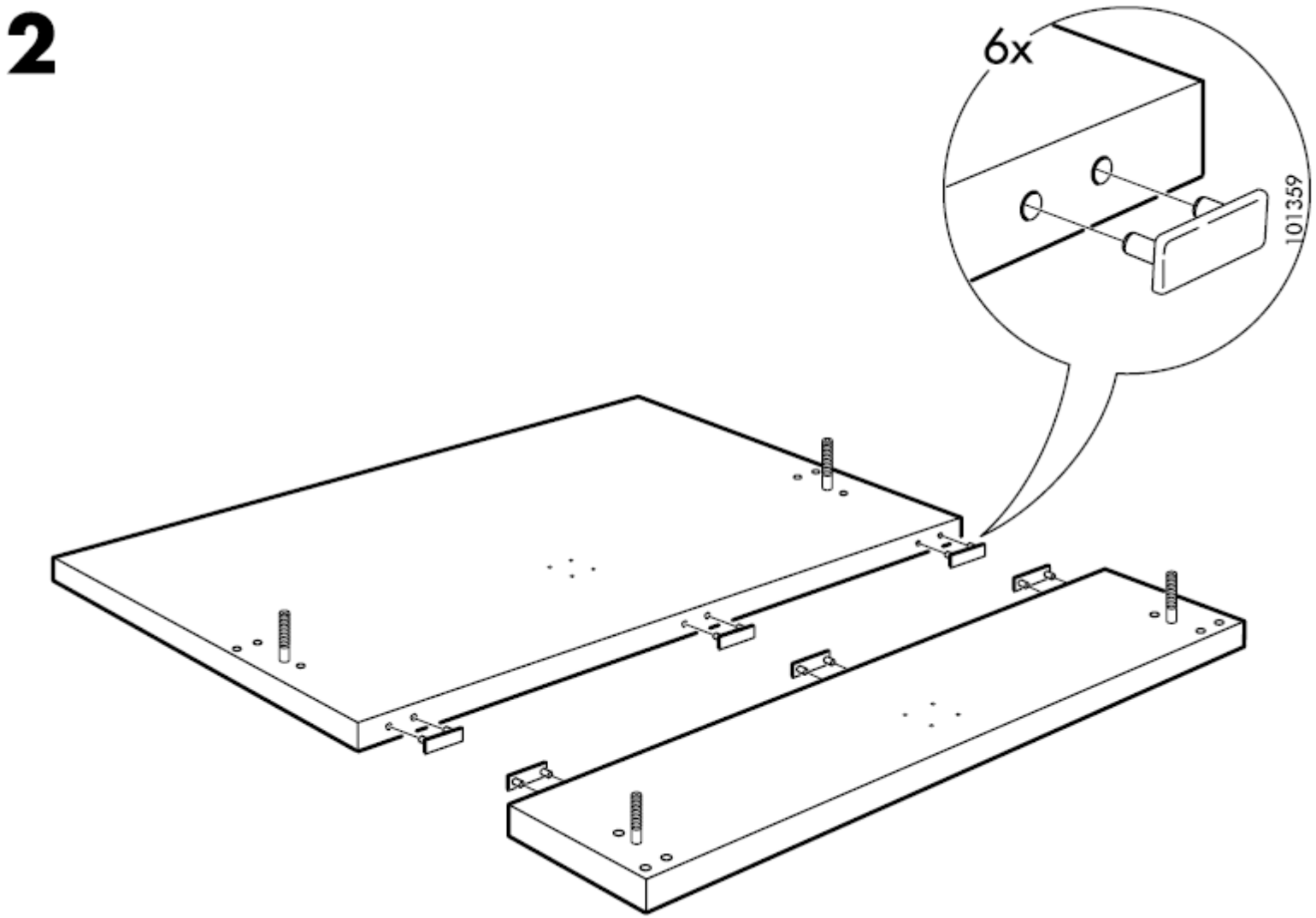


IKEA®

Design and Quality
IKEA of Sweden


101359
12x
110789
20x/22x
105163
8x
117327
12x
102267
8x
102372
6x
114334
4x
114254
4x
122998
4x
113453
1x


1
4x
114334


2
6x
101359

# Proprietà fondamentali degli algoritmi

**Correttezza**:

- l'algoritmo risolve il compito senza errori o difetti

**Efficienza**:

- l'algoritmo usa risorse (tempo e spazio) in modo minimale (o almeno ragionevole)

Nota:

- le istruzioni IKEA sono fatte per esecutori intelligenti (noi, *ndr*) l'interpretazione dei disegni richiede diverse capacità
- quando l'esecutore è meno intelligente occorre esprimere le istruzioni in un linguaggio più preciso

# Il Linguaggio

Supponiamo di avere un bambino che sa contare, ma non sa fare operazioni aritmetiche e non sa scrivere

Scriviamo le istruzioni per utilizzare il pallottoliere

# Linguaggi di Programmazione

# Programmazione a Basso/Alto Livello

**Linguaggio macchina**: poche istruzioni, difficile codificare algoritmi e interpretare il codice

- Linguaggio preciso
- Completo controllo delle risorse

# Esempio di Linguaggio a basso livello

| | |
|---|---|
| 0100000000010000 | **Leggi** un valore dall'input e mettilo nella cella 16 (**a**) |
| 0100000000010001 | **Leggi** un valore dall'input e mettilo nella cella 17 (**b**) |
| 0100000000010010 | **Leggi** un valore dall'input e mettilo nella cella 18 (**c**) |
| 0100000000010011 | **Leggi** un valore dall'input e mettilo nella cella 19 (**d**) |
| 0000000000010000 | **Carica** il contenuto della cella 16 (**a**) **nel registro A** |
| 0001000000010001 | **Carica** il contenuto della cella 17 (**b**) **nel registro B** |
| 0110000000000000 | **Somma** i registri A e B |
| 0010000000010100 | **Scarica** il contenuto di A nella cella 20 (**z**) (ris.parziale) |
| 0000000000010010 | **Carica** il contenito della cella 18 (**c**) **nel registro A** |
| 0001000000010011 | **Carica** il contenito della cella 19 (**d**) **nel registro B** |
| 0110000000000000 | **Somma** i registri A e B |
| 0001000000010100 | **Carica** il contenuto della cella 20 (**z**) (ris. parziale) **in B** |
| 1000000000000000 | **Moltiplica** i registri A e B |
| 0010000000010100 | **Scarica** il contenuto di A nella cella 20 (**z**) (ris. totale) |
| 0101000000010100 | **Scrivi** il contenuto della cella 20 (**z**) (ris. totale)  output |
| 1101000000000000 | **Halt** |

# Programmazione a Basso/Alto Livello

**Linguaggio macchina**: poche istruzioni, difficile codificare algoritmi e interpretare il codice

- Linguaggio preciso
- Completo controllo delle risorse

**Linguaggi di alto livello**: linguaggi più comprensibili per l'uomo

- Linguaggio preciso e sintetico
- Riferimenti simbolici
- Esprimere istruzioni in linguaggio vicino a quello naturale

# Esempio di linguaggio ad alto livello

```
value = 1000;
year = 0;
while value < 2000
    value = value * 1.08
    year = year + 1;
    fprintf('%g years: $%g\n', year,value)
end
```

Nota: il programma calcola gli interessi fino al raddoppio del capital con un tasso dell’ 8%

# Programmazione a Basso/Alto Livello

**Linguaggio macchina**: poche istruzioni, difficile codificare algoritmi e interpretare il codice

- Linguaggio preciso
- Completo controllo delle risorse

**Linguaggi di alto livello**: linguaggi più comprensibili per l'uomo

- Linguaggio preciso e sintetico
- Riferimenti simbolici
- Esprimere istruzioni in linguaggio vicino a quello naturale

La traduzione dal linguaggio ad alto livello al linguaggio macchina è eseguita da un altro programma, il **compilatore** o dall'**interprete**

# Elementi Caratteristici di un Linguaggio di Programmazione

Proviamo a capire quali sono le caratteristiche e le operazioni essenziali richieste quando pensiamo di progettare un linguaggio di programmazione

Nota: le funzionalità che andremo ad analizzare sono comuni a gran parte dei linguaggi di programmazione esistenti

# La Sequenzialità

# Esempio: Algoritmo per Andare in Università

Mi alzo quando suona la sveglia

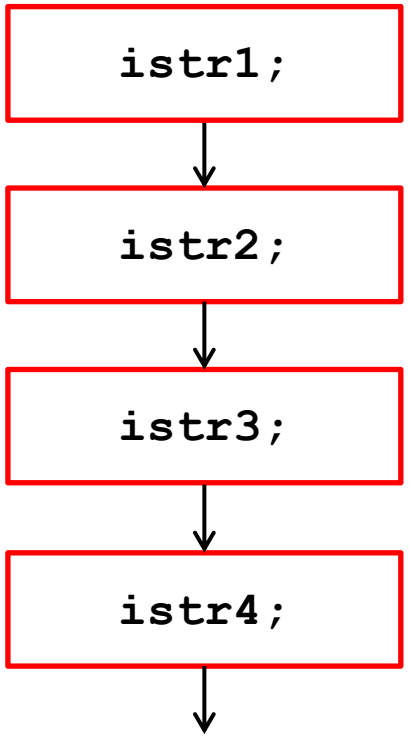
Mi dirigo verso la cucina

Mangio e bevo un caffè

Mi lavo

Mi vesto

# La Sequenzialità

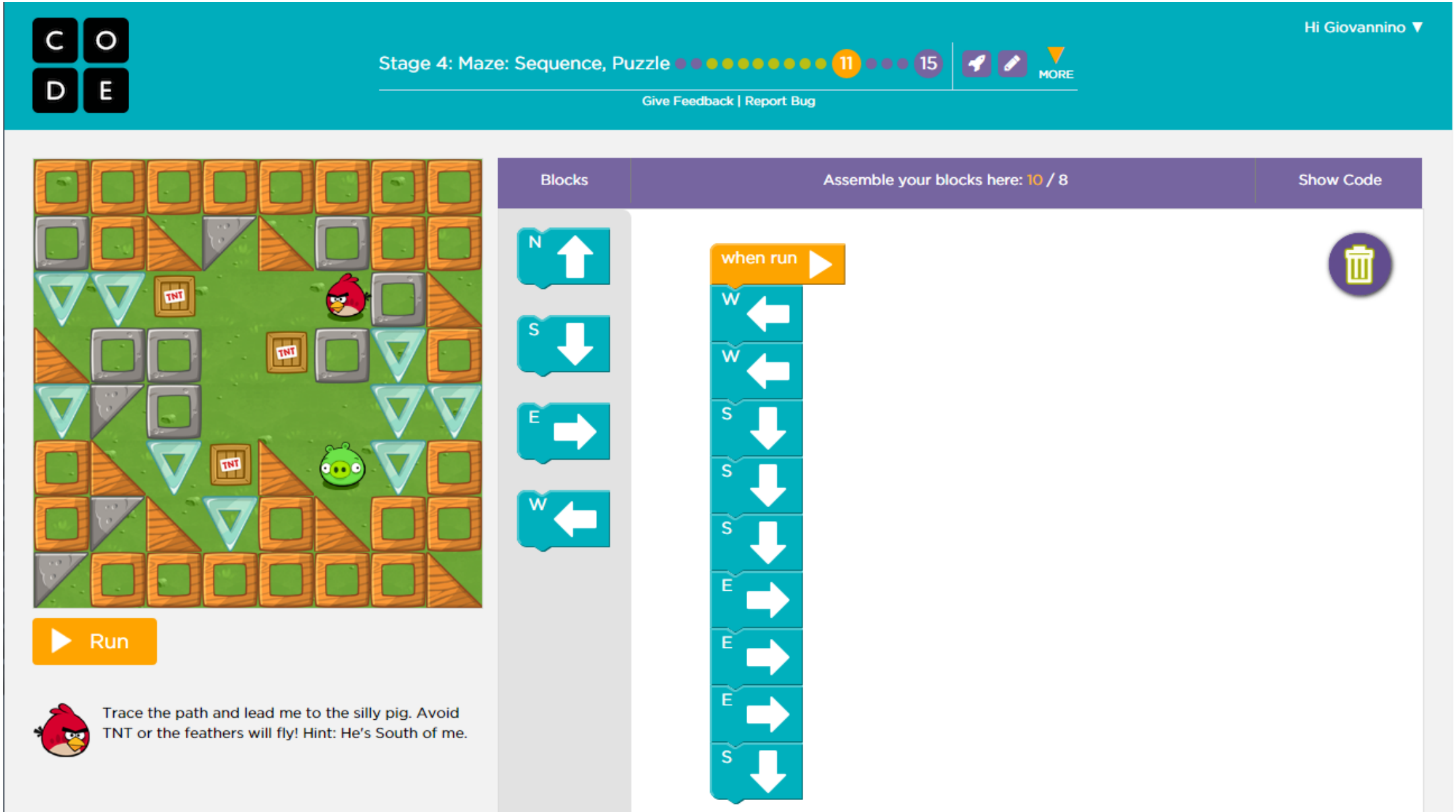



```
istr1;
istr2;
istr3;
istr4;
...
```

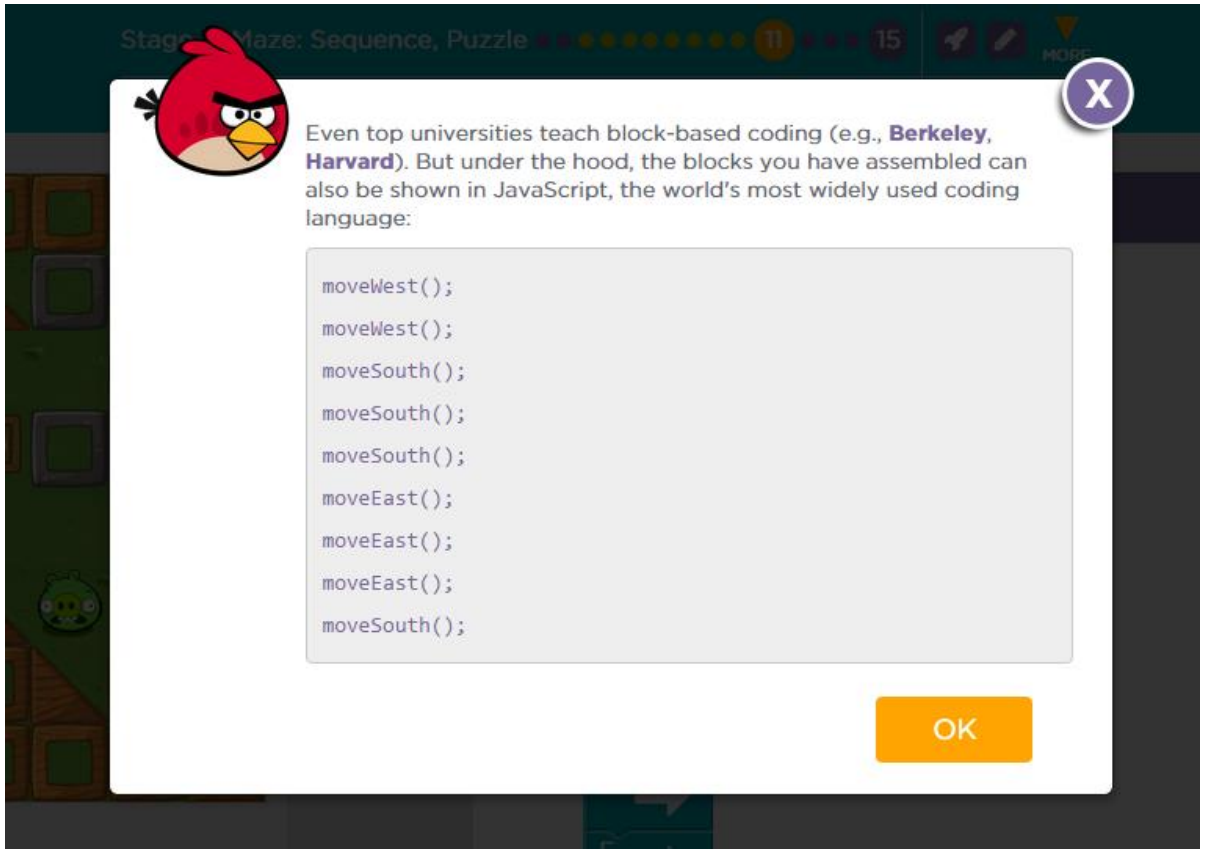
Le istruzioni vengono eseguite dalla prima all'ultima

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

# La Sequenzialità su code.org



# La Sequenzialità su code.org



# Costrutto Condizionale

# 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

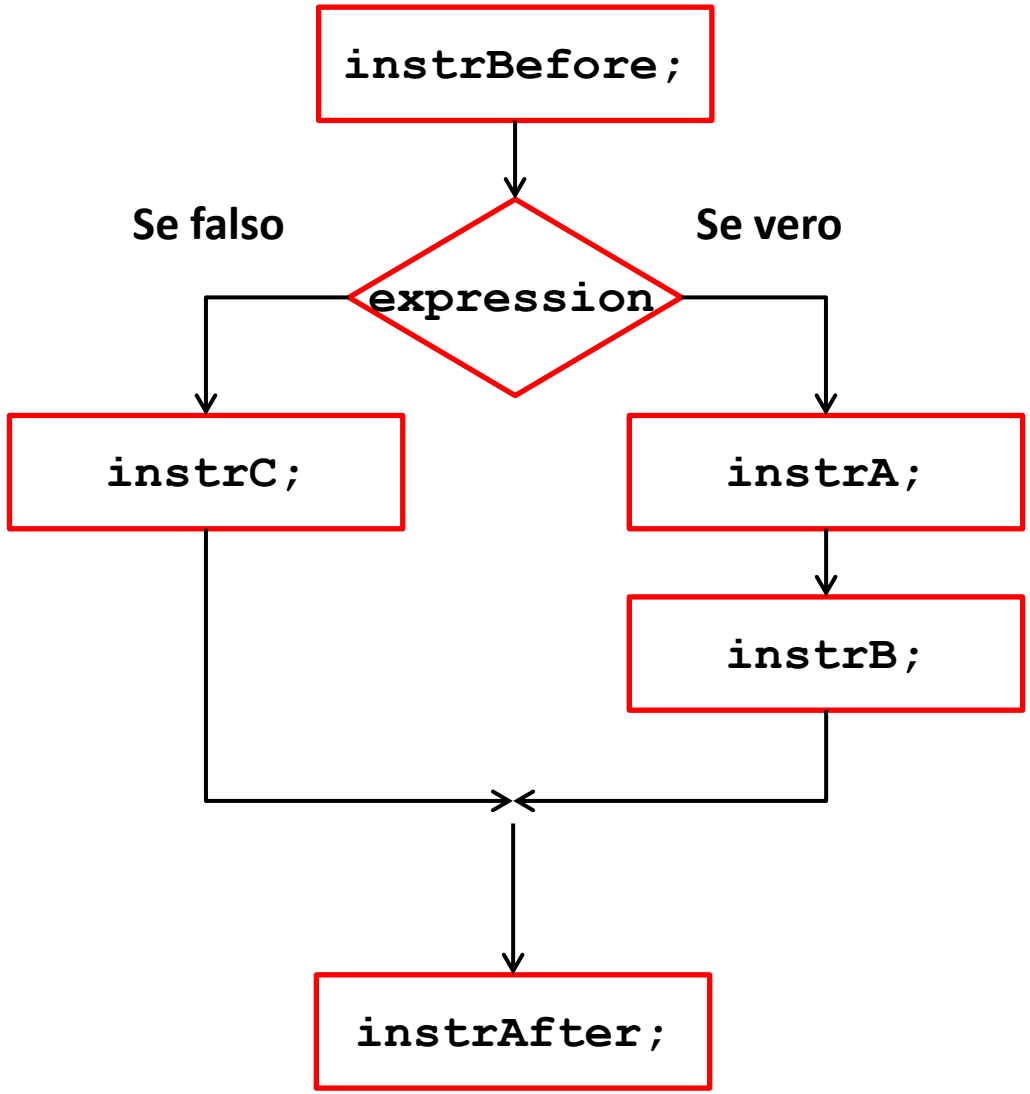
**Se** devo mangiare fuori

- prendo uno snack per metà mattina

**Altrimenti**

- prendo il pranzo

# Costrutto Condizionale:

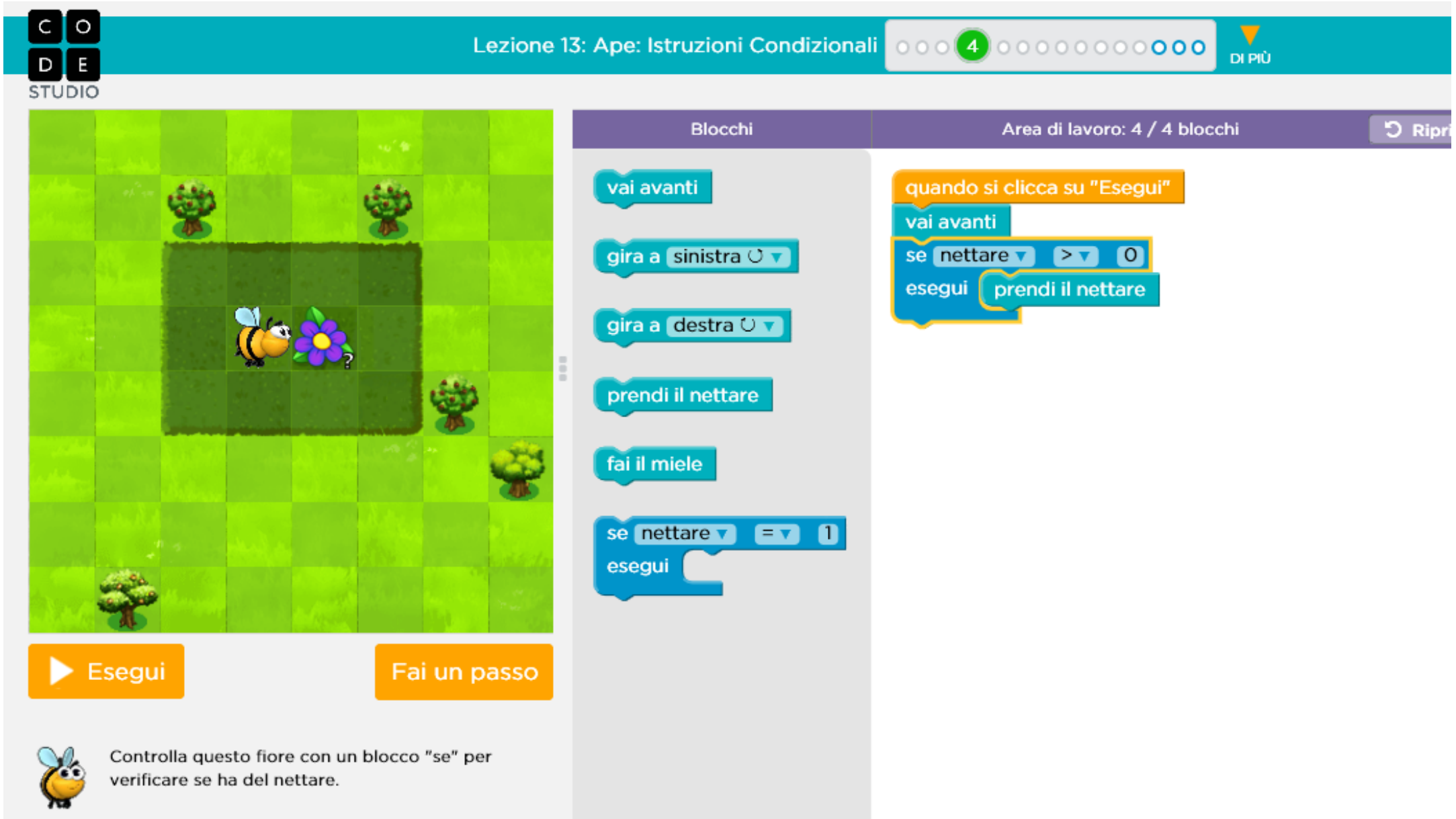



Dopo aver eseguito **`instrBefore`** si valuta **`expression`**

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

Altrimenti eseguo **`instrC;`**

# Il costrutto condizionale su code.org



# Il Costrutto Condizionale su code.org

# Esempio: Algoritmo per Andare in Università

Mi alzo quando suona la sveglia

Mi dirigo verso la cucina

Mangio e bevo un caffè

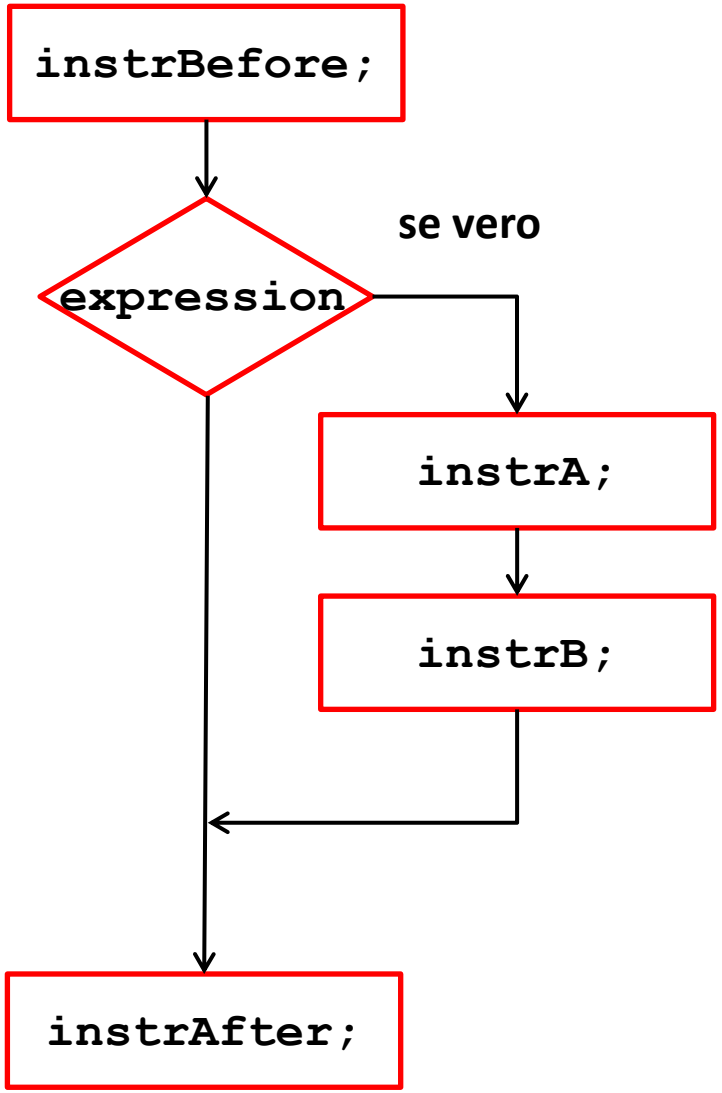
Mi lavo e mi vesto

**Se** c'è il laboratorio di informatica

- Prendo il laptop

Esco di casa

# Costrutto Condizionale:




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

# Costrutto Iterativo

# 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

**Ripeti**: piove ancora?

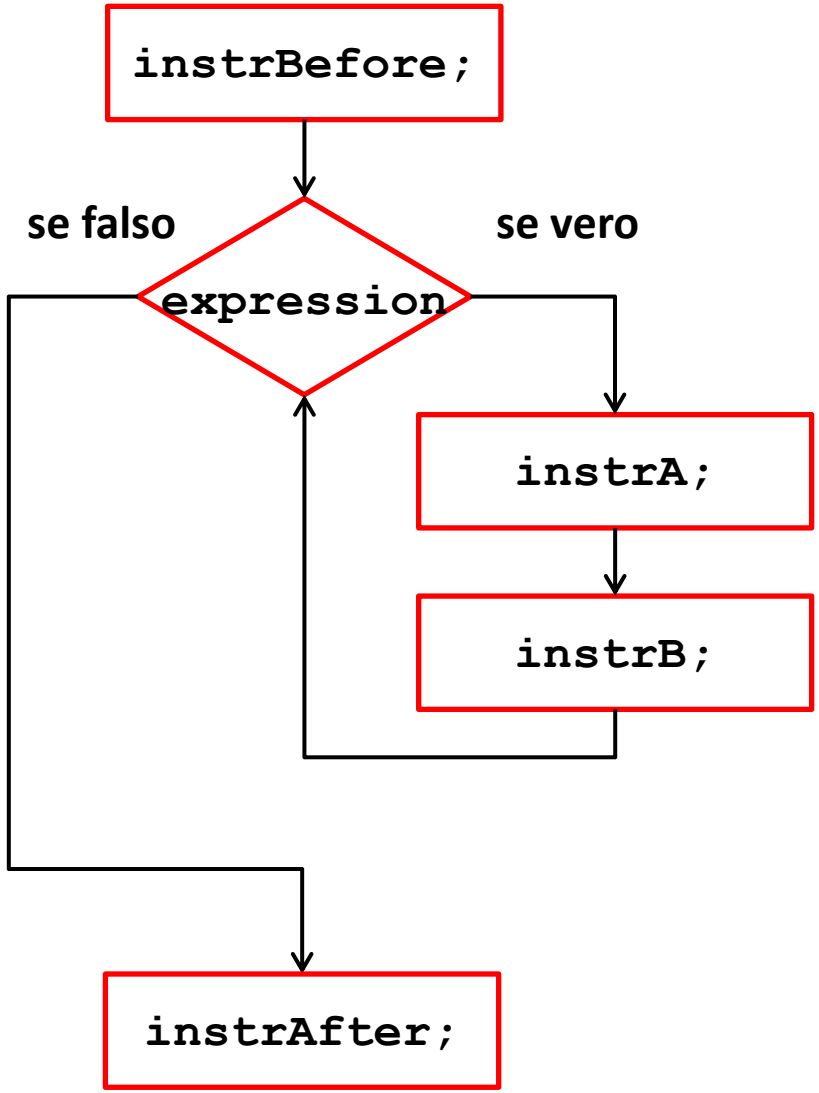
- Aspetta a casa

Vado in stazione

Sali sul treno

Arrivi a lezione, wow

# Costrutto Iterativo

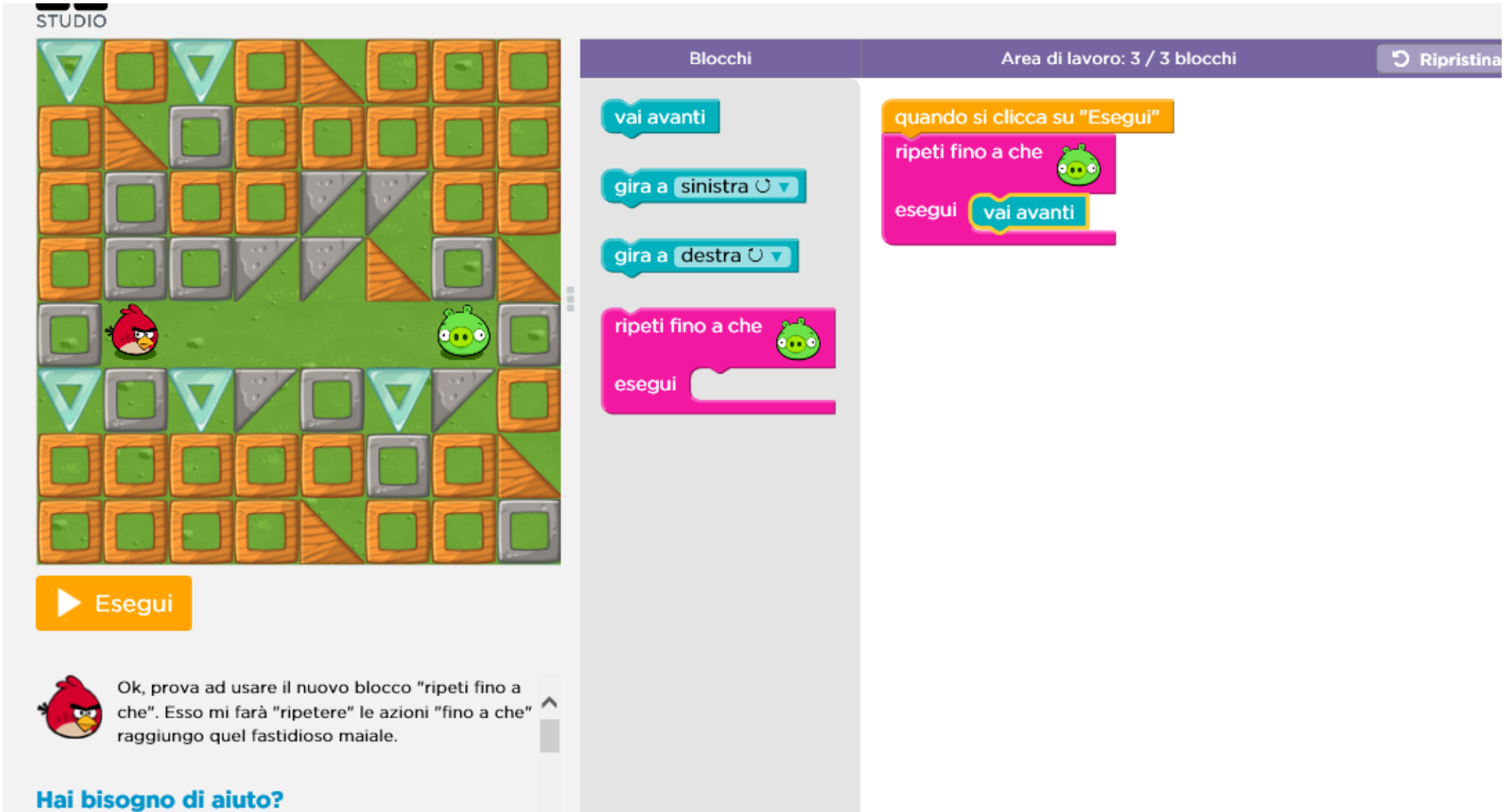



Se **`expression`** è vera eseguo il ramo di istruzioni contenente **`instrA;`** e **`instrB;`** (corpo del ciclo)

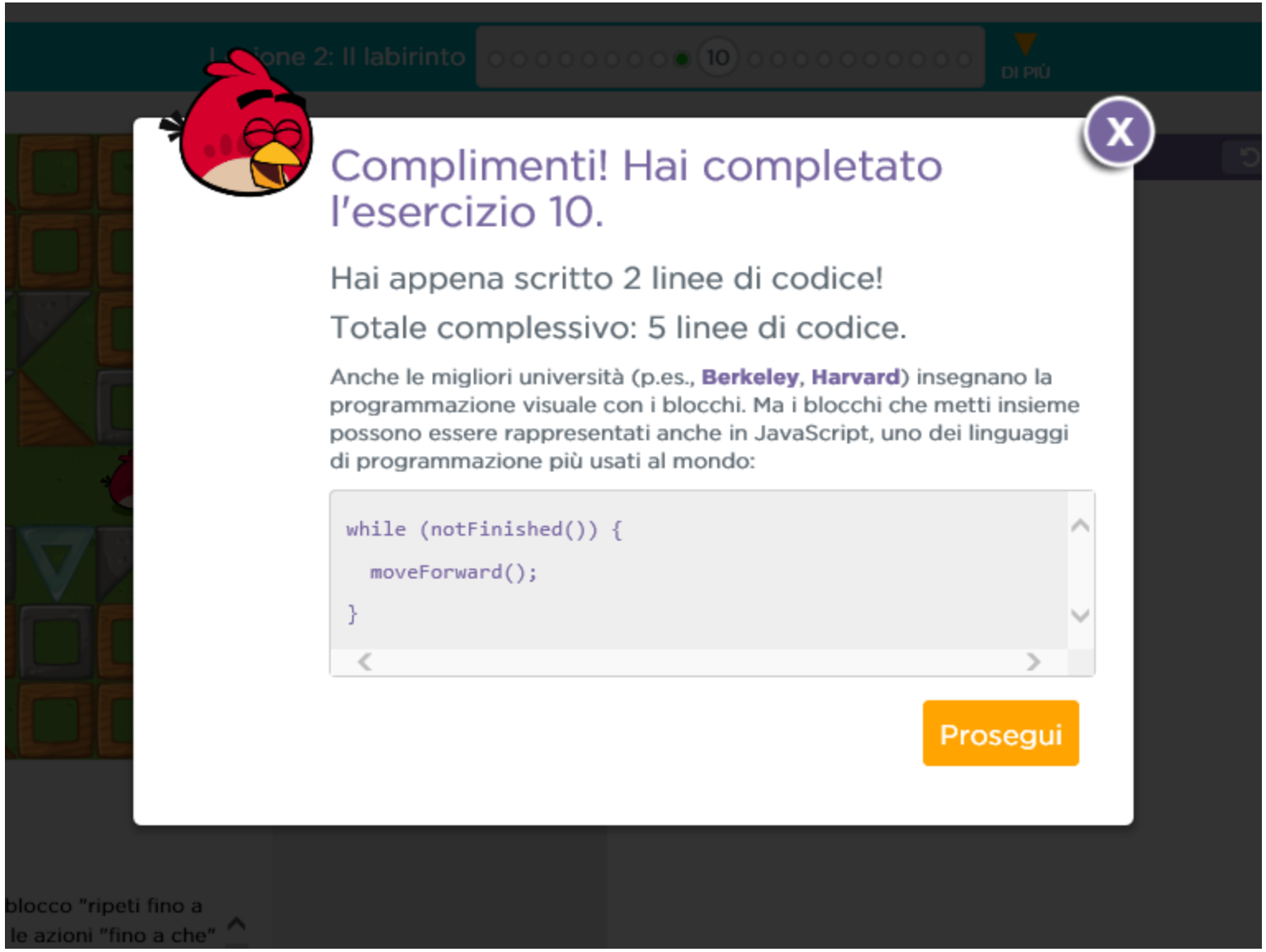
Al termine valuta nuovamente **`expression`**

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

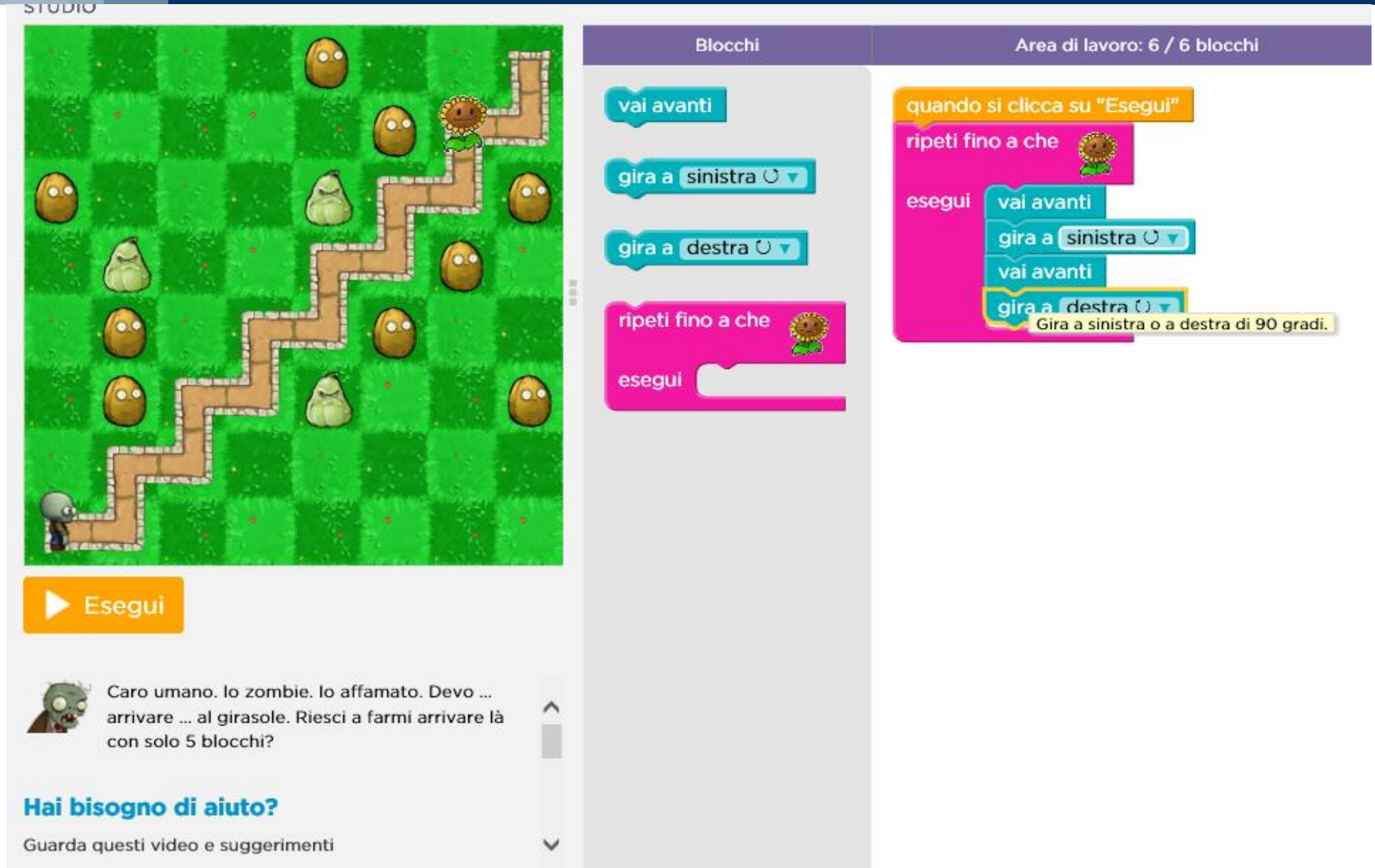
# Il Costrutto Iterativo su code.org



# Il Costrutto Iterativo su code.org



# Il costrutto iterativo su code.org



# Il costrutto iterativo su code.org

# Esempi di Algoritmi

# Il Pallottoliere

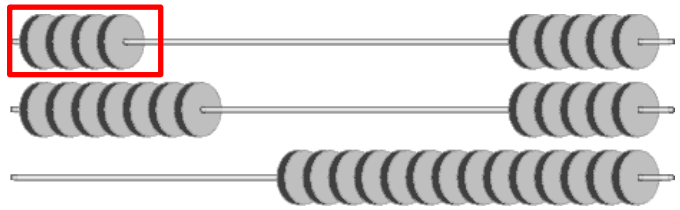
Supponiamo di avere un bambino che sa contare ma non sa fare operazioni aritmetiche

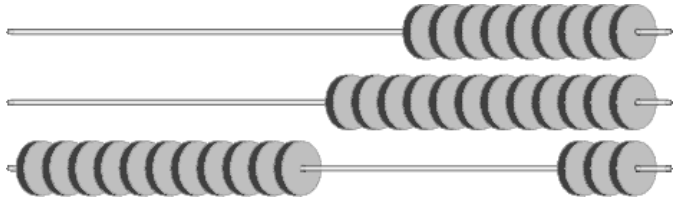
Scriviamo le istruzioni per utilizzare il pallottoliere (utilizzando un corstrutto iterativo)

# Algoritmo per sommare col pallottoliere

Supponiamo che:

- Primo addendo sia riportato sulla prima riga a sinistra
- Secondo addendo sia sulla seconda riga a sinistra
- Risultato deve apparire nella terza riga (e che all'inizio questa abbia tutte le palline a destra).
- Operiamo con numeri «piccoli», non c'è bisogno del riporto

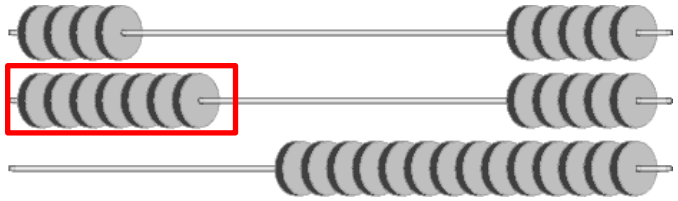


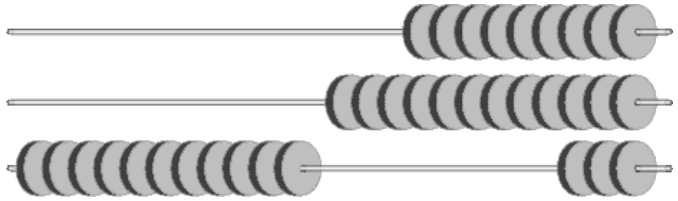


# Algoritmo per sommare col pallottoliere

Supponiamo che:

- Primo addendo sia riportato sulla prima riga a sinistra
- Secondo addendo sia sulla seconda riga a sinistra
- Risultato deve apparire nella terza riga (e che all'inizio questa abbia tutte le palline a destra).
- Operiamo con numeri «piccoli», non c'è bisogno del riporto

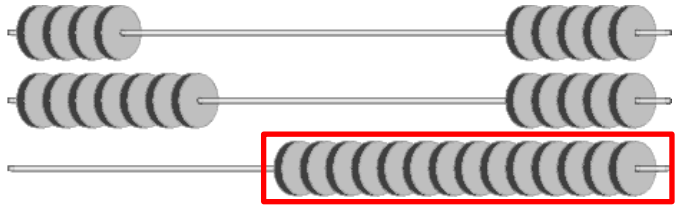


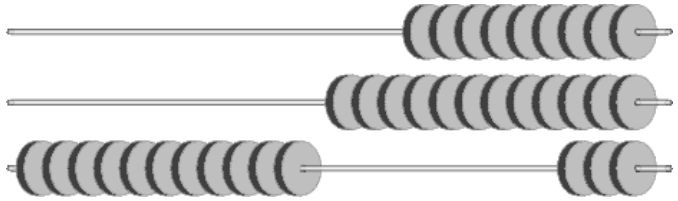


# Algoritmo per sommare col pallottoliere

Supponiamo che:

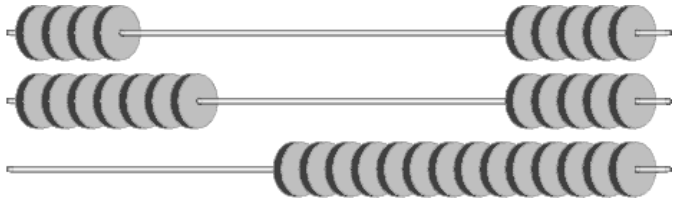
- Primo addendo sia riportato sulla prima riga a sinistra
- Secondo addendo sia sulla seconda riga a sinistra
- Risultato deve apparire nella terza riga (e che all'inizio questa abbia tutte le palline a destra).
- Operiamo con numeri «piccoli», non c'è bisogno del riporto

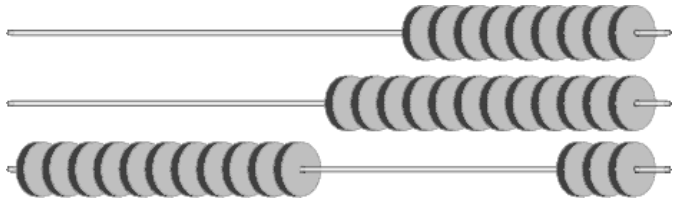




# Algoritmo del Pallottoliere

1. Sposta una pallina da sinistra a destra nella prima riga, al contempo da destra a sinistra nella terza riga
2. **Ripeti** operazione precedente **fino a** svuotare parte sinistra prima riga
3. Sposta pallina da sinistra a destra nella seconda riga, al contempo da destra a sinistra nella terza.
4. **Ripeti** operazione precedente **fino a** svuotare parte sinistra seconda riga
5. «Conta» quante palline trovi sulla terza riga.





# Analisi dell'Algoritmo del Pallottoliere

**Correttezza**:

- l'algoritmo risolve il compito senza errori o difetti?

**Efficienza**:

- l'algoritmo usa risorse in modo minimale (o almeno ragionevole)?

# Altri Esempi:

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

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

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

Algoritmo per ricercare i libri in biblioteca

# 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 (con le variabili a volte non occorre il bicchiere C)**

# Altri Esempi:

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

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

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

Algoritmo per ricercare i libri in biblioteca

# Esempio: ricerca del prodotto migliore

1. Prendi in mano il primo prodotto: assumi che sia il migliore
2. Procedi fino al prossimo prodotto
3. Confrontalo con quello che hai in mano
4. **Se** il prodotto davanti a te è migliore: 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 migliore.

**Algoritmo per trovare il 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 buono nella corsia del supermercato, passando per la corsia una sola volta

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

Algoritmo per ricercare i libri in biblioteca

# 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.

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

## Altri Esempi:

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

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

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

Algoritmo per ricercare i libri in biblioteca

# Gestione di una biblioteca

Si supponga di avere una biblioteca gestita mediante archivio cartaceo.

Ogni libro ha una data posizione identificata da SCAFFALE e POSIZIONE

Esiste un archivio ordinato che contiene, per ogni libro un foglio del tipo:

AUTORE / I:
GHEZZI CARLO,
JAZAYERI MEHDI.

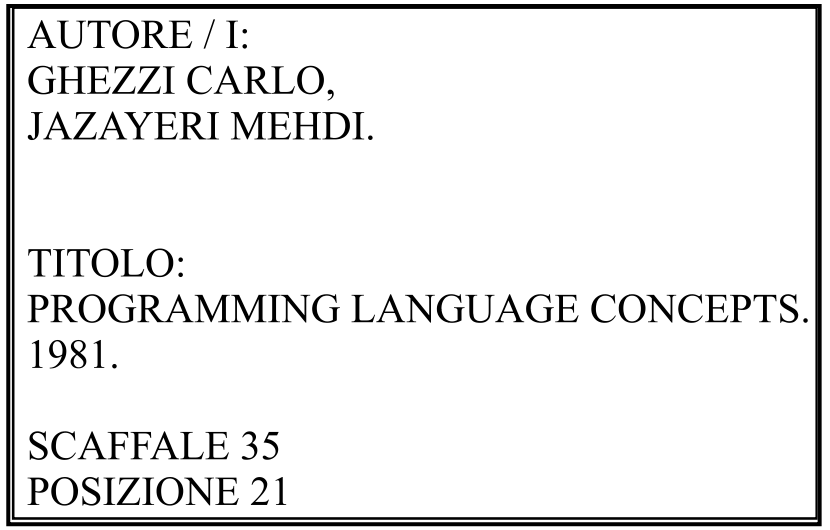
TITOLO:
PROGRAMMING LANGUAGE CONCEPTS.
1981.

SCAFFALE 35
POSIZIONE 21

# Gestione di una biblioteca

Algoritmo per trovare un libro

1. **ricerca** la scheda del libro nello schedario
2. trovata la scheda, **segna** su un **foglio** numero scaffale e posizione del libro
3. raggiungi lo scaffale indicato
4. individuato lo scaffale, **ricerca** la posizione del libro
5. Prendi il libro



# Algoritmo: Ricerca

Non tutti gli esecutori sono in grado di «cercare», e comunque la ricerca può essere fatta in diversi modi

1. esamina la **prima** scheda dello schedario
2. **se** autore e titolo coincidono con quelli cercati
   - ricerca **conclusa** con successo

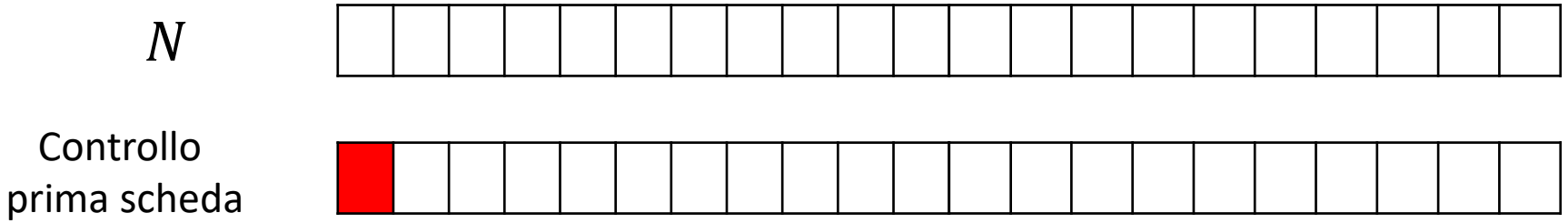
   **altrimenti** passa a scheda successiva
3. **Ripeti** istruzione **2, fino a conclusione o fino a raggiungere l'ultima scheda**
4. **se** trovata ⇒ ricerca conclusa con successo
   - **altrimenti** ⇒ ricerca conclusa con insuccesso

# Algoritmo: Ricerca

Assumendo che l'archivio abbia $N$ schede:

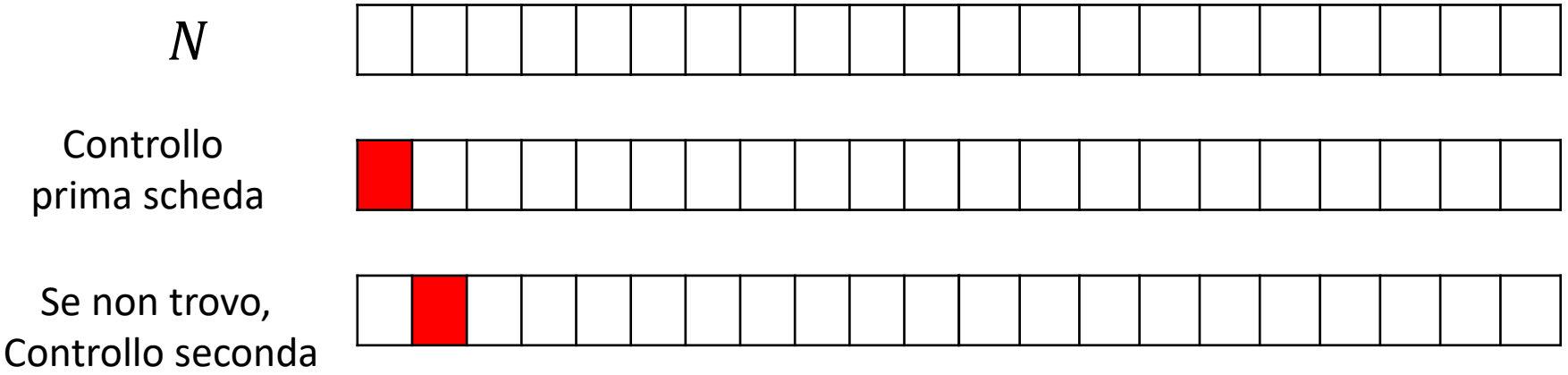
# Algoritmo: Ricerca

Assumendo che l'archivio abbia $N$ schede:



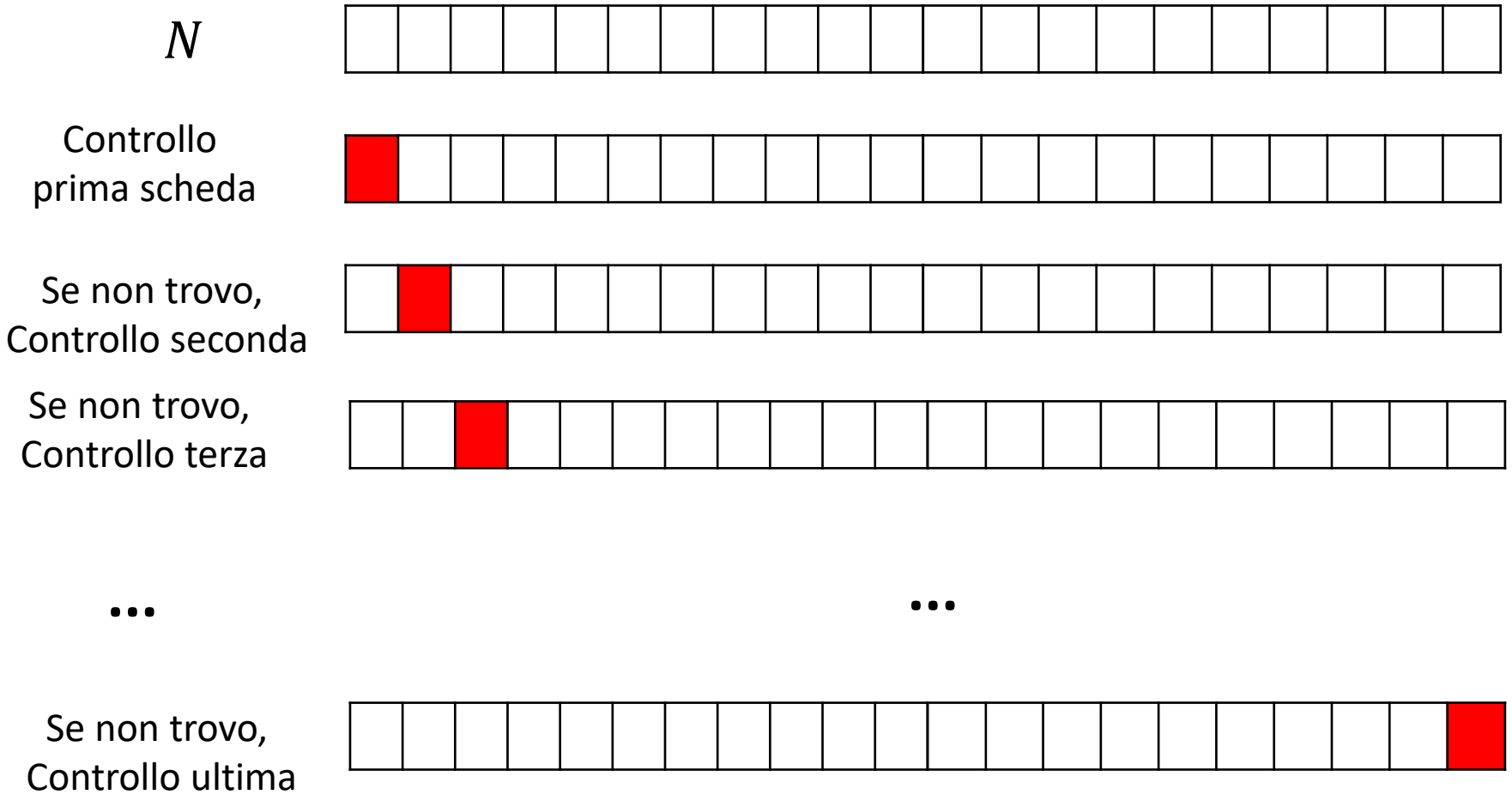
# Algoritmo: Ricerca

Assumendo che l'archivio abbia $N$ schede:



# Algoritmo: Ricerca

Assumendo che l'archivio abbia $N$ schede:



# Algoritmo: Ricerca

Non tutti gli esecutori sono in grado di «cercare», e comunque la ricerca può essere fatta in diversi modi

1. esamina la **prima** scheda dello schedario
2. **se** autore e titolo coincidono con quelli cercati
   - ricerca **conclusa** con successo

   **altrimenti** passa a scheda successiva
3. **Ripeti** istruzione **2, fino a conclusione o fino a raggiungere l'ultima scheda**
4. **se** trovata ⇒ ricerca conclusa con successo
   - **altrimenti** ⇒ ricerca conclusa con insuccesso

Ricerca semplice ma **inefficiente**: non sfrutta l'ordinamento delle schede nell'archivio

# Algoritmo: Ricerca tra elementi ordinati

Lo schedario contiene $N$ schede ordinate in cui cercare

1. prendi la scheda centrale, i.e. alla posizione $N/2$.
2. **se** è la scheda cercata, termina con successo.

**altrimenti,**

3. **se** la scheda cercata segue alfabeticamente quella esaminata,
    - continua la **ricerca** nella seconda metà dello schedario
    - **altrimenti** continua la **ricerca** nella prima metà

# Algoritmo: Ricerca tra elementi ordinati

Lo schedario contiene $N$ schede ordinate in cui cercare

1. prendi la scheda centrale, ovvero alla posizione $N/2$
2. **se** è la scheda cercata, termina con successo

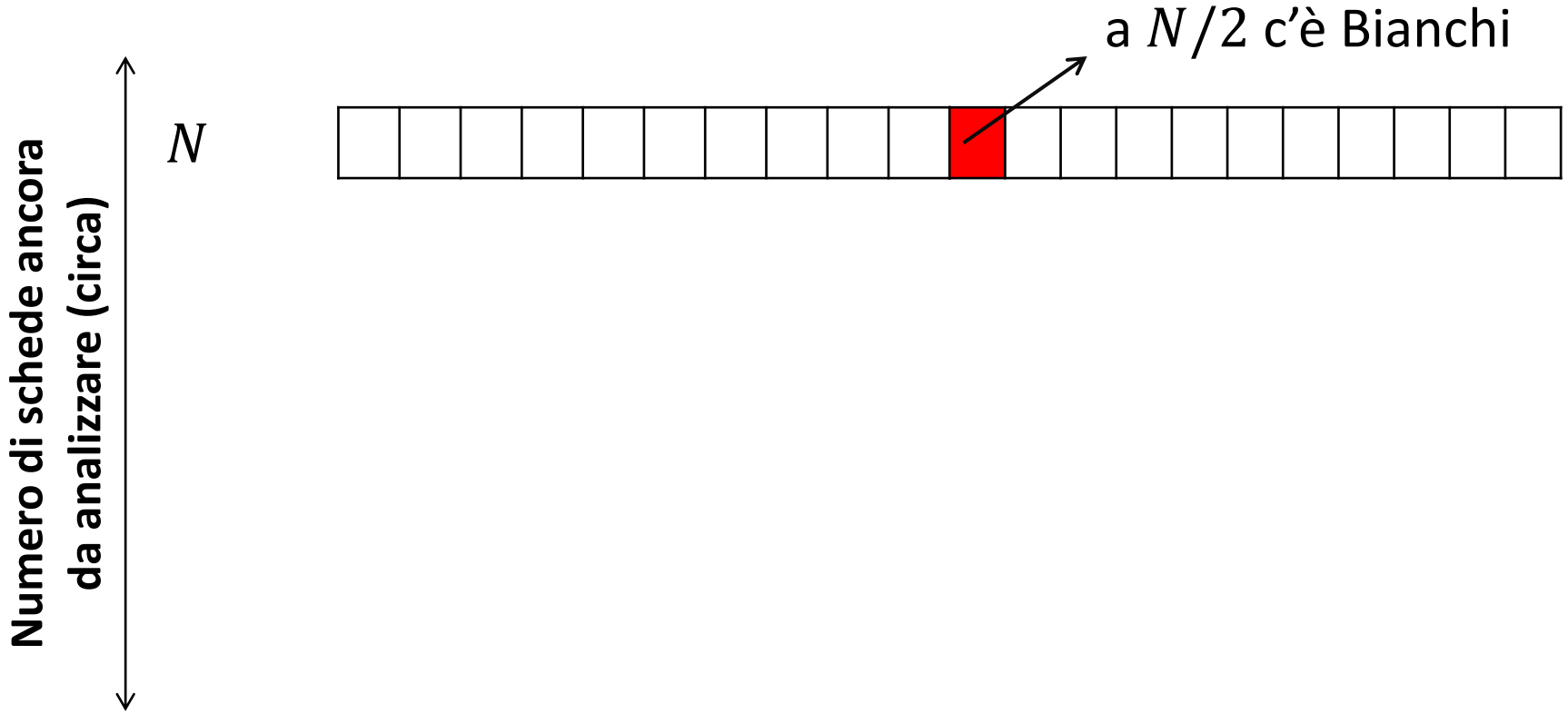
**altrimenti,**

3. **se** la scheda cercata segue alfabeticamente quella esaminata,
    - continua la **ricerca** nella seconda metà dello schedario
    - **altrimenti** continua la **ricerca** nella prima metà

**Questo algoritmo** ricerca **è ricorsivo, perché** richiama se stesso (riduce però il numero di schede da analizzare ed esiste una condizione per cui non chiama se stesso)

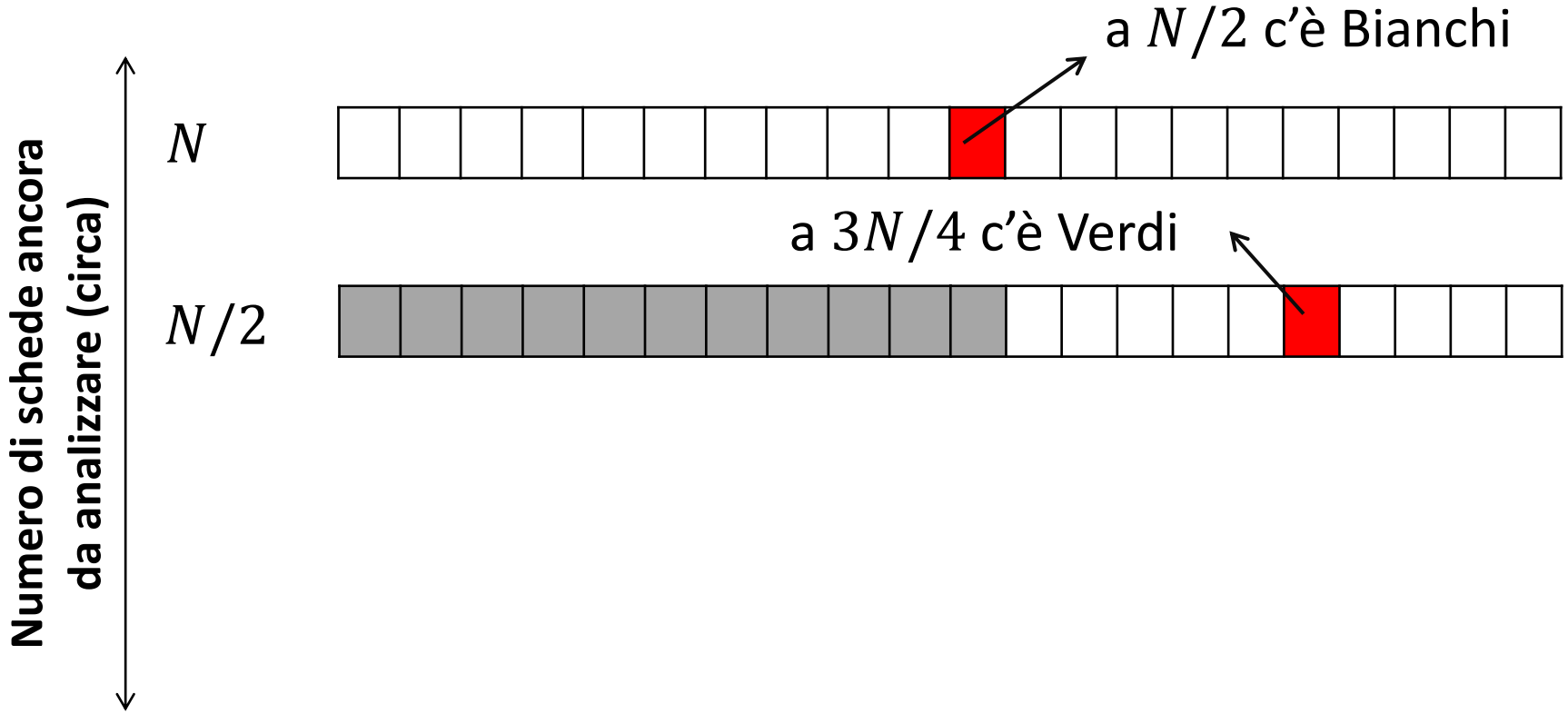
# Algoritmo: Ricerca tra elementi ordinati

es: Ricerca dell'autore ROSSI



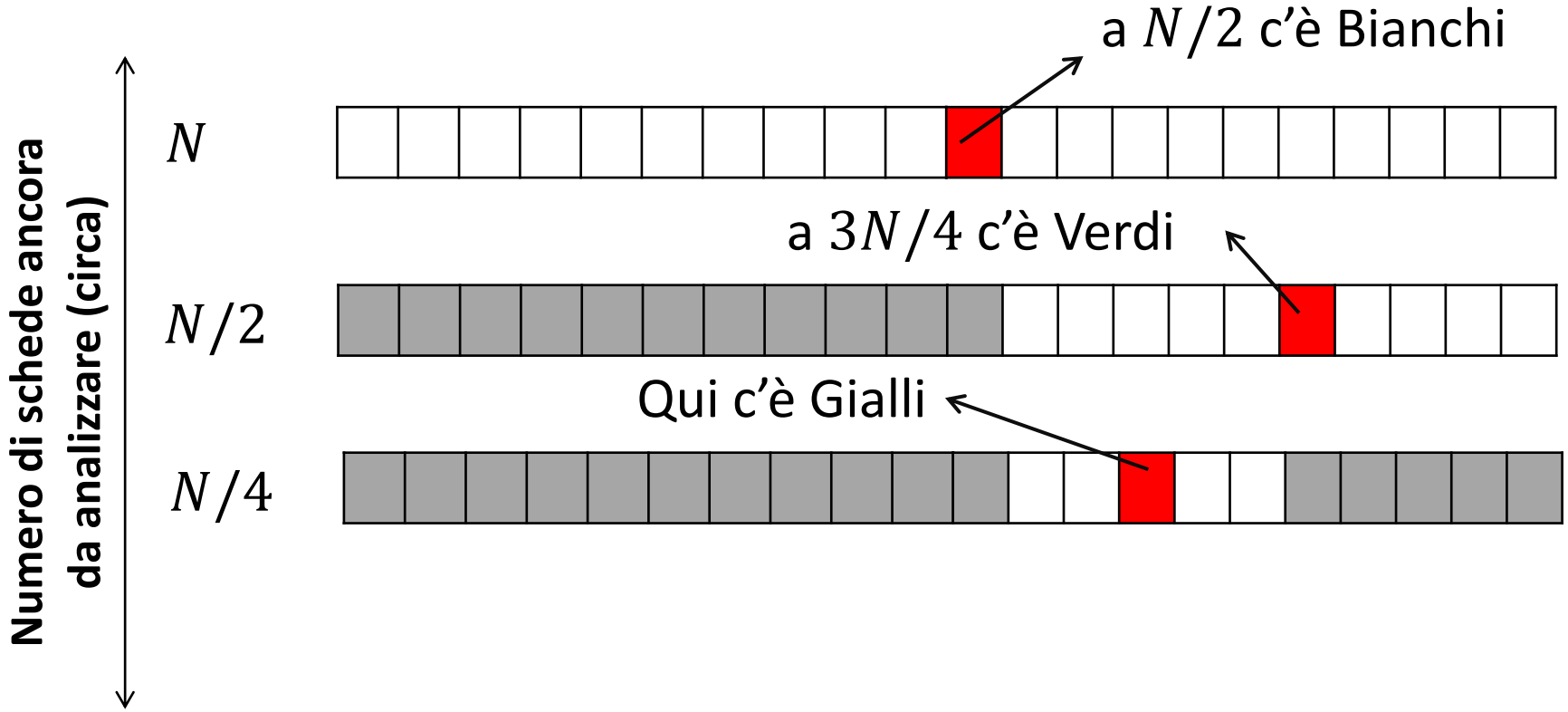
# Algoritmo: Ricerca tra elementi ordinati

es: Ricerca dell'autore ROSSI



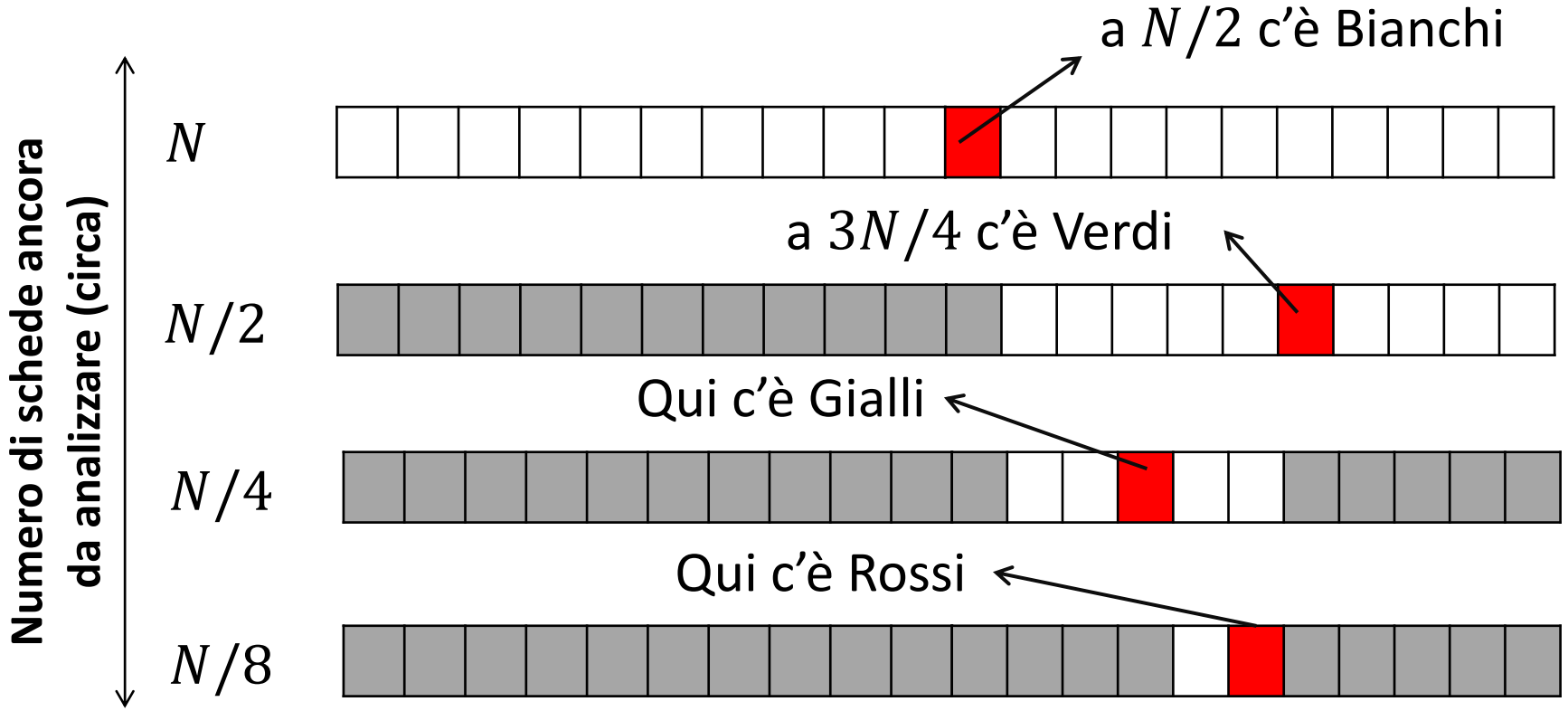
# Algoritmo: Ricerca tra elementi ordinati

es: Ricerca dell'autore ROSSI



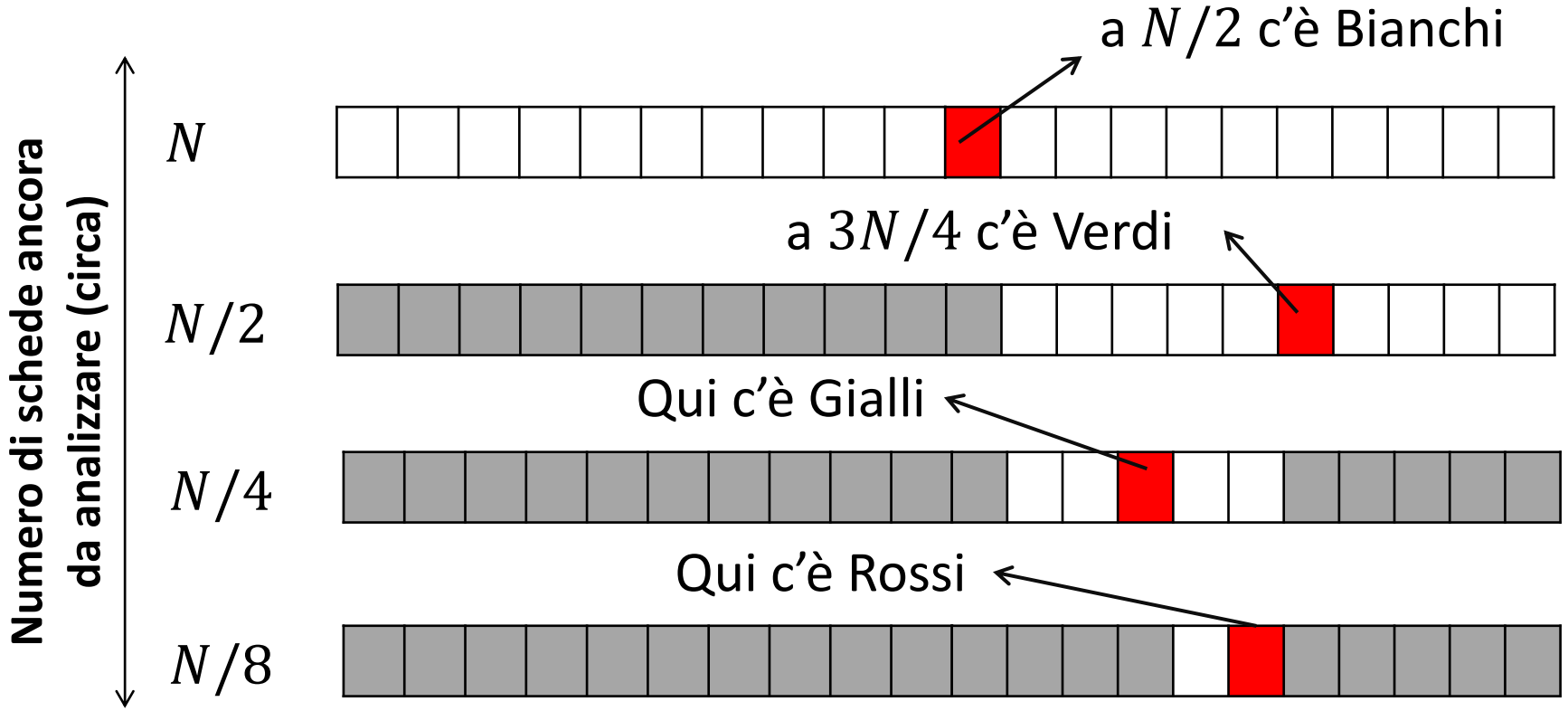
# Algoritmo: Ricerca tra elementi ordinati

es: Ricerca dell'autore ROSSI



# Algoritmo: Ricerca tra elementi ordinati

es: Ricerca dell'autore ROSSI




**N.B**: cosa succede se non esiste il nome cercato nell'archivio?
L'algoritmo deve terminare anche in questo caso!

# Algoritmo: Ricerca tra elementi ordinati

Lo schedario contiene $N$ schede ordinate in cui cercare

1. prendi la scheda centrale, i.e. alla posizione $N/2$.
2. **se** è la scheda cercata, termina con successo.

**altrimenti**,

3. **se** la scheda cercata segue alfabeticamente quella esaminata,
    - continua la **ricerca** nella seconda metà dello schedario
    - **altrimenti** continua la **ricerca** nella prima metà

Cosa devo modificare per far terminare l'algoritmo quando la zona di ricerca è vuota?

# Algoritmo: Ricerca tra elementi ordinati

Lo schedario contiene $N$ schede ordinate in cui cercare

1. **se** $N$ è zero (porzione di schedario vuota) allora termina la ricerca, **altrimenti**, prendi la scheda alla posizione $N/2$
2. **se** è la scheda cercata termina con successo

**altrimenti**,

3. **se** la scheda cercata segue alfabeticamente quella esaminata:
    - continua la **ricerca** nella seconda metà dello schedario
    - **altrimenti** continua la **ricerca** nella prima metà

# Intermezzo: Cosa fa questo algoritmo?

1. Alzatevi tutti in piedi
2. Ognuno di voi vale 1
3. **Ripeti:** Ciascuno cerca un compagno/a ancora in piedi
4. **Se** non avete trovato un compagno, il vostro valore non cambia e dovete restare in piedi
5. **Altrimenti** ogni coppia somma i loro valori, il risultato è il nuovo valore di ciascuno
6. Uno dei due si deve sedere e l'altro deve restare in piedi
7. Ricominciate dal punto 3, **finché** non resta in piedi una sola persona in tutta la stanza
8. Il valore dell'ultima persona rimasta in piedi è

# Intermezzo: Cosa fa questo algoritmo?

1. Alzatevi tutti in piedi
2. Ognuno di voi vale 1
3. **Ripeti:** Ciascuno cerca un compagno/a ancora in piedi
4. **Se** non avete trovato un compagno, il vostro valore non cambia e dovete restare in piedi
5. **Altrimenti** ogni coppia somma i loro valori, il risultato è il nuovo valore di ciascuno
6. Uno dei due si deve sedere e l'altro deve restare in piedi
7. Ricominciate dal punto 3, **finchè** non resta in piedi una sola persona in tutta la stanza
8. Il valore dell'ultima persona rimasta in piedi è il numero di persone presenti nella stanza

# I Programmi

Dall’algoritmo al programma

# 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

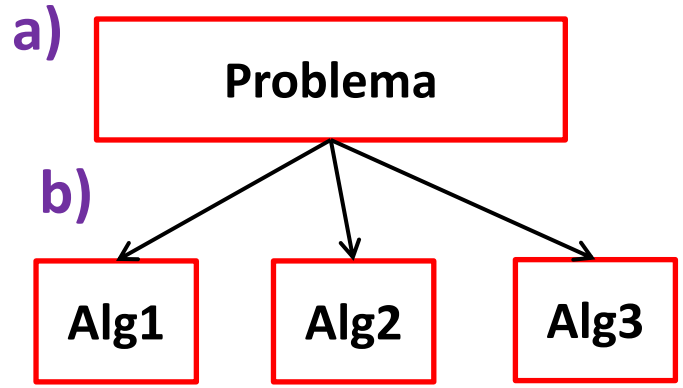
# Riepilogando: Come Procedere

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

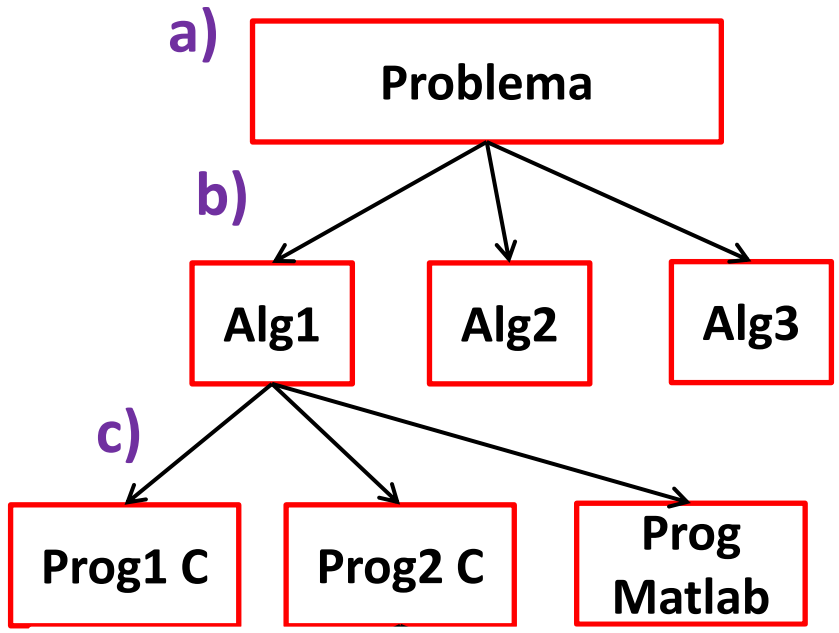


# Riepilogando: Come Procedere

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,



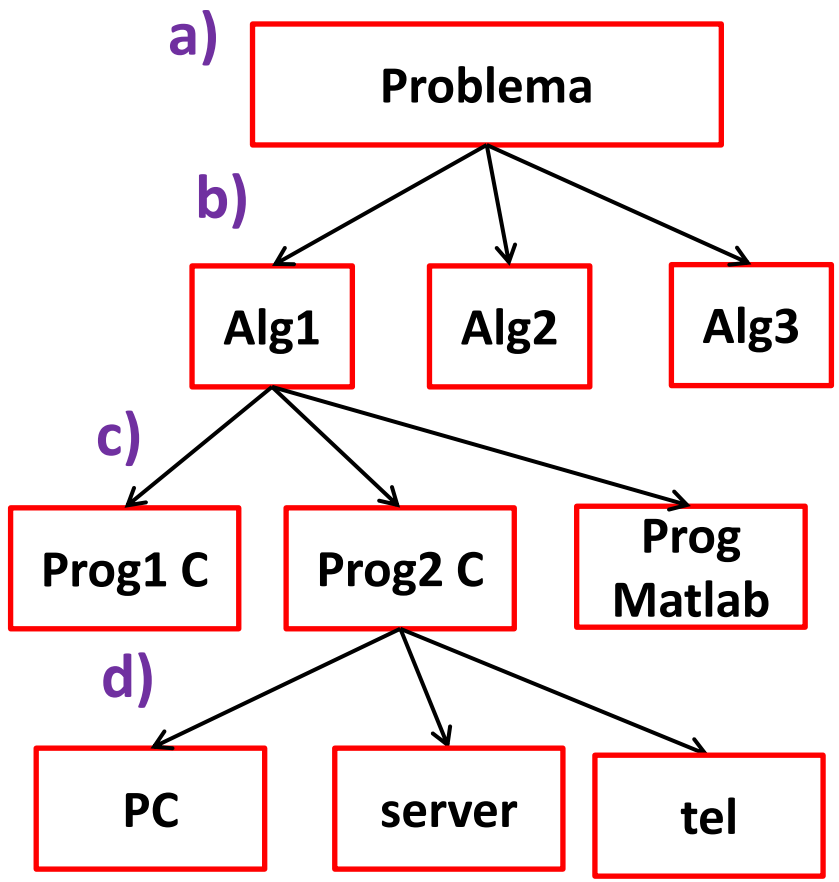
# Riepilogando: Come Procedere

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



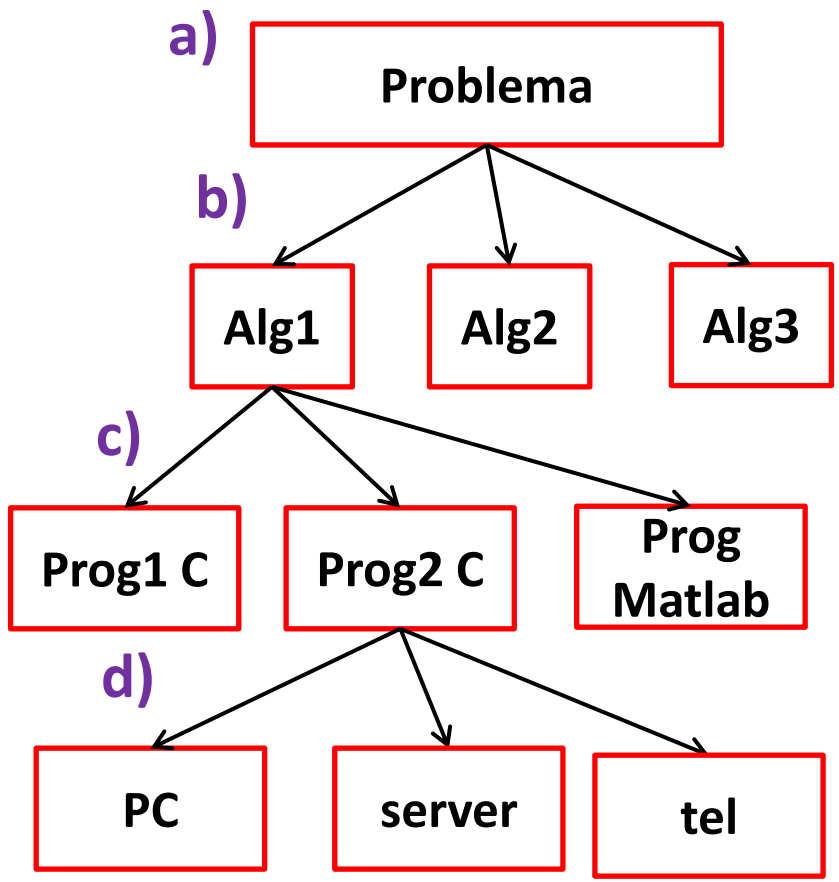
# Riepilogando: Cosa Fare

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 (nel vostro caso l'interprete Matlab)



# La Rappresentazione dell'Informazione

# Dati che Vogliamo Rappresentare

Quando cerchiamo di risolvere un problema quali sono i dati che pensiamo possano essere utilizzati più spesso:

- Numeri
- Caratteri alfanumerici
- Immagini
- Suoni
- Video

# Codifica dei Numeri in Base 10

Le cifre che abbiamo a disposizione sono 10

$$A_{10} = \{0, 1, \dots, 9\}$$

Utilizziamo una **codifica posizionale**, quindi le cifre in posizioni differenti hanno un significato differente

- Es numero di 4 cifre
  - $\mathbf{3401} = \mathbf{3} \times 10^3 + \mathbf{4} \times 10^2 + \mathbf{0} \times 10^1 + \mathbf{1} \times 10^0$

Con $m$ cifre posso rappresentare $10^m$ numeri distinti:

$$0, \dots, 10^m - 1$$

# Codifica dei Numeri in una Base Qualsiasi

Ogni codifica ha un insieme di cifre (dizionario) $A$

- In base 10, il dizionario è $A_{10} = \{0, \dots, 9\}$

Un numero è una sequenza di $m$ cifre

$$a_{m-1} \dots a_1 a_0 \text{ con } a_i \in A$$

- 8522 è una sequenza di 4 cifre di $A_{10}$, $\{8,5,2\} \subset A_{10}$

Manteniamo un **codifica posizionale**: ogni cifra assume un significato diverso in base alla sua posizione nel numero:

- $a_n$ è la cifra **più significativa**
- $a_0$ è la cifra **meno significativa**

*Es:* in **8**52**2**, **8** è la cifra più significativa, **2** quella meno

8522 è diverso da 2852, 8252,… che pur contengono le stesse cifre

# Codifica dei Numeri: Notazione Posizionale

Dato un numero $N_{10}$, in base 10 contenente $m$ cifre scritto come $a_{m-1}a_{m-2}\ldots a_1a_0$ questo corrisponde a:

$$N_{10} = a_{m-1} \times 10^{m-1} + a_{m-2} \times 10^{m-2} + \ldots + a_0 \times 10^0$$

$$(a_{m-1}a_{m-2}\ldots a_1a_0)_{10} = \sum_{i=0}^{m-1} a_i \times 10^i \ , \qquad a_i \in A_{10}$$

*Es:* $(8522)_{10} = 8 \times 10^3 + 5 \times 10^2 + 2 \times 10^1 + 2 \times 10^0$

Con $m$ cifre in $A_{10}$ quanti numeri posso esprimere? $10^m$

Considerando gli interi positivi, posso scrivere tutti numeri tra $[0, 10^m - 1]$

- *Es:* $m = 1$ copro $[0,10 - 1]$ (cioè $[0,9]$ i.e., $A_{10}$)

  $m$ =3 copro $[0,10^3 - 1]$ (cioè $[0, 999]$)

# Rappresentazioni Posizionali in Base $p$

Consideriamo **rappresentazioni posizionali in base $p$** (con $p > 0$) e chiamiamo $A_p$ il dizionario di $p$ cifre:

- se $p \leq 10$ prendiamo le cifre di $A_{10}$, $A_p = \{0, \dots, p-1\}$
- se $p > 10$ aggiungiamo simboli $A_p = \{0, \dots, 9, A, B \dots\}$

Un numero di $m$ cifre in base $p$:

$$N_p = a_{m-1} \times p^{m-1} + a_{m-2} \times p^{m-2} + \dots + a_0 \times p^0$$

$$N_p = a_{m-1}a_{m-2} \dots a_1 a_0 = \sum_{i=0}^{m-1} a_i \times p^i \,, \qquad a_i \in A_p$$

Con $m$ cifre in $A_p$ quanti numeri posso esprimere: $p^m$

Considerando gli interi positivi, posso scrivere tutti numeri tra $[0, p^m - 1]$

# Codifica dei numeri in base $p$: Esempi

*Es:* $m = 1$ e $p = 7$, copro $[0, 7 - 1]$ (cioè $[0, 6]$)

$m = 4$ e $p = 7$, copro $[0, 7^4 - 1]$ (cioè $[0, 2400]$)

$m = 1$ e $p = 13$, copro $[0, 13 - 1]$ (cioè $[0, 12]$)

$m = 4$ e $p = 13$, copro $[0, 13^4 - 1]$ (cioè $[0, 28560]$)

Nota: Al crescere di $p$ cresce il «potere espressivo» del dizionario (con lo stesso numero di cifre posso scrivere molti più numeri)

# Le targhe

Quante diverse targhe italiane ci possono essere (ignoriamo targhe speciali come CC, CRI, EI, …)?


Lettere in uso: 24 (cioè 26 – 2, I e O non vengono utilizzate)

N° di targe : $24^4 * 10^3 = 331776000$

Se usassi numeri in base 10: 9 cifre

Se usassi numeri in base 2 $\approx \log_2(331776000) = 23$ cifre

# Codifica dei Numeri in Base 2

I calcolatori sono in grado di operare con informazioni **binarie**. Quindi $p = 2$ e $A_2 = \{0, 1\}$

$$N_2 = a_{m-1} \times 2^{m-1} + a_{m-2} \times 2^{m-2} + .. + a_0 \times 2^0$$

$$N_2 = a_{m-1}a_{m-2} \dots a_1 a_0 = \sum_{i=0}^{m-1} a_i \times 2^i \ , \qquad a_i \in \{0,1\}$$

Un bit (*binary digit*) assume valore 0/1 corrispondente ad un determinato *stato fisico* (alta o bassa tensione nella cella di memoria)

Con $m$ bit posso scrivere $2^m$ numeri diversi, ad esempio tutti gli interi nell'intervallo $[0, 2^m - 1]$

Il byte è una sequenza di 8 bit ed esprime $2^8 = 256$ numeri diversi (ad esempio gli interi in $[0,255]$)

00000000, 00000001, 00000010, …, 11111111

# Le potenze del 2

È necessario imparare le potenze di 2!

| $2^0$ | $2^1$ | $2^2$ | $2^3$ | $2^4$ | $2^5$ | $2^6$ | $2^7$ | $2^8$ | $2^9$ | $2^{10}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 |

E i loro legami con l'informatica:

- Byte = $8$ bit
- KiloByte (kB) = $10^3$ Byte
- MegaByte (MB) = $10^6$ Byte
- GigaByte (GB) = $10^9$ Byte
- TheraByte (TB) = $10^{12}$ Byte

# Altre Codifiche che consideriamo

Codifica **ottale** (in **base 8**)

- $A_8 = \{0, 1, \dots, 7\}$
- con $m$ cifre in $A_8$ scrivo i numeri da $[0, 8^m - 1]$

Codifica **esadecimale**, (in **base 16**)

- $A_{16} = \{0, 1, \dots, 9, A, B, C, D, E, F\}$,
- Per le conversioni $A = 10$, …, $F = 15$.
- con $m$ cifre in $A_{16}$ scrivo i numeri da $[0,16^m - 1]$

# Rappresentazione dei Caratteri

Ogni carattere viene mappato in un numero intero (che è espresso da sequenza di bit) utilizzando dei codici

Il codice più usato è l'*ASCII (American Standard Code for Information Interchange)* a 8 bit che contiene:

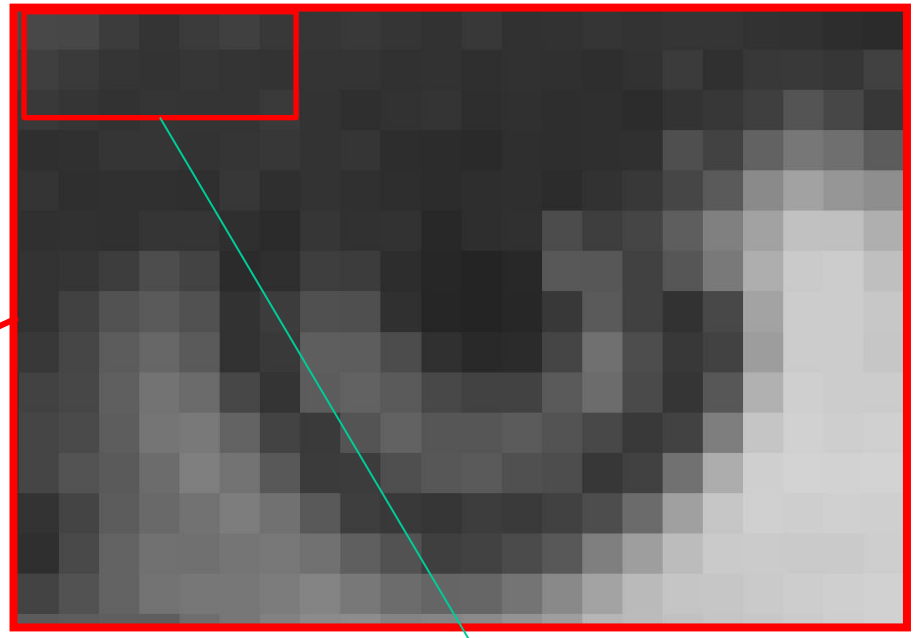
- Caratteri alfanumerici
- Caratteri simbolici (es. punteggiatura, e caratteri speciali @&%$)
- Caratteri di comando (es. termina riga, vai a capo, tab)

# La codifica ASCII (parziale)

| DEC | CAR | DEC | CAR | DEC | CAR | DEC | CAR | DEC | CAR |
|---|---|---|---|---|---|---|---|---|---|
| 48 | 0 | 65 | A | 75 | K | 97 | a | 107 | k |
| 49 | 1 | 66 | B | 76 | L | 98 | b | 108 | l |
| 50 | 2 | 67 | C | 77 | M | 99 | c | 109 | m |
| 51 | 3 | 68 | D | 78 | N | 100 | d | 110 | n |
| 52 | 4 | 69 | E | 79 | O | 101 | e | 111 | o |
| 53 | 5 | 70 | F | 80 | P | 102 | f | 112 | p |
| 54 | 6 | 71 | G | 81 | Q | 103 | g | 113 | q |
| 55 | 7 | 72 | H | 82 | R | 104 | h | 114 | r |
| 56 | 8 | 73 | I | 83 | S | 105 | i | 115 | s |
| 57 | 9 | 74 | J | 84 | T | 106 | j | 116 | t |
| | | | | 85 | U | | | 117 | u |
| | | | | 86 | V | | | 118 | v |
| | | | | 87 | W | | | 119 | w |
| | | | | 88 | X | | | 120 | x |
| | | | | 89 | Y | | | 121 | y |
| | | | | 90 | Z | | | 122 | z |

# Codifica delle Immagini

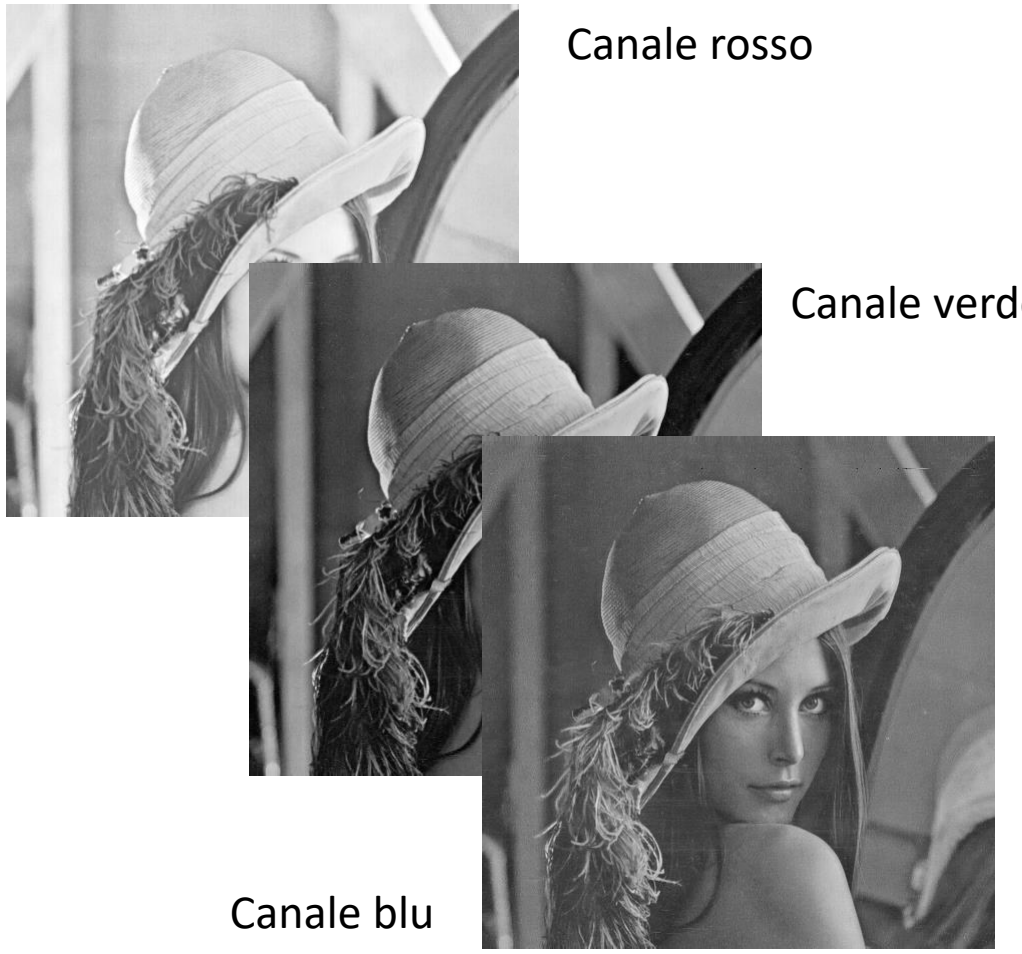
Le immagini nei calcolatori sono digitali, i.e. tabella di pixel, ciascuno caratterizzato da uno o più valori di intensità.



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 123 | 122 | 134 | 121 | 132 | 133 | 145 | 134 |
| 122 | 121 | 125 | 132 | 124 | 121 | 116 | 126 |
| 119 | 127 | 137 | 119 | 139 | 127 | 128 | 131 |

# Codifica delle Immagini

Le immagini nei calcolatori sono digitali, i.e. tabella di pixel, ciascuno caratterizzato da uno o più valori di intensità.



Canale rosso

Canale verde

Canale blu

# Codifica delle Immagini

Esistono diverse codifiche dell'immagine, non sempre questa viene scritta pixel per pixel

È spesso conveniente rappresentare l'immagine secondo codifiche che permettano di ridurre le dimensioni

Codifiche ***lossless***: permettono, senza perdita di informazione, di comprimere l'immagine

- e.g., non serve ripetere il valore nelle aree costanti, conviene registrare le variazioni (e.g., gif, png)

Codifiche ***lossy****:* comportano una perdita di informazione e di qualità

- e.g., le immagini jpeg sono compresse a blocchi, ogni blocco contiene meno dettagli dell'immagine originale

# Convertire in base 10

da base 2 (o altre basi) a base 10

# Conversione Binario-Decimale

Utilizziamo la definizione di numero in notazione posizionale

$$N_2 = a_{m-1} \times 2^{m-1} + a_{m-2} \times 2^{m-2} + .. + a_0 \times 2^0$$

*Es.*

$$(101)_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = (5)_{10}$$

$$(1100010)_2 = 1 \times 2^6 + 1 \times 2^5 + 1 \times 2 = (98)_{10}$$

# Osservazioni

In binario i numeri che terminano con 1 sono dispari, quelli con 0 sono pari.

- L'unico modo per avere un numero dispari nella somma è aggiungere $2^0 = 1$

Le conversioni di numeri con bit tutti a 1 si calcolano facilmente

$$(111111)_2 = (1000000)_2 - (1)_2 = 2^6 - 1 =$$

## Conversioni ottale/esadecimale  decimale ➡

È possibile utilizzare le definizioni precedenti per convertire da ottale/esadecimale in base 10

$$N_{16} = a_{m-1}a_{m-2} \dots a_1 a_0 = \sum_{i=0}^{m-1} a_i \times 16^i \ , \qquad a_i \in A_{16}$$

$$(31)_8 = 3 * 8 + 1 = (25)_{10}$$

$$(A170)_{16} = A * 16^3 + 1 * 16^2 + 7 * 16$$

$$= 10 * 4096 + 1 * 256 + 7 * 16 = (41328)_{10}$$

$$(623)_8 = 6 * 8^2 + 2 * 8 + 3 * 8^0$$

$$= 6 * 64 + 16 + 3 = (403)_{10}$$

$$(623)_{16} = 6 * 16^2 + 2 * 16 + 3 * 16^0$$

$$= 6 * 256 + 32 + 3 = (1571)_{10}$$

# Convertire in base 2

da base 10 (o altre basi) a base 2

# Conversione Decimale ➡ Binario

Metodo delle divisioni successive:

Per convertire 531 opero come segue:

- 531 / 2 = 265 + 1
- 265 / 2 = 132 + 1
- 132 / 2 = 66 + 0
- 66 / 2 = 33 + 0
- 33 / 2 = 16 + 1
- 16 / 2 = 8 + 0
- 8 / 2 = 4 + 0
- 4 / 2 = 2 + 0
- 2 / 2 = 1 + 0
- 1 / 2 = 0 + 1

Divisione intera tra il numero e 2

Il risultato della divisione precedente viene successivamente diviso

Si continua fino a quando il risultato della divisione non diventa 0 (e considero comunque il resto!)

# Conversione Decimale ➡ Binario

Metodo delle divisioni successive:

Per convertire 531 opero come segue:

- 531 / 2 = 265 + 1
- 265 / 2 = 132 + 1
- 132 / 2 = 66 + 0
- 66 / 2 = 33 + 0
- 33 / 2 = 16 + 1
- 16 / 2 = 8 + 0
- 8 / 2 = 4 + 0
- 4 / 2 = 2 + 0
- 2 / 2 = 1 + 0
- 1 / 2 = 0 + 1

Cifra meno significativa

I resti della divisione intera, letti dall'ultimo al primo, identificano il numero binario

$$(531)_{10} = (1000010011)_2$$

Cifra più significativa

# Algoritmo per convertire da base $10$ a base $2$

Assumiamo di scrivere il numero binario in un'opportuna struttura dati: un array!

1. Sia $n$ il numero da convertire da base $10$ a base $2$
2. Se $n = 0$ oppure $n = 1$ , allora $n$ è già in base $2$

Altrimenti

3. Salva il resto della divisione tra $n$ e $2$ in una cella di un array
4. Associa ad $n$ la divisione intera tra $n$ e $2$
5. Ripeti 3 – 4 fino a quando $n == 0$
6. Leggi l'array al contrario per ottenere $n$ in base $2$

# Conversioni ottale/esadecimale ➡ decimale

Convertire in base 2 i seguenti numeri:

- $(31)_8 =$
- $(A170)_{16} =$
- $(623)_8 =$
- $(623)_{16} =$

Idea: convertire in base 10 e poi in base 2

# Conversioni ottale/esadecimale  decimale

Convertire in base 2 i seguenti numeri

- $(31)_8 = (25)_{10} = (11001)_2$
- $(A170)_{16} = (41328)_{10} = TBD$
- $(623)_8 = (403)_{10} = (110010011)_2$
- $(623)_{16} = (1571)_{10} = (11000100011)_2$

# Conversioni ottale/esadecimale ➡ binario

È possibile passare in base 10 e quindi utilizzare l'algoritmo delle divisioni successive

È tuttavia più comodo fare delle conversioni direttamente dalla rappresentazione binaria e

Esprimere ogni sequenza di 3 numeri binari in base 8

- $(1231)_{10} = (10011001111)_2 = (\underbrace{010}\ \underbrace{011}\ \underbrace{001}\ \underbrace{111})_2$
- $(1231)_{10} = (2317)_8$ $\qquad (2 \quad 3 \quad 1 \quad 7)_8$

Esprimere ogni sequenza di 4 numeri binari in base 16

- $(1231)_{10} = (10011001111)_2 = (\underbrace{0100}\ \underbrace{1100}\ \underbrace{1111})_2$
- $(1231)_{10} = (4CF)_{16}$ $\qquad (4 \quad C \quad F)_{16}$

Queste tecniche possono essere usate per convertire da base decimale in esadecimale/ottale passando facilmente in binario

# Somma tra Numeri Interi Positivi

Somma in base 2

# Somma tra Numeri Binari

Si eseguono «in colonna» e si opera cifra per cifra

Si considera il riporto come per i decimali

- 0 + 0 = 0 riporto 0
- 1 + 0 = 1 riporto 0
- 0 + 1 = 1 riporto 0
- 1 + 1 = 0 riporto 1

Occorre sommare il riporto della cifra precedente

$$\begin{array}{r} \mathbf{1} \\ \mathbf{0101} + (5)_{10} \\ \mathbf{1001} = (9)_{10} \\ \hline \mathbf{1110} \quad (14)_{10} \end{array}$$

$$\begin{array}{r} \mathbf{111} \\ \mathbf{1111} + (15)_{10} \\ \mathbf{1010} = (10)_{10} \\ \hline (\mathbf{1})\mathbf{1001} \quad (25)_{10} \end{array}$$

Riporto

# Somma tra Numeri Binari

A volte i bit utilizzati per codificare gli addendi non bastano a contenere il risultato

- In questi casi occorrono più bit per codificare il risultato
- Si ha quindi un bit di **carry**

$$\begin{array}{r} \mathbf{1} \\ \mathbf{0101} + (5)_{10} \\ \mathbf{1001} = (9)_{10} \\ \hline \mathbf{1110} \quad (14)_{10} \end{array} \qquad \begin{array}{r} \mathbf{111} \\ \mathbf{1111} + (15)_{10} \\ \mathbf{1010} = (10)_{10} \\ \hline (\mathbf{1})\mathbf{1001} \quad (25)_{10} \end{array}$$

# I numeri Interi

Positivi e Negativi

# Rappresentazione Modulo e Segno

È possibile dedicare **il primo bit** alla codifica del **segno**

- `"1"` il numero che segue è negativo
- `"0"` il numero che segue è positivo

Con $m$ cifre in binario e codifica modulo dedico $2^{m-1}$ per i positivi e $2^{m-1}$ per gli stessi cambiati di segno

- posso rappresentare tutti i numeri nell'intervallo

$$X \in [-2^{m-1}+1, 2^{m-1}-1]$$

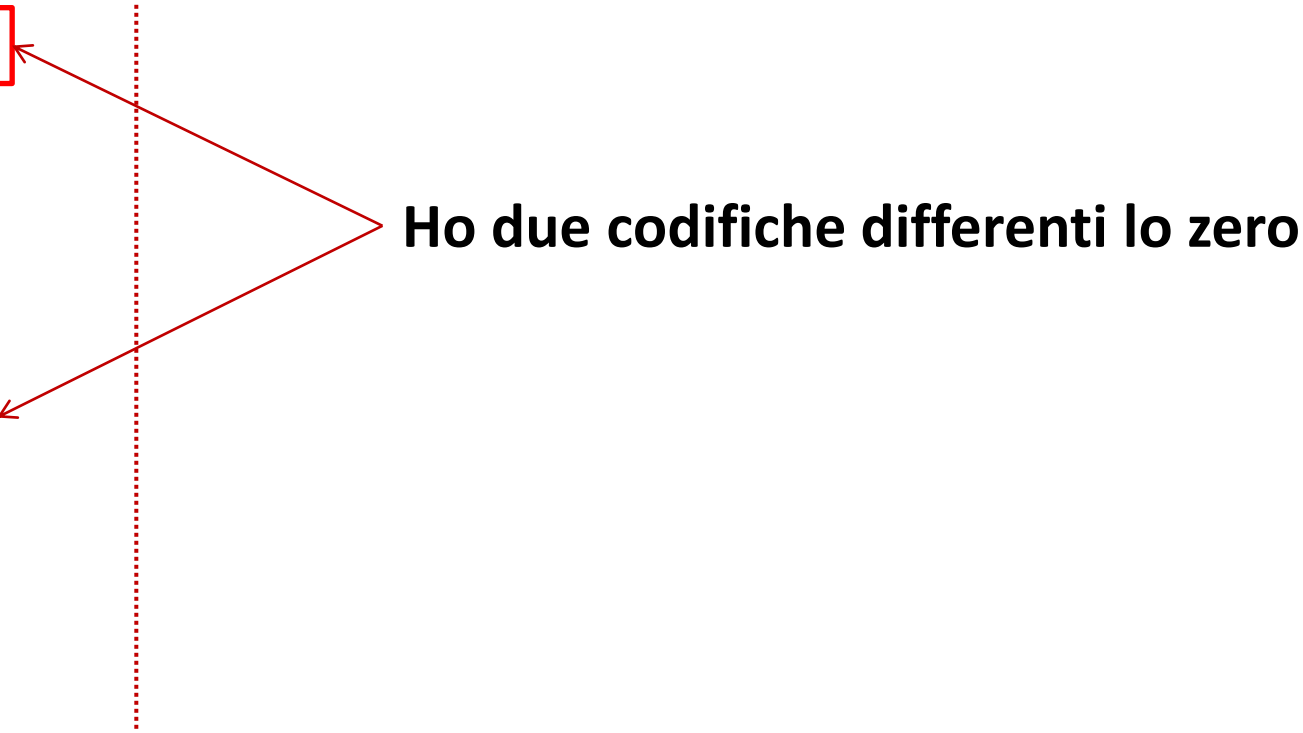
Es

- `01010 = + 10`
- `11101 = - 13`
- `-27 = 111011`
- `122 = 01111010`

# Rappresentazione Modulo e Segno

Esempio $m = 3$

- 0 = 000
- 1 = 001
- 2 = 010
- 3 = 011
- -0 = 100
- -1 = 101
- -2 = 110
- -3 = 111




C'è uno «spreco» nella codifica

Ostacola realizzazione circuitale delle operazioni algebriche (non lo mostriamo)

Occorre trovare una rappresentazione migliore!

# Rappresentazione in Complemento a 2 (CP2)

Date $m$ cifre binarie, disponibili $2^m$ configurazioni distinte

In CP2 se ne usano:

- $2^{m-1} - 1$ per valori positivi
- $1$ per lo zero
- $2^{m-1}$ per i valori negativi

Con $m$ bit rappresento l'intervallo $[-2^{m-1}, 2^{m-1} - 1]$

# Rappresentazione in Complemento a 2 (CP2)

Sia $X \in [-2^{m-1}, 2^{m-1}-1]$ il numero da rappresentare in CP2, con $m$ bit.

- se $X$ è **positivo** o nullo **scrivo $\boldsymbol{X}$** in binario con **$\boldsymbol{m}$ bit**
- se $X$ è **negativo scrivo $\boldsymbol{2^m - |X|}$** in binario con **$\boldsymbol{m}$ bit**

Questo equivale alla seguente codifica:

$$N_{CP2} = a_{m-1}a_{m-2} \dots a_1 a_0$$

$$= -a_{m-1} \times 2^{m-1} + a_{m-2} \times 2^{m-2} + \dots + a_0 \times 2^0$$

$$= -a_{m-1} \times 2^{m-1} + \sum_{i=0}^{m-2} a_i \times 2^i \ , \qquad a_i \in \{0,1\}$$

# Rappresentazione in Complemento a 2 (CP2)

Sia $X \in [-2^{m-1}, 2^{m-1}-1]$ il numero da rappresentare in CP2, con $m$ bit.

- se $X$ è **positivo** o nullo **scrivo $\boldsymbol{X}$** in binario con **$\boldsymbol{m}$ bit**
- se $X$ è **negativo scrivo $\boldsymbol{2^m - |X|}$** in binario con **$\boldsymbol{m}$ bit**

Questo equivale alla seguente codifica:

$$N_{CP2} = a_{m-1}a_{m-2}\ldots a_1 a_0$$

$$= \boxed{-a_{m-1}\times 2^{m-1}} + a_{m-2}\times 2^{m-2} + \ldots + a_0 \times 2^0$$

$$= \boxed{-a_{m-1}\times 2^{m-1}} + \sum_{i=0}^{m-2} a_i \times 2^i \ , \qquad a_i \in \{0,1\}$$

i.e., viene cambiato il segno dell'addendo relativo alla cifra più significativa

# Rappresentazione in CP2

Esempio $m = 3$

- `-4 = 100`
- `-3 = 101`
- `-2 = 110`
- `-1 = 111`
- `0 = 000`
- `1 = 001`
- `2 = 010`
- `3 = 011`

# Rappresentazione in CP2

Esempio $m = 3$

- `-4 = 100`
- `-3 = 101`
- `-2 = 110`
- `-1 = 111`
- `0 = 000`
- `1 = 001`
- `2 = 010`
- `3 = 011`

# Rappresentazione in CP2

Con i positivi copro solo il range $[0, 2^{m-1}-1]$, quindi la prima cifra è 0 (il numero è minore di $2^{m-1}$)

Con i negativi copro il range $[-2^{m-1}, -1]$ e scrivo $2^m - |X|$, e quindi la prima cifra è 1 (il numero è maggiore di $2^{m-1}$)

Quindi, il primo bit **indica il segno** del numero

- Attenzione: questo numero **non è il segno**: **cambiandolo non** si ottiene il **numero opposto**
- $45 = (0101101)_{CP2}$ se cambio di segno alla prima cifra
- $(1101101)_{CP2} \rightarrow -2^6 + 2^5 + 2^3 + 2^2 + 1 =$
  $= -64 + 45 = -19$

Inoltre, un solo valore per lo 0 (cioè $m$ volte 0), nessuna configurazione “sprecata” dalla codifica

# Rappresentazione in CP2

*Es, definire un intervallo che contenga -23 e 45*

- $m = 7$, copro $[-2^6, 2^6 - 1\ ] = [-64, 63]$
- ~~$m = 6$, copro $[-2^5, 2^5 - 1\ ] = [-32, 32]$~~ (non cont. 45)
- $-23 \rightarrow\ 2^7\ - 23 = 128\ - 23 = 105 = (1101001)_{CP2}$
- $45 = (0101101)_{CP2}$

**NB:** occorre utilizzare **sempre $m$ bit**. Se non avessi messo lo 0 iniziale in $(45)_{CP2}$ avrei ottenuto un numero negativo a 6 bit!

# Conversione Decimale CP2

Metodo "operativo" per rappresentare $X$ ad $m$ bit

1. Controllo che $X \in [-2^{m-1}, 2^{m-1} - 1]$, altrimenti $m$ bit non bastano
2. Se $X$ è positivo, scrivo $X$ utilizzando $m$ bit
   **NB:** ricordandosi di aggiungerei zeri se necessario all'inizio del numero!
3. Se $X$ è negativo:
   a) Scrivo $|X|$ utilizzando $m$ bit
   b) **Complemento** tutti i bit di $X$ $(1 \rightarrow 0, 0 \rightarrow 1)$
   c) **Sommo 1** al numero ottenuto

# Esempi Conversione Decimale CP2

Esempio: scrivere -56 in CP2 con il numero di bit necessari

$m = 7$ c

Scrivo (5

Comple

Sommo

# Esempi Conversione Decimale CP2

Esempio: scrivere -56 in CP2 con il numero di bit necessari

*i.* $m = 7$ copre $[-2^6, 2^6 - 1] = [-64\,, 63]$

ii. Scrivo $(56)_{10} \rightarrow 0111000$

iii. Complemento $\rightarrow 1000111$

iv. Sommo 1 $\quad 1$

*v.* $(1001000)_{CP2} = (-56)_{10}$

| | |
|---|---|
| 56 | 0 |
| 28 | 0 |
| 14 | 0 |
| 7 | 1 |
| 3 | 1 |
| 1 | 1 |
| 0 | |

# Esercizi

Esercizio: convertire in complemento a 2 i seguenti numeri, utilizzando il numero di bit necessario per esprimerli tutti

$$(12)_{10} =$$
$$(-12)_{10} =$$
$$(-8)_{10} =$$
$$(1)_{10} =$$
$$(-101)_{10} =$$
$$(-54)_{10} =$$

# Esercizi

Esercizio: convertire in complemento a 2 i seguenti numeri, utilizzando il numero di bit necessario per esprimerli tutti

$$
\begin{aligned}
(12)_{10} &= (0000\ 1100)_{CP2} \\
(-12)_{10} &= (1111\ 0100)_{CP2} \\
(-8)_{10} &= (1111\ 1000)_{CP2} \\
(1)_{10} &= (0000\ 0001)_{CP2} \\
(-101)_{10} &= (1001\ 1011)_{CP2} \\
(-54)_{10} &= (1100\ 1010)_{CP2}
\end{aligned}
$$

## Conversione CP2 Decimale ➡

Possiamo utilizzare la definizione

$$N_{CP2} = a_{m-1}a_{m-2} \dots a_1 a_0$$

$$= -a_{m-1} \times 2^{m-1} + a_{m-2} \times 2^{m-2} + \dots + a_0 \times 2^0$$

$$= -a_{m-1} \times 2^{m-1} + \sum_{i=0}^{m-2} a_i \times 2^i \ , \qquad a_i \in \{0,1\}$$

$$Es\ (1001000)_{CP2} = -2^6 + 2^3 = -64 + 8 = (-56)_{10}$$

$$(10011011)_{CP2} = -2^7 + 2^4 + 2^3 + 2^1 + 2^0 =$$

$$= -128 + 16 + 8 + 2 + 1 = (-101)_{10}$$

**NB** convertite sempre in decimale con questo metodo per controllare le vostre operazioni

## Conversione CP2 Decimale ➡

… in alternativa è possibile utilizzare un metodo operativo:

1. Se $(X)_{CP2}$ inizia per $0$, allora è positivo: lo converto normalmente
2. Se $(X)_{CP2}$ inizia per $1$, allora è negativo
   a) **Complemento** tutti i bit di $(X)_{CP2}$ $(1 \rightarrow 0, 0 \rightarrow 1)$
   b) **Sommo 1** al numero ottenuto
   c) **Converto** in decimale e cambio di segno

# Esempio

Esercizio: riconvertire in decimale i seguenti numeri in complemento a 2

$(1001010)_{CP2} \rightarrow (0110101) \rightarrow (0110110) \rightarrow (-54)_{10}$

$(1001010)_{CP2} = -2^6 + 2^3 + 2^1 = -64 + 8 + 2 = -54$

$(011)_{CP2}$

$(1101001)_{CP2}$

$(11111)_{CP2}$

$(10100)_{CP2}$

$(101)_{CP2}$

# Somma tra Numeri in CP2

In CP2 l'operazione di somma si realizza **come nella rappresentazione binaria posizionale**

Grazie alla rappresentazione in CP2 **è possibile eseguire** anche **sottrazioni** tra numeri binari con lo stesso meccanismo (i.e., somme tra interi di segno opposto)

# Carry e Overflow in CP2

In CP2 occorre **ignorare il bit di carry**, cioè il riporto che cade sul bit che cade sul segno

In CP2 occorre individuare **l'overflow**, i.e., casi in cui il risultato è fuori dall'intervallo rappresentabile con i bit utilizzati.

Quando c'è **overflow il risultato è inconsistente** con gli addendi:

- Somma di due addendi positivi da un numero negativo
- Somma di due addendi negativi da un numero positivo

**NB** non può esserci overflow quando sommo due numeri di segno opposto

# Esempio no overflow

*Esempio:* 60 − 54

diventa 60 + (−54)

$$\begin{array}{rcl} & & \phantom{(}1\ 1\ \ 1\ 1 \\ (60)_{10} & = & (0\ 1\ 1\ 1\ 1\ 0\ 0)_{CP2} \\ (-54)_{10} & = & (1\ 0\ 0\ 1\ 0\ 1\ 0)_{CP2} \\ \hline & & (1)\ 0\ 0\ 0\ 0\ 1\ 1\ 0 \end{array}$$

Il riporto (carry) viene ignorato

Quando sommo numeri di segno opposto non può esserci overflow

Il risultato è positivo $(0000110)_{CP2} = (6)_{10}$

# Esempio

Esempio: $(100)_{CP2} + (101)_{CP2}$

1

1 0 0 +

1 0 1 =

[1] (1) 0 0 1

Ignoro il bit di carry

**Overflow:** la somma di due numeri negativi mi ha dato un numero positivo.

L'overflow si indica quadre: [1] c'è overflow, [0] non c'è

Il risultato non ha senso, occorre scrivere gli addendi con un bit in più per rappresentare il risultato dell'operazione

# Esempi

Esempi: con $m = 4$ bit

Indico tra () bit di carry, tra [] bit di overflow

-3 => 1101
-4 => 1100
-7 => [0](1)1001

-3 => 1101
+6 => 0110
+3 => [0](1)001

-3 =>
-7 =>
-10 =>

+2 =>
+5 =>
+7 =>

+3 =>
+6 =>
+9 =>

# Esempi

Esempi: con $m = 4$ bit

Indico tra () bit di carry, tra [] bit di overflow

-3 => 1101
-4 => 1100
-7 => [0] (1)1001

-3 => 1101
+6 => 0110
+3 => [0] (1)001

-3 =>
-7 =>
-10 =>

+2 =>
+5 =>
+7 =>

+3 =>
+6 =>
+9 =>

# Esempi

Esempi: con $m = 4$ bit

Indico tra () bit di carry, tra [] bit di overflow

-3 => 1101
-4 => 1100
-7 => [0] (1)1001

-3 => 1101
+6 => 0110
+3 => [0] (1)0011

-3 =>
-7 =>
-10 =>

+2 =>
+5 =>
+7 =>

+3 =>
+6 =>
+9 =>

# Esempi

Esempi: con $m = 4$ bit

Indico tra () bit di carry, tra [] bit di overflow

-3 => 1101
-4 => 1100
-7 => [0] (1)1001

-3 => 1101
+6 => 0110
+3 => [0] (1)0011

-3 => 1101
-7 => 1001
-10 => [1](1) 0110

+2 =>
+5 =>
+7 =>

+3 =>
+6 =>
+9 =>

# Esempi

Esempi: con $m = 4$ bit

Indico tra () bit di carry, tra [] bit di overflow

-3 => 1101
-4 => 1100
-7 => [0] (1)1001

-3 => 1101
+6 => 0110
+3 => [0] (1)0011

-3 => 1101
-7 => 1001
-10 => [1](1) 0110

+2 => 0010
+5 => 0101
+7 => [0](0) 0111

+3 =>
+6 =>
+9 =>

# Esempi

Esempi: con $m = 4$ bit

Indico tra () bit di carry, tra [] bit di overflow

-3 => 1101
-4 => 1100
-7 => [0] (1)1001

-3 => 1101
+6 => 0110
+3 => [0] (1)0011

-3 => 1101
-7 => 1001
-10 => [1](1) 0110

+2 => 0010
+5 => 0101
+7 => [0](0) 0111

+3 => 0011
+6 => 0110
+9 => [1](0) 1001

# Esempio TDE 11/2009

a) Si dica qual è l'intervallo di valori interi rappresentabile con la codifica in complemento a due a 9 bit.

b) Con riferimento a tale codifica indicare, giustificando brevemente le risposte, quali delle seguenti operazioni possono essere effettuate correttamente:
   i. -254 – 255
   ii. + 254 – 253
   iii. -18 + 236
   iv. + 217 + 182

c) Mostrare in dettaglio come avviene il calcolo delle operazioni (i) e (ii), evidenziando il bit di riporto e il bit di overflow così ottenuti. (Il bit di overflow è pari ad 1 se si verifica overflow, 0 altrimenti.)

# Esempio TDE 11/2009

a. Valori rappresentabili vanno da -256 a +255.

b. Le soluzioni:

   i. -254 – 255: NO, si ottiene un valore negativo troppo grande in valore assoluto

   ii. + 254 – 253: SI, si ottiene un valore piccolo in valore assoluto

   iii. -18 + 236: SI, si ottiene un valore positivo, grande in valore assoluto ma nei limiti

   iv. + 217 + 182: NO, si ottiene un valore positivo troppo grande in valore assoluto

c.

| | |
|---|---|
| 100000010 (-254) | ii. 011111110 (+254) |
| 100000001 (-255) | 100000011 (-253) |
| [1](1)000000011 (-509) | [0](1)000000001 (+1) |

Convertire il messaggio pubblicitario in base 10 assumendo sia scritto in CP2