



# Introduzione al Corso

Informatica a, AA 2023/2024

Giacomo Boracchi

13 Settembre 2023

[giacomo.boracchi@polimi.it](mailto:giacomo.boracchi@polimi.it)

Ogni prof è responsabile degli studenti del proprio scaglione, anche se le lezioni saranno spesso svolte congiuntamente

Prof. Campi ha lo scaglione A – LZZ

Prof. Boracchi ha lo scaglione M – ZZZ

# Chi siamo

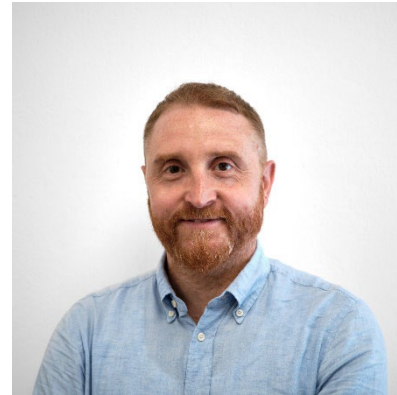
## Chi siamo

Giacomo Boracchi ([giacomo.boracchi@polimi.it](mailto:giacomo.boracchi@polimi.it))

Matematico (Università Statale degli Studi di Milano 2004),

PhD in Information Technology (DEIB, Politecnico di Milano 2008)

Professore Associato dal 2019 al DEIB, Polimi (Computer Science)



Nella mia ricerca mi occupo di modelli e metodi matematici e statistici per:

Analisi/elaborazione di immagini

Machine Learning (unsupervised learning), change and anomaly detection

### Giacomo Boracchi (docente)

- Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano
- homepage: <https://boracchi.faculty.polimi.it>
- email [giacomo.boracchi@polimi.it](mailto:giacomo.boracchi@polimi.it)
- Webex Room: <https://politecnicomilano.webex.com/meet/giacomo.boracchi>
- ricevimento studenti:
  - **Su appuntamento**, da prendere via email.
- Ufficio 157 via Ponzio 34/5, Milano.
- tel 02 2399 3467

# Chi siamo

Ing. Andrea Porfiri Dal Cin (esercitatore)

- Homepage: **TBD**
- email: [andrea.porfiridalcin@polimi.it](mailto:andrea.porfiridalcin@polimi.it)

Pagina esercitazioni: <https://andreadalcin.github.io/informaticaA/>



## Chi siamo

Ing. Edoardo Peretti (responsabile di laboratorio, Squadra 1)

- email: [edoardo.peretti@polimi.it](mailto:edoardo.peretti@polimi.it)

Ing. Gian Enrico Conti (responsabile di laboratorio, Squadra 2)

- email: [gianenrico.conti@polimi.it](mailto:gianenrico.conti@polimi.it)

Pagina Laboratorio: **TBD**



# Il Corso

<https://boracchi.faculty.polimi.it/teaching/InfoA.htm>



La pagina del corso è

<https://boracchi.faculty.polimi.it/teaching/InfoA.htm>

Troverete:

- Materiale didattico usato a lezione (queste slides sono da considerare **un supporto** allo studio)
- Link ai siti delle esercitazioni e laboratori
- Temi d'esame con soluzioni
- Calendario del corso (lezioni, esercitazioni, laboratori)
- Avvisi, esiti esami

**N.B.** Le slides caricate prima della lezione non contengono le soluzioni agli esercizi che affronteremo in aula. **Le slides vengono completate, annotate ed aggiornate dopo la lezione.**

## Organizzazione, lezioni ed esercitazioni

Le lezioni si terranno

- Raramente il Lunedì dalle **13:30 alle 16:15, Aula 5.0.3**
- Circa un Mercoledì su due dalle **13:30 alle 15:15 Aula 5.1.1**
- Spesso il Venerdì dalle **8:45 alle 11:15 Aula 5.1.1**

Le esercitazioni si terranno

- Spesso il Lunedì dalle **13:30 alle 16:15, Aula 5.0.3**
- Circa un Mercoledì su due dalle **13:30 alle 15:15 Aula 5.1.1**
- Raramente il Venerdì dalle **8:45 alle 11:15 Aula 5.1.1**

**I Laboratori si terranno in presenza e partiranno dal 26 Settembre**

- **Martedì in Aula TBD (21.0.1 oppure 21.0.2)**
- **Mercoledì in Aula TBD (21.0.1 oppure 21.0.2)**

**I laboratori saranno solo 15 ore (6 incontri da due ore + 3 da un'ora): non si terranno quindi ogni settimana.**

Riceverete a breve indicazioni sul laboratorio

**E' bene non cambiare squadra** per garantire un bilanciamento delle squadre. Tuttavia, scambi possono essere concordati con i responsabili di laboratorio.

**Controllate sempre il calendario sul sito:**

<https://boracchi.faculty.polimi.it/teaching/InfoACalendar.htm>

I link alle registrazioni verranno riportati nel Google calendar

Nei prossimi giorni potrebbe esserci qualche cambiamento al  
calendario

## GIACOMO BORACCHI - TEACHING

### Informatica A Ingegneria Matematica, AA 2023/2024

#### Calendario

Oggi mercoledì, 13 settembre Stampa Settimana Mese Agenda

mercoledì, 13 settembre	
13:30	Lezione
Quando	mer, 13 settembre, 13:30 – 15:00
Dove	Aula 5.1.1 ( <a href="#">mappa</a> ) <a href="#">altri dettagli»</a> <a href="#">copia nel mio calendario</a>
venerdì, 15 settembre	
08:45	Lezione
Quando	ven, 15 settembre, 08:45 – 11:00
Dove	Aula 5.1.1 ( <a href="#">mappa</a> ) <a href="#">altri dettagli»</a> <a href="#">copia nel mio calendario</a>
lunedì, 18 settembre	
13:30	Lezione
Quando	lun, 18 settembre, 13:30 – 16:00
Dove	Aula 5.0.3 ( <a href="#">mappa</a> ) <a href="#">altri dettagli»</a> <a href="#">copia nel mio calendario</a>
mercoledì, 20 settembre	
13:30	Lezione
venerdì, 22 settembre	
08:45	Lezione
lunedì, 25 settembre	
13:30	Esercitazione
venerdì, 29 settembre	
08:45	Esercitazione
lunedì, 2 ottobre	
13:30	Lezione



# Lezioni ed Esercitazioni

How to..

### Due consigli:

1. Sfruttate al massimo la presenza di docenti ed esercitatori.
2. Prendete appunti e riguardate regolarmente almeno gli esercizi svolti in aula.

### Quindi:

- Molto bene l'interazione docente-studente:
- Farò domande, rispondete!
- Chiedete le cose che non vi sono chiare!
- Durante le lezioni ed esercitazioni useremo **prevalentemente la lavagna (non tutte le slides verranno mostrate)**

### Due consigli:

1. Sfruttate al massimo la presenza di docenti ed esercitatori.
2. Prendete appunti e riguardate regolarmente almeno gli esercizi svolti in aula.

**Non vale arrendersi: non si esce  
prima della fine della  
lezione/esercitazione  
(o della pausa)!**

### Quindi:

- Molto bene l'ir
- Farò domande, rispondete!
- Chiedete le cose che non vi sono chiare!
- Durante le lezioni ed esercitazioni useremo prevalentemente la lavagna (non tutte le slides verranno mostrate)



## Esercizi a lezione:

- Farò **molti esercizi** durante la lezione
- **Spesso svolti** al PC o alla lavagna
- Non sempre ci saranno le soluzioni nei materiali online
- 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)

## Esercizi ad esercitazione:

- Prevalentemente soluzioni alla lavagna
- Il laptop non serve
- I testi degli esercizi saranno caricati sul sito il giorno prima
- Le soluzioni verranno caricate nei giorni seguenti: potete esercitarvi e controllare le vostre soluzioni
- Tipicamente trovate sui nostri siti molti più esercizi svolti di quanti riusciremo a farne a lezione

**Mi auspico la massima interazione durante lezioni ed esercitazioni**

- Ricorda: *La tua domanda può risolvere il dubbio di molti tuoi colleghi*

# Laboratori

Nei laboratori vi sarà richiesto di **programmare autonomamente**

Sarete divisi in due squadre

- **'Squadra 1' CP DISPARI**
- **'Squadra 2' CP PARI**

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.

Nei laboratori vi sarà richiesto di **programmare autonomamente**

Sarete divisi in due squadre

- 'Squadra 1' CP DISPARI.
- 'Squadra

Il Laboratorio è

- prendere
- **consolidare** la lezione.

**Non si impara a programmare su carta!**

**Durante il lab fate domande e chiedete chiarimenti, non potete risolvere tutti gli esercizi.**

nti introdotti a

**Arrivare preparati e con una certa familiarità con l'ambiente di sviluppo prima del laboratorio**

I testi degli esercizi di laboratorio verranno pubblicati e presentati nella lezione del Lunedì:

- incominciate a lavorarci, userete il lab per fare domande e chiedere chiarimenti

I responsabili sono disponibili per fornire assistenza, chiamateli alla vostra postazione e chiedete consigli/aiuto!

Prima di chiamare un responsabile/tutor:

1. Sistemare il codice
2. Leggere tutti i messaggi del compilatore
3. Formulare una domanda chiara

Vi invitiamo ad installare l'ambiente di programmazione

- Installare Code::Blocks per il C, versione con compilatore (<http://www.codeblocks.org/>)
- Sul sito del corso troverete le istruzioni per installare su Windows e Mac OS
- [https://boracchi.faculty.polimi.it/teaching/InfoA/install\\_code\\_blocks\\_Win10-OSX-AA2019-20.pdf](https://boracchi.faculty.polimi.it/teaching/InfoA/install_code_blocks_Win10-OSX-AA2019-20.pdf)

Potete anche usare alternative come Xcode (utenti iOS) o DevC++

# 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.

Potete anche rivolgere domande via mail.

- Però, per facilitare il lavoro di tutti, è bene evitare richieste vaghe e del tipo: « è corretto così? »

Quando scrivete una mail per una richiesta di chiarimento

- Dite di che corso siete (è buona norma!)
- Allegate il codice sorgente, pulito e commentato
- Allegate il testo dell'esercizio (no foto, al massimo screenshot dell'errore non del codice)
- Un breve commento che spiega cosa non funziona e i tentativi che avete fatto.

# L'Esame

## Modalità di Verifica

L'esame sarà svolto **programmando sul vostro computer.**

- **Fase 1:** Esercizi di programmazione da svolgere sul vostro laptop in aula (0-22 punti)
- **Fase 2:** Orale per verificare la capacità di programmare e la conoscenza degli argomenti del corso (0-10 punti, ammissione condizionata a fase 1)

**Ci aspettiamo sappiate scrivere programmi funzionanti.**

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

Stiamo valutando di fare una prova intermedia durante l'interruzione di Novembre

# Modalità di esame dal AA20/21



## 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

Esame scritto AA19/20

## Informatica A

Cognome \_\_\_\_\_

Nome \_\_\_\_\_

Matricola o Codice studente \_\_\_\_\_

### Istruzioni

Non separate questi fogli. Scrivete la soluzione **solo sui fogli distribuiti**, utilizzando il retro delle pagine in caso di necessità. **Cancellate le parti di brutta** (o ripudiate) con un tratto di **penna**.

Ogni parte non cancellata a penna sarà considerata parte integrante della soluzione.

**È possibile scrivere a matita** (e non ricalcare al momento della consegna).

È **vietato** utilizzare **calcolatrici** o **telefoni**. Chi tenti di farlo vedrà **annullata** la sua prova.

È ammessa la consultazione di **libri** e **appunti**, purché con pacata discrezione e senza disturbare.

Qualsiasi **tentativo** di comunicare con altri studenti comporta **l'espulsione** dall'aula.

È possibile **ritirarsi senza penalità**.

Non è possibile lasciare l'aula conservando il tema della prova in corso.

Tempo a disposizione: 2 ore e 40 minuti

Esame scritto AA19/20

## Informatica A

Cognome \_\_\_\_\_

Nome \_\_\_\_\_

Dall'AA 2021-2022 i temi d'esame saranno svolti unicamente al computer e vi sarà chiesto di scrivere programmi funzionanti.

Ogni parte non cancellata a penna sarà considerata parte integrante della soluzione.

È **possibile scrivere a matita** (e non ricalcare al momento della consegna).

È **vietato** utilizzare **calcolatrici** o **telefoni**. Chi tenti di farlo vedrà **annullata** la sua prova.

È ammessa la consultazione di **libri** e **appunti**, purché con pacata discrezione e senza disturbare.

Qualsiasi **tentativo** di comunicare con altri studenti comporta **l'espulsione** dall'aula.

È possibile **ritirarsi senza penalità**.

Non è possibile lasciare l'aula conservando il tema della prova in corso.

Tempo a disposizione: 2 ore e 40 minuti



Esame scritto AA19/20

## Informatica A

Cognome \_\_\_\_\_

Nome \_\_\_\_\_

Matricola o Codice studente \_\_\_\_\_

### Istruzioni

Non separate questi fogli. Scrivete la soluzione **solo sui fogli distribuiti**, utilizzando il retro delle pagine in caso di necessità. **Cancellate le parti di brutta** (o ripudiate) con un tratto di **penna**.

Ogni parte non cancellata a penna sarà considerata parte integrante della soluzione.

**È possibile scrivere a matita** (e non ricalcare al momento della consegna).

È **vietato** utilizzare **calcolatrici** o **telefoni**. Chi tenti di farlo vedrà **annullata** la sua prova.

È ammessa la consultazione di **libri e appunti**, purché con pacata discrezione e senza disturbare.

Qualsiasi **tentativo** di comunicare con altri studenti comporta **l'espulsione** dall'aula.

È possibile **ritirarsi senza penalità**.

Non è possibile lasciare l'aula conservando il tema della prova in corso.

Tempo a disposizione: 2 ore e 40 minuti

Esame scritto AA19/20

## Informatica A

Cognome \_\_\_\_\_

Nome \_\_\_\_\_

Dall'AA 2019 – 2020 non sarà più possibile consultare nulla durante gli esami.  
Vi forniremo noi un eventuale cheat sheet per la sintassi delle varie funzioni C

È **vietato** utilizzare **calcolatrici** o **telefoni**. Chi tenti di farlo vedrà **annullata** la sua prova.

~~È ammessa la consultazione di libri e appunti, purché con discrezione e senza disturbare.~~

Qualsiasi **tentativo** di comunicare con altri studenti comporta **l'espulsione** dall'aula.

È possibile **ritirarsi senza penalità**.

Non è possibile lasciare l'aula conservando il tema della prova in corso.

Tempo a disposizione: 2 ore e 40 minuti

# Argomenti Trattati

Introduzione all'informatica

Codifica binaria, Algebra di Boole, Introduzione all'architettura di Von Neumann

## **Programmazione in linguaggio C**

- Le variabili e le strutture di controllo
- Array
- Funzioni
- Strutture dati
- Puntatori
- Ricorsione
- Memoria dinamica
- Strutture dati dinamiche (... liste, alberi...)

Fondamenti di Database e SQL

## Cosa Imparerete:

- Gli **elementi fondamentali** ed i **principi** che regolano il funzionamento di un **sistema informatico**
- Come **sviluppare algoritmi** per risolvere problemi
- Come **codificare** tali **algoritmi** in programmi che ne permettano l'automatizzazione.
- Le basi della **programmazione** in linguaggio C
- Come sono organizzati i dati in un database relazionale e come interrogare un DB mediante query SQL

## Perché studiare informatica?

- Perché l'informatica è, a livello mondiale, uno dei maggiori settori industriali, e ha importanza strategica
- Perché, oltre ad essere una tecnologia *primaria*, è una tecnologia *abilitante* per altre tecnologie e per altri settori industriali
- Per **capire** la società dell'informazione

Testo di riferimento: D.Mandrioli, S.Ceri, L.Sbattella, P. Cremonesi, G. Cugola:  
*“Informatica: arte e mestiere”*, McGraw-Hill

Altri testi, eserciziari e manuali vari di C:

- A. Bellini, A. Guidi: *“Linguaggio C - Guida alla programmazione”* – ed. McGraw-Hill [graduale e completo, adatto a chi parte da zero]
- D. Braga, D. Martinenghi: *“Fondamenti di Informatica, Temi d’esame risolti”* – ed. Esculapio

# Cos'è l'informatica?

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



## Cos'è l'Informatica?

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

**Scienza:** ovvero una **conoscenza sistematica e rigorosa** di tecniche e metodi.

**Informazione:** l'oggetto dell'investigazione scientifica (informazione intesa sia come entità astratta che 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.

[da «Informatica Arte e Mestiere»]

## Cos'è l'Informatica?

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

**Scienza:** ovvero una **conoscenza sistematica e rigorosa** di tecniche e metodi.

**Informazione:** l'oggetto dell'investigazione scientifica (informazione intesa sia come entità astratta che 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.

[da «Informatica Arte e Mestiere»]

# 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)]

# Gli Algoritmi

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

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

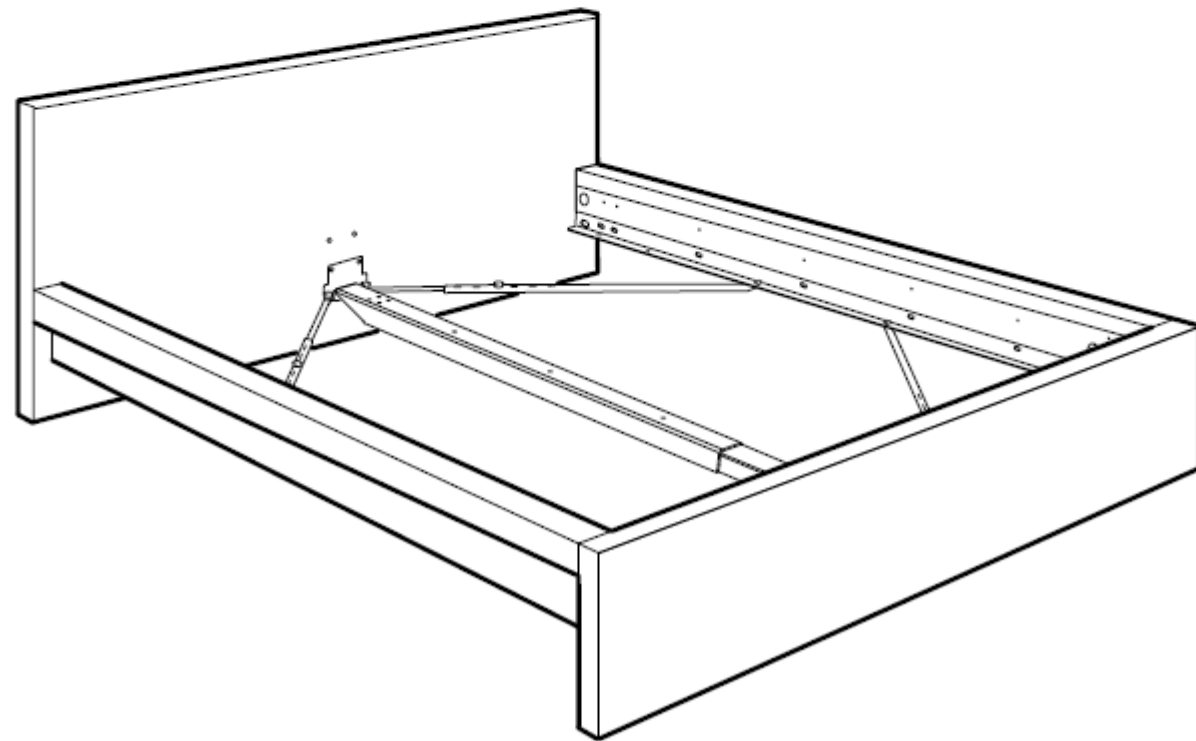
Le operazioni devono:

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

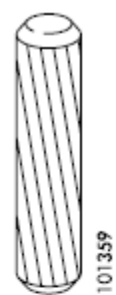
Non è necessario un computer per fare algoritmi!

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

# MALM



Design and Quality  
IKEA of Sweden



101359

12x



110789

20x/22x



105163

8x



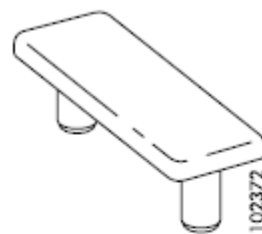
117327

12x



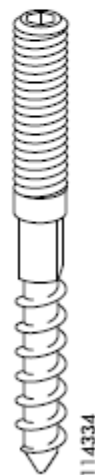
102267

8x



102372

6x



114334

4x



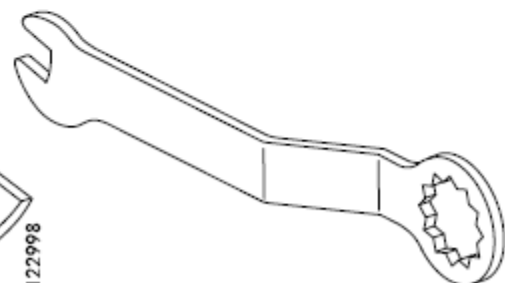
114254

4x



122998

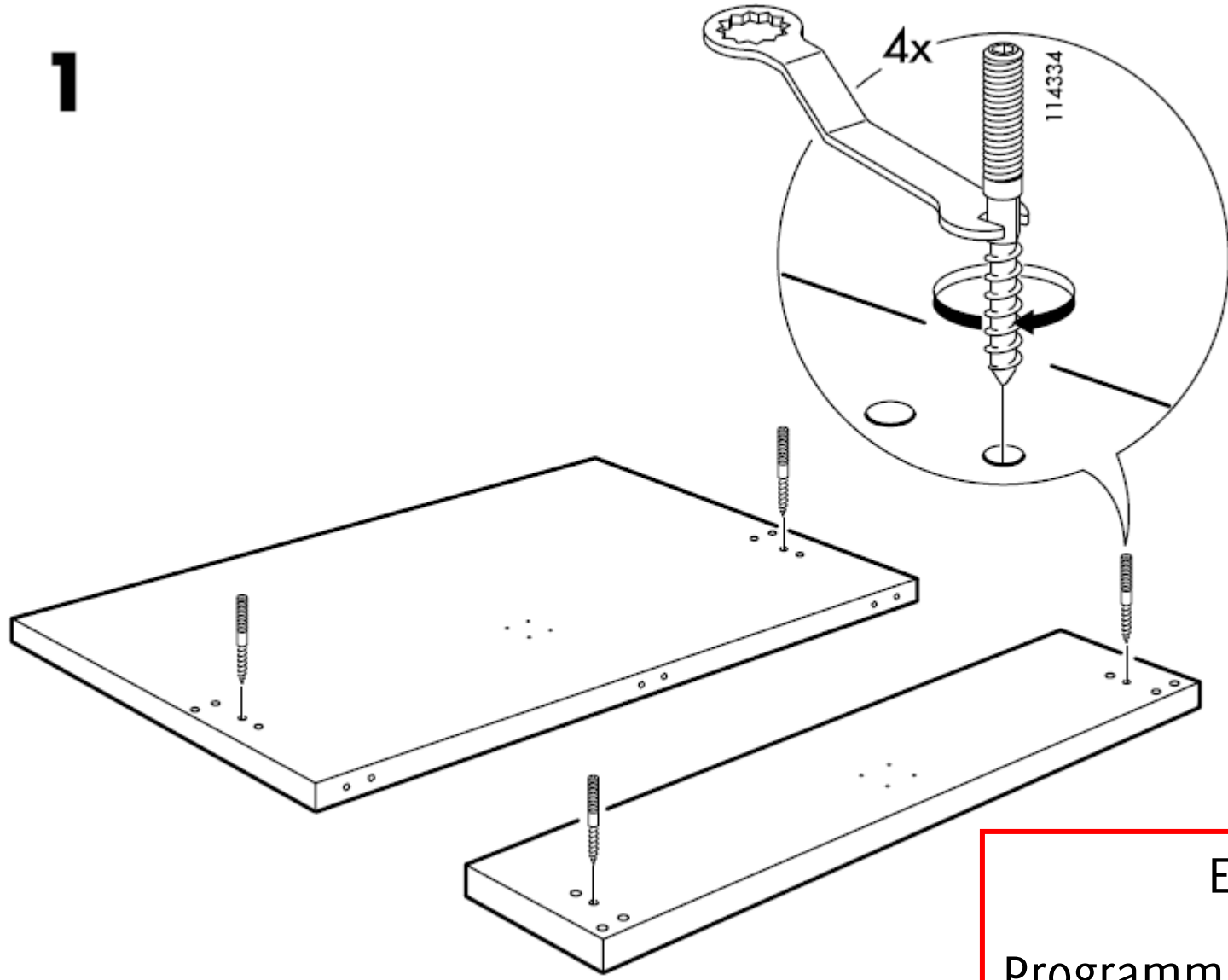
4x



113453

1x

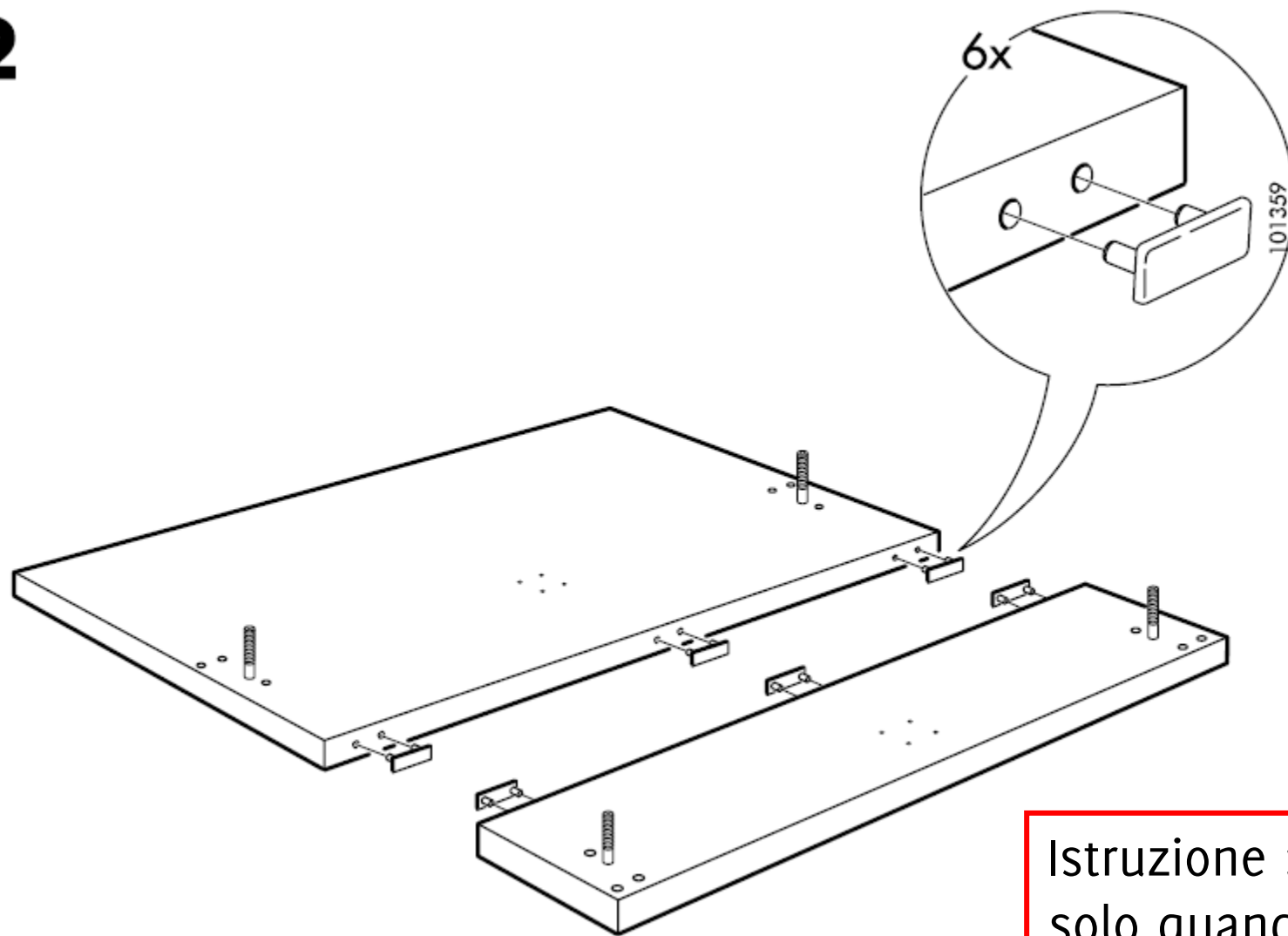
1



Esecutore: Voi  
Programmatore: Designer di IKEA  
Linguaggio: illustrazioni



**2**



Istruzione 2 viene iniziata  
solo quando Istruzione 1  
termina

## Il Linguaggio «IKEA»

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

## 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 («se» / «if»)
- Costrutto iterativo («finché» / «while»)

Inoltre, potremo utilizzare “foglietti” dove scrivere o registrare valori

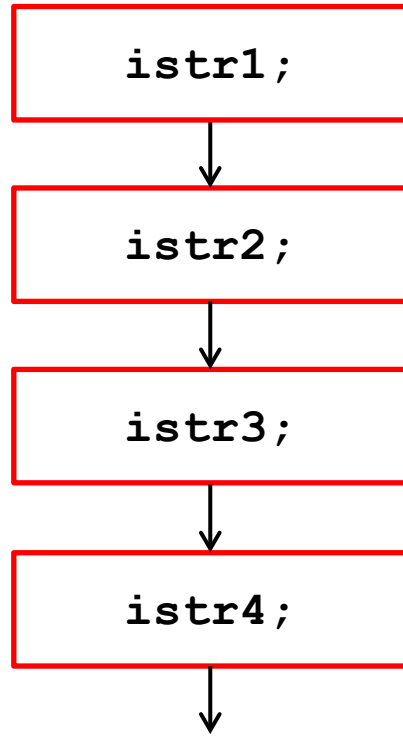
# La Sequenzialità

## Esempio: algoritmo per andare in università

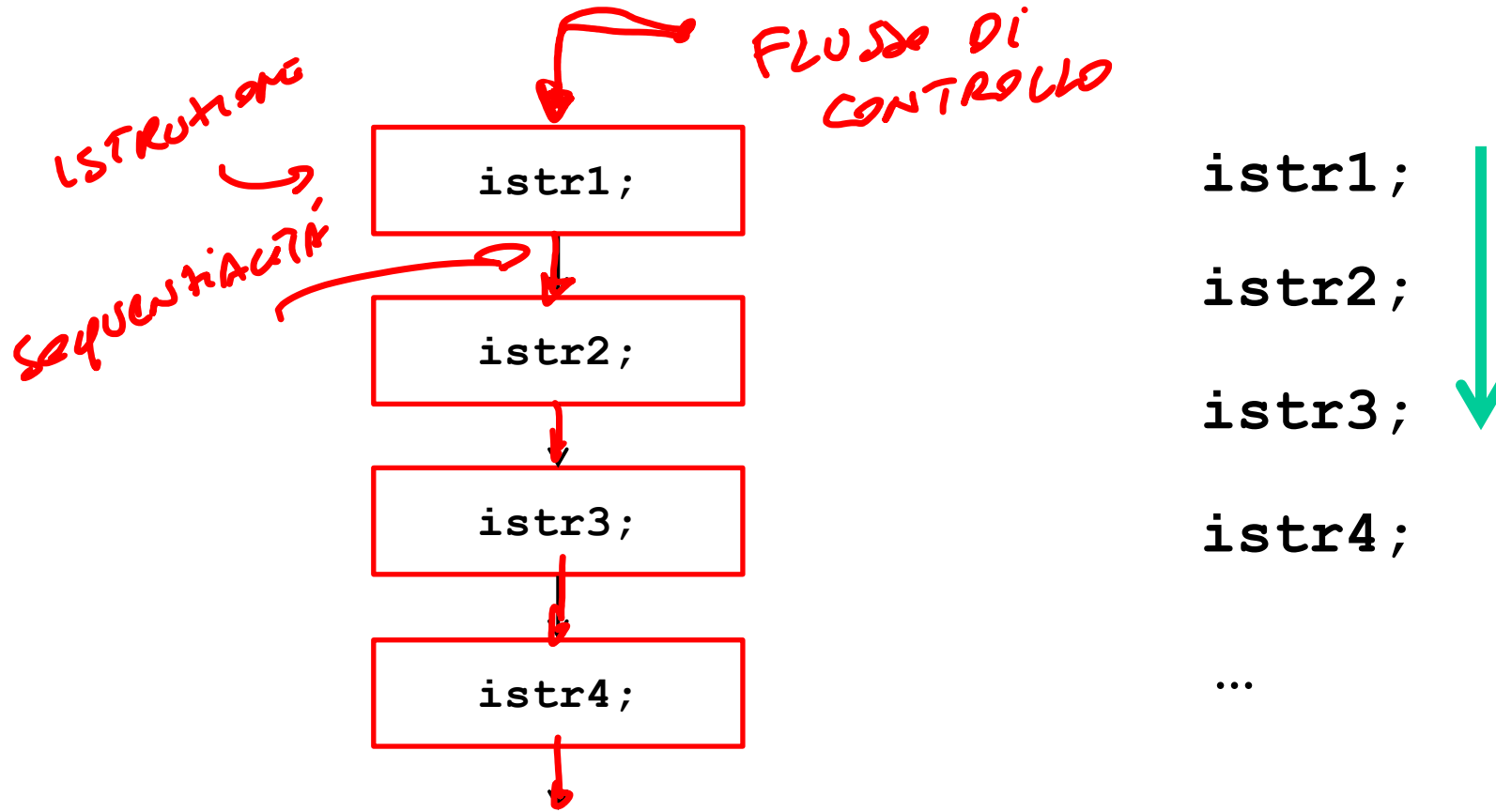
1. Mi alzo quando suona la sveglia
2. Mi dirigo verso la cucina
3. Mangio e bevo un caffè
4. Mi lavo
5. Mi vesto
6. Esco
7. Corro

Istruzione  $i + 1$  viene iniziata  
solo quando Istruzione  $i$   
termina

# La sequenzialità



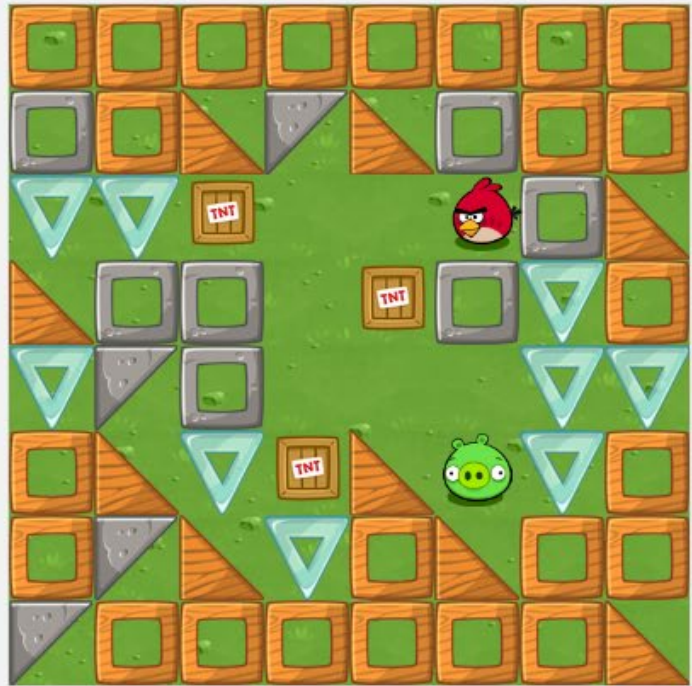
# La sequenzialità




Le istruzioni vengono eseguite **dalla prima all'ultima**.

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

# La sequenzialità per i bimbi su code.org



▶ Run

 Trace the path and lead me to the silly pig. Avoid TNT or the feathers will fly! Hint: He's South of me.

Blocks Assemble your blocks here: 10 / 8 Show Code

N ↑

S ↓

E →

W ←

when run ▶

W ←

W ←

S ↓

S ↓


S ↓

E →

E →

E →

S ↓





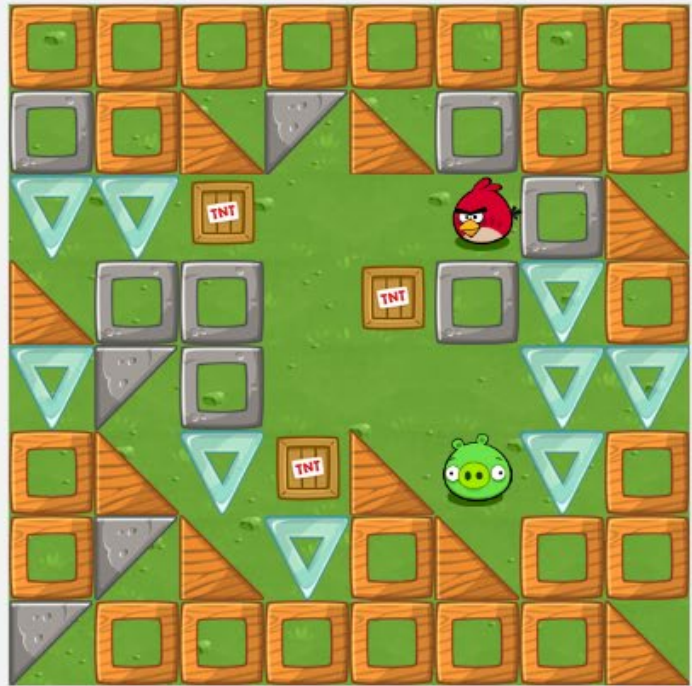
# La sequenzialità per i bimbi su code.org

The screenshot shows the code.org interface for a puzzle stage. The browser address bar displays `studio.code.org/s/course1/stage/4/puzzle/11`. The page title is "Stage: Maze: Sequence, Puzzle". The user is logged in as "Hi Giovannino". The puzzle area shows a maze with TNT blocks and a pig. A modal dialog box is open, featuring a red Angry Bird icon and the text: "Even top universities teach block-based coding (e.g., **Berkeley**, **Harvard**). But under the hood, the blocks you have assembled can also be shown in JavaScript, the world's most widely used coding language:". The dialog contains a code editor with the following JavaScript code: 


```
moveWest();
moveWest();
moveSouth();
moveSouth();
moveSouth();
moveEast();
moveEast();
moveEast();
moveSouth();
```

 An "OK" button is located at the bottom right of the dialog. In the background, a "Reset" button and a hint are visible: "Trace the path and lead me to the silly pig. Avoid TNT or the feathers will fly! Hint: He's South of me." A vertical stack of directional buttons (E, E, S) is also visible.

# La sequenzialità per i bimbi su code.org



▶ Run

 Trace the path and lead me to the silly pig. Avoid TNT or the feathers will fly! Hint: He's South of me.

Blocks Assemble your blocks here: 10 / 8 Show Code

N ↑

S ↓

E →

W ←

when run ▶

W ←

W ←

S ↓

S ↓


S ↓

E →

E →

E →

S ↓

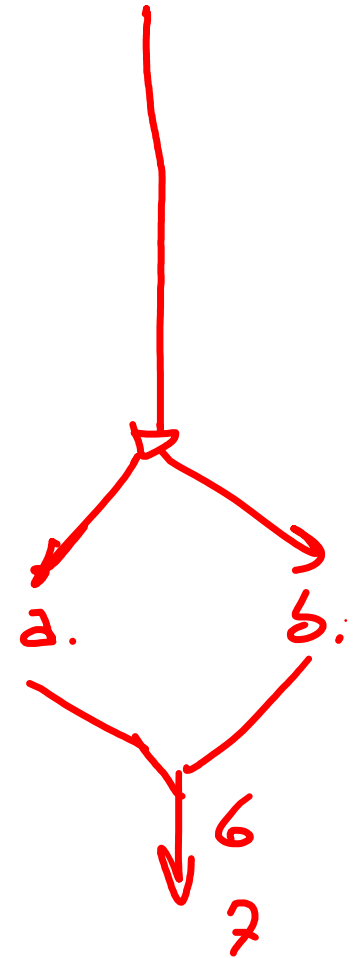


Btw, trovate l'errore...

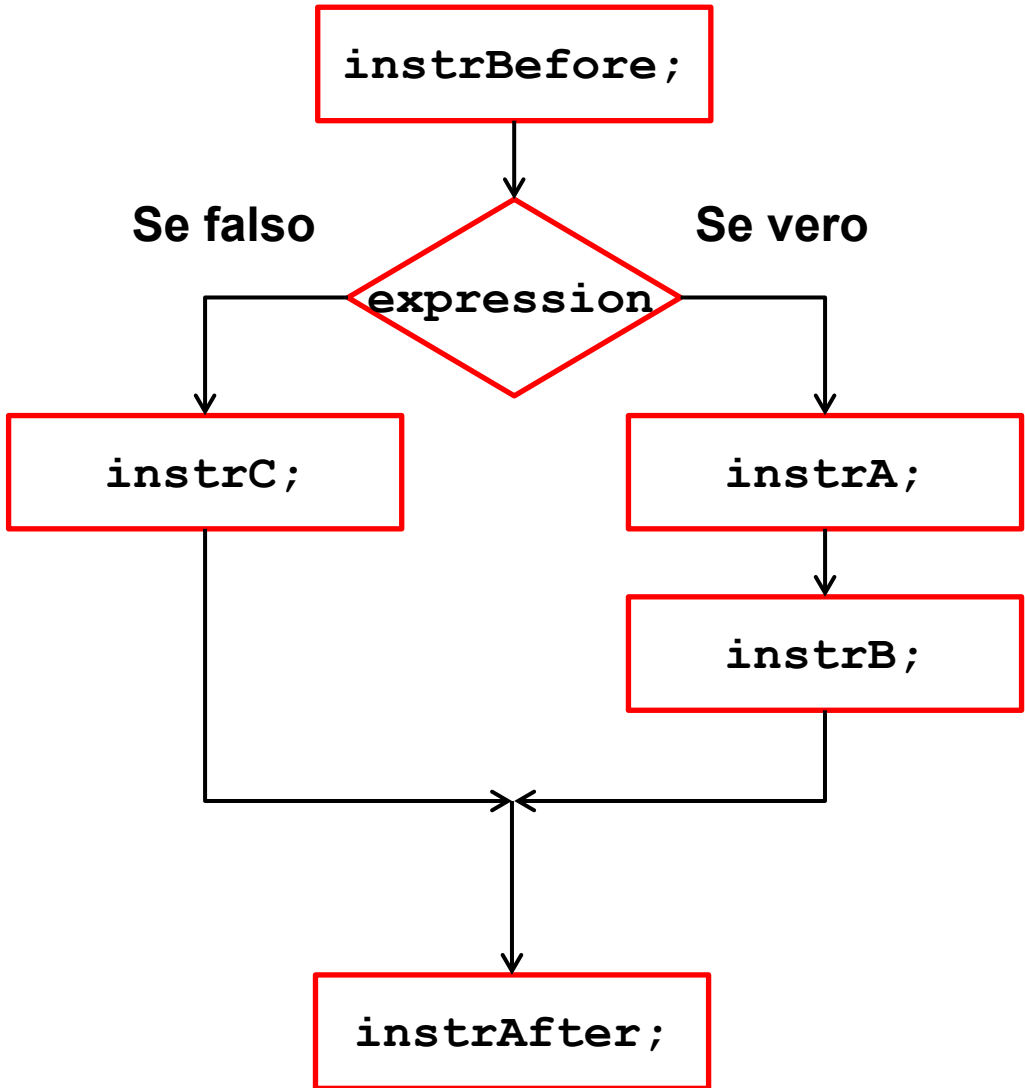
# Costrutto Condizionale

## Esempio: algoritmo per andare in Università

1. Mi alzo quando suona la sveglia
2. Mi dirigo verso la cucina
3. Mangio e bevo un caffè
4. Mi lavo e mi vesto
5. **Se** devo mangiare fuori
  - a. prendo il pranzo**Altrimenti**
  - b. prendo uno snack per metà mattina
6. Esco
7. Corro

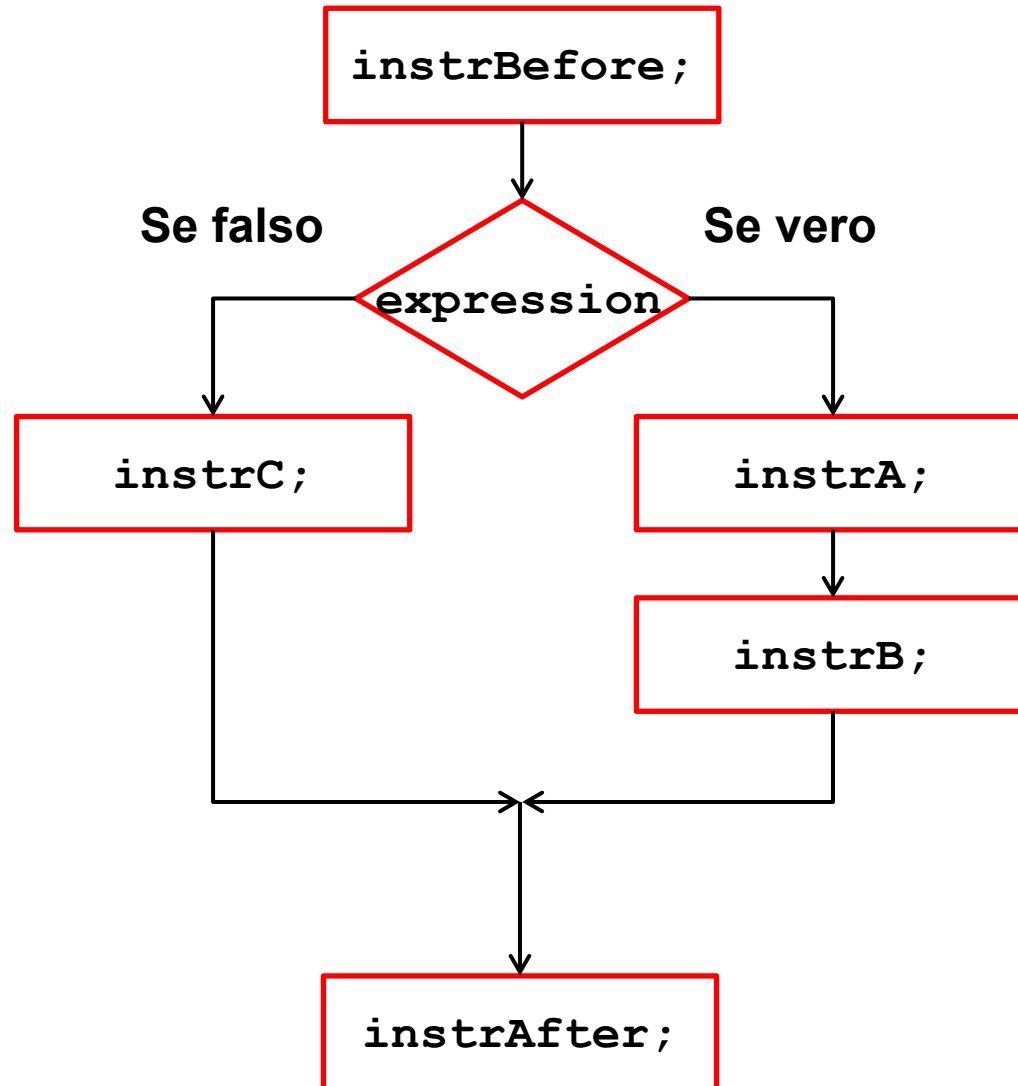


# Costrutto Condizionale:



**Expression**  
corrisponde ad una condizione che deve essere valutata per capire se è vera o falsa

## Costrutto Condizionale:

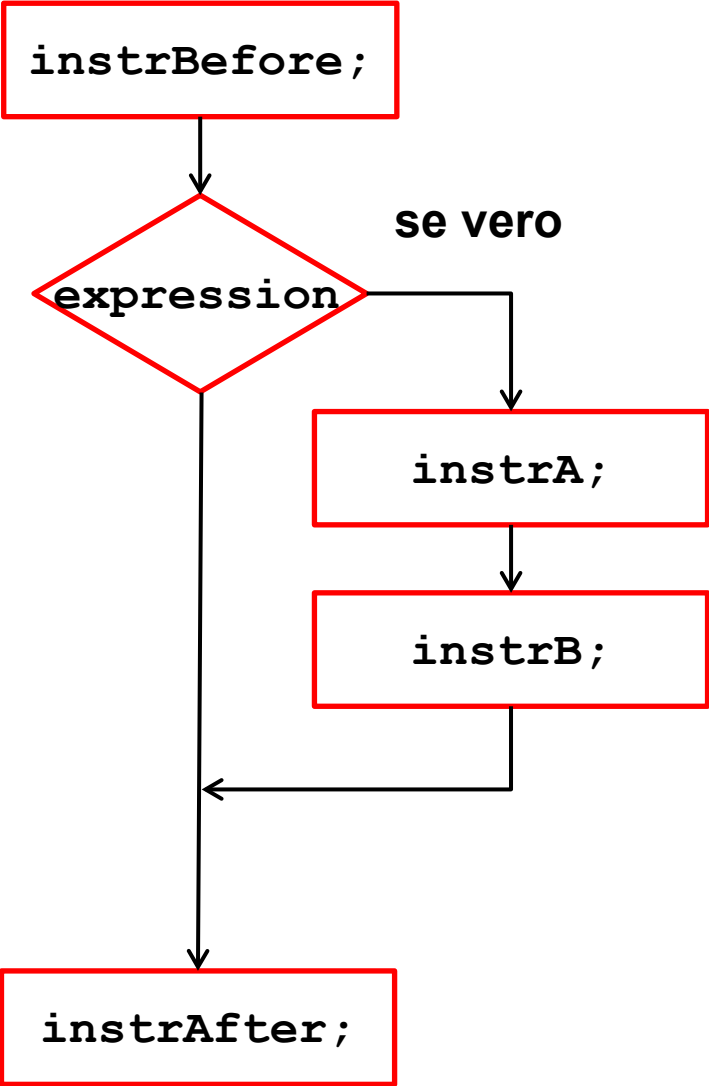


Dopo aver eseguito `instrBefore` si valuta `expression`

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

Altrimenti eseguo `instrC;`

# Costrutto Condizionale:



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


*if (condizione)  
→ [ISTR se vero]  
[else] opzionale*

# Il costrutto condizionale su code.org

code.org STUDIO

Lezione 13: Ape: Istruzioni Condizionali 4 DI PIÙ

Blocchi Area di lavoro: 4 / 4 blocchi Ripr



vai avanti

gira a sinistra ◻ ▾

gira a destra ◻ ▾

prendi il nettare

fai il miele

se nettare ▾ = ▾ 1  
esegui

quando si clicca su "Esegui"

vai avanti

se nettare ▾ > ▾ 0  
esegui prendi il nettare

condizione

Esegui Fai un passo

Controlla questo fiore con un blocco "se" per verificare se ha del nettare.



# Il costrutto condizionale su code.org



Complimenti! Hai completato l'esercizio 4.

Hai appena scritto 3 linee di codice!

Anche le migliori università (p.es., **Berkeley**, **Harvard**) insegnano la programmazione visuale con i blocchi. Ma i blocchi che metti insieme possono essere rappresentati anche in JavaScript, uno dei linguaggi di programmazione più usati al mondo:

```
moveForward();  
if (nectarRemaining() > 0) {  
  getNectar();  
}
```

Istruzione eseguita quando la condizione è vera

Prosegui

...a questo fiore con un blocco "se" per  
...e se ha del nettare.

## Esempio: algoritmo per andare in Università

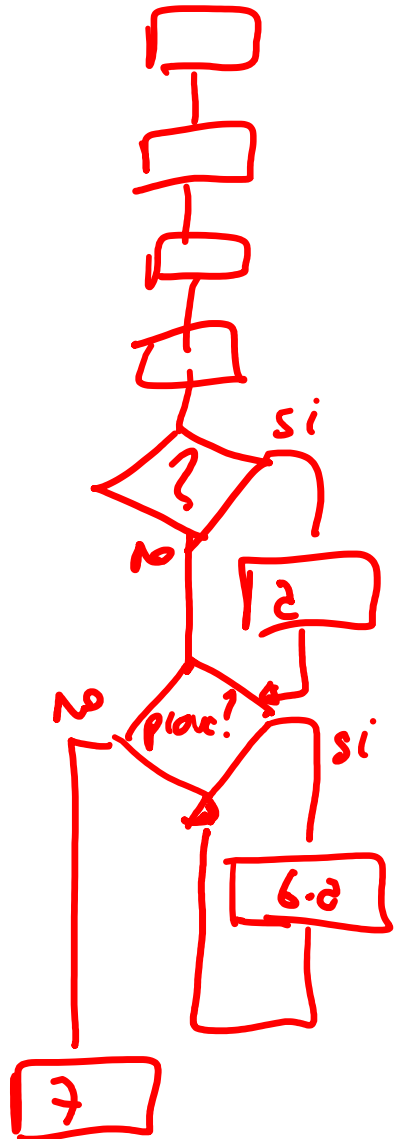
1. Mi alzo quando suona la sveglia
2. Mi dirigo verso la cucina
3. Mangio e bevo un caffè
4. Mi lavo e mi vesto
5. Se piove
  - a. Prendo l'ombrello
6. Esco
7. Corro

# Costrutto Iterativo

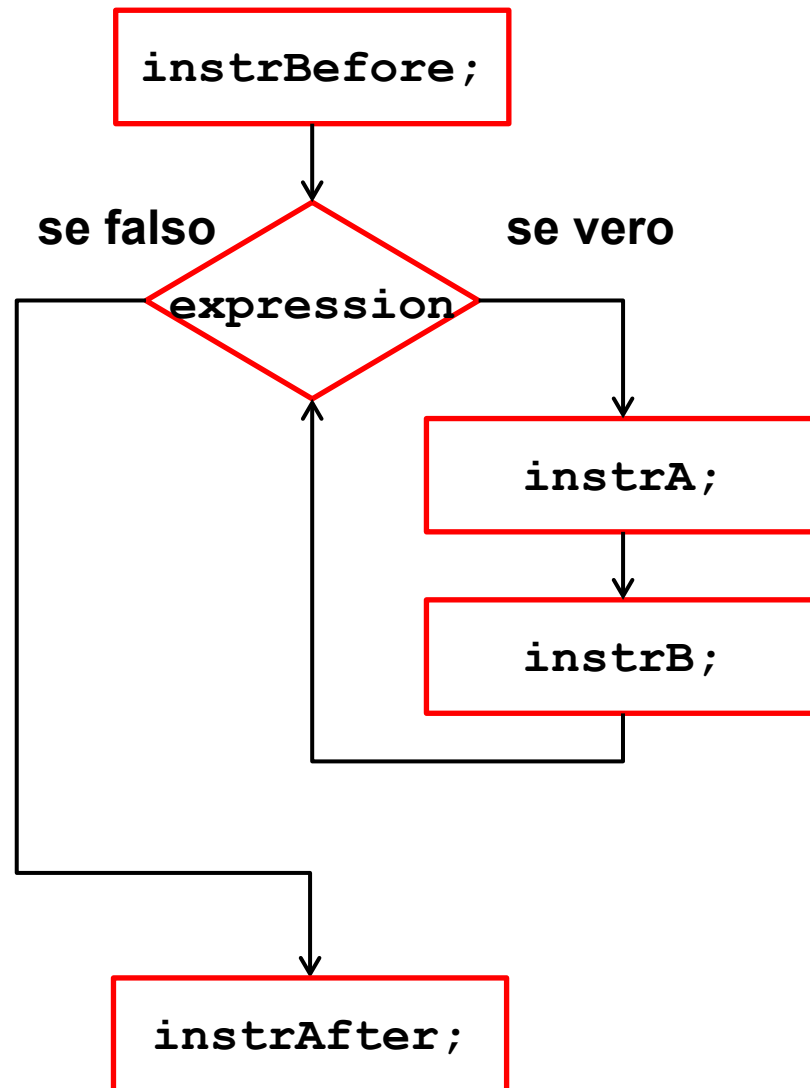
## Esempio: algoritmo per andare in Università

1. Mi alzo quando suona la sveglia
2. Mi dirigo verso la cucina
3. Mangio e bevo un caffè
4. Mi lavo e mi vesto
5. Se c'è il laboratorio di informatica
  - a. Prendo il laptop
6. **Ripeti:** piove ancora?
  - a. Aspetta a casa 5 minuti
7. Vado in stazione
8. Sali sul treno
9. Arrivi a lezione, wow

## Esempio: algoritmo per andare in Università



1. Mi alzo quando suona la sveglia
2. Mi dirigo verso la cucina
3. Mangio e bevo un caffè
4. Mi lavo e mi vesto
5. Se c'è il laboratorio di informatica
  - a. Prendo il laptop
6. **Ripeti:** piove ancora?
  - a. Aspetta a casa 5 minuti
7. Vado in stazione
8. Sali sul treno
9. Arrivi a lezione, wow



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,  
proseguo oltre, altrimenti  
esegui le istruzioni nel corpo  
del ciclo.

**Expression** rappresenta  
quindi la condizione di  
permanenza nel ciclo

# Costrutto Iterativo

```
instrBefore;
```

se falso      se vero

expression

```
instrA;
```

```
instrB;
```

```
instrAfter;
```

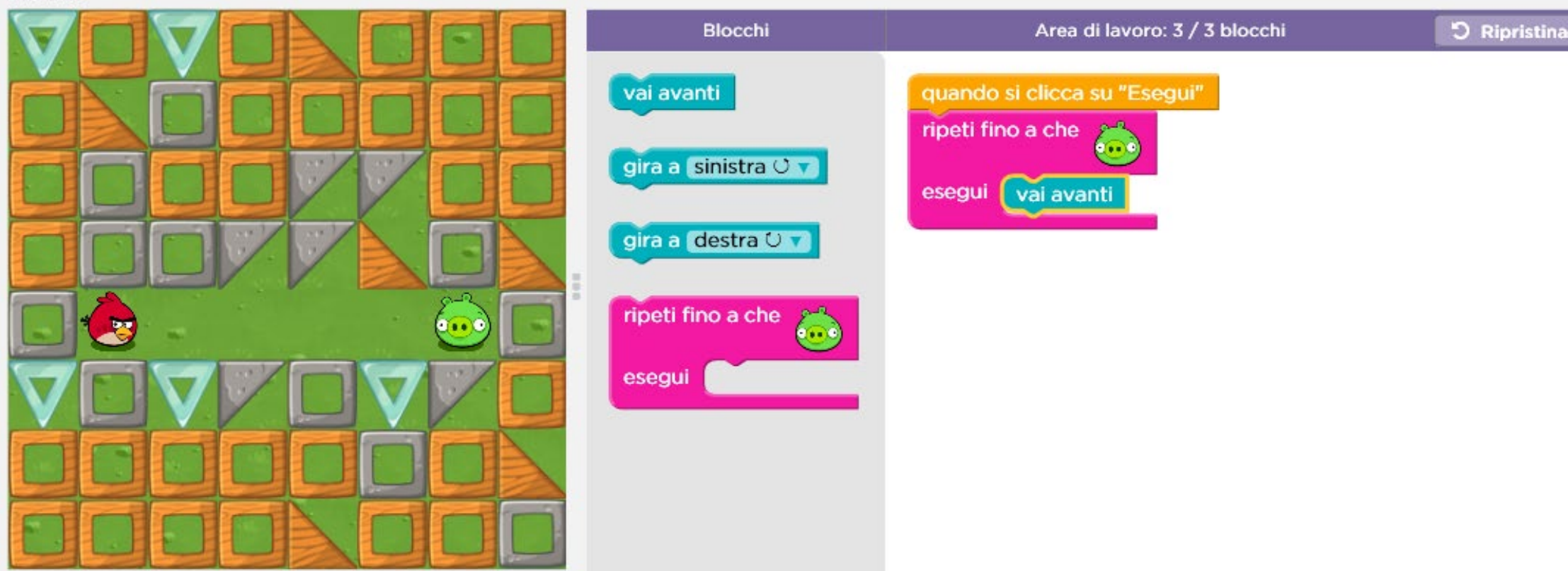
*Condizione  
di permanenza  
nel ciclo  
VALUTATA  
ALLA FINE  
DEL CORPO*

*CORPO*

*potrebbe non venir mai  
eseguito*

# Il costrutto iterativo su code.org

STUDIO



Area di lavoro: 3 / 3 blocchi Ripristina

vai avanti

gira a sinistra

gira a destra


ripeti fino a che

esegui vai avanti

ripeti fino a che

esegui

**Esegui**

 Ok, prova ad usare il nuovo blocco "ripeti fino a che". Esso mi farà "ripetere" le azioni "fino a che" raggiungo quel fastidioso maiale.

**Hai bisogno di aiuto?**

Guarda questi video e suggerimenti



# Il costrutto iterativo su code.org



Complimenti! Hai completato l'esercizio 10.

Hai appena scritto 2 linee di codice!

Totale complessivo: 5 linee di codice.

Anche le migliori università (p.es., **Berkeley, Harvard**) insegnano la programmazione visuale con i blocchi. Ma i blocchi che metti insieme possono essere rappresentati anche in JavaScript, uno dei linguaggi di programmazione più usati al mondo:

```
while (notFinished()) {  
  moveForward();  
}
```

Prosegui

Il nuovo blocco "ripeti fino a ripetere" le azioni "fino a che" idioso maiale.


# Il costrutto iterativo su code.org



L'azione 2: Il labirinto 10

DI PIÙ

Ripristina Mostra




Complimenti! Hai completato l'esercizio 10.

Hai appena scritto 2 linee di codice!  
Totale complessivo: 5 linee di codice.

Anche le migliori università (p.es., **Berkeley, Harvard**) insegnano la programmazione visuale con i blocchi. Ma i blocchi che metti insieme possono essere rappresentati anche in JavaScript, uno dei linguaggi di programmazione più usati al mondo:

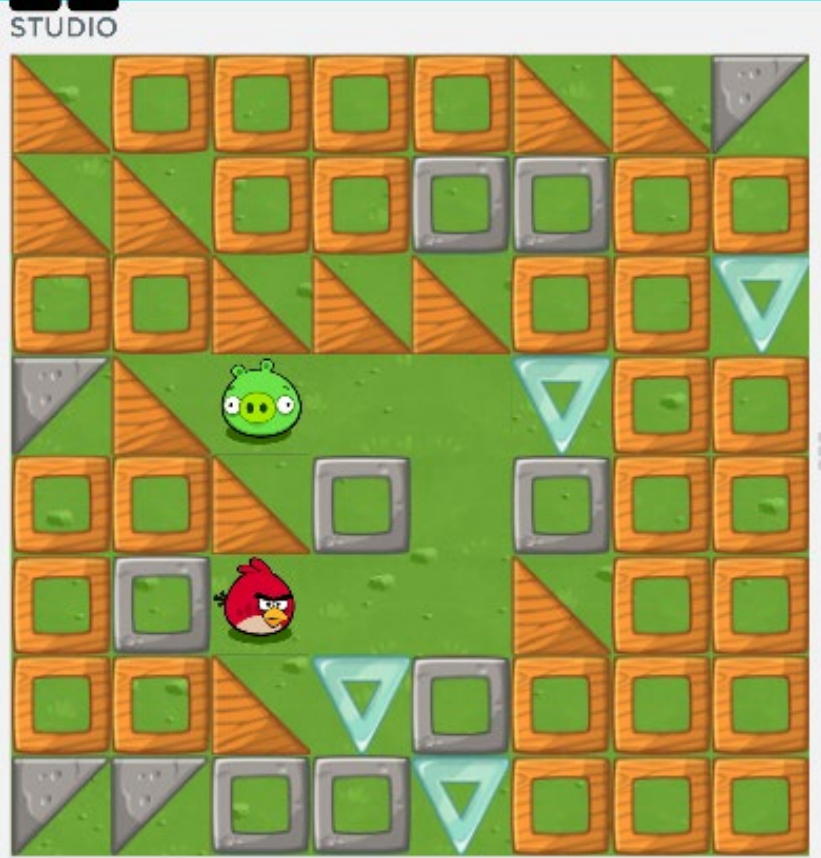
```
while (notFinished()) {  
  moveForward();  
}
```




Prosegui

Il nuovo blocco "ripeti fino a ripetere" le azioni "fino a che" idioso maiale.

# Il costrutto iterativo su code.org



 Esegui

 Ok, un'ultima volta per fare pratica: riesci a risolvere questo esercizio utilizzando solo 4 blocchi?

[Hai bisogno di aiuto?](#)

Blocchi Area di lavoro: 5 / 5 blocchi

```
vai avanti
gira a sinistra
gira a destra


ripeti fino a che
  esegui
    vai avanti
    vai avanti
    gira a sinistra

ripeti fino a che
  esegui
```

# Il costrutto iterativo su code.org



 Esegui

 Caro umano. Io zombie. Io affamato. Devo ... arrivare ... al girasole. Riesci a farmi arrivare là con solo 5 blocchi?

**Hai bisogno di aiuto?**

Guarda questi video e suggerimenti

Blocchi Area di lavoro: 6 / 6 blocchi

```
vai avanti
gira a sinistra
gira a destra

ripeti fino a che
  esegui
    vai avanti
    gira a sinistra
    vai avanti
    gira a destra
    Gira a sinistra o a destra di 90 gradi.

ripeti fino a che
  esegui
```

# Il costrutto iterativo su code.org

STUDIO



Esegui



Caro umano. Io zombie. Io affamato. Devo ... arrivare ... al girasole. Riesci a farmi arrivare là con solo 5 blocchi?

Hai bisogno di aiuto?

Guarda questi video e suggerimenti

Blocchi

Area di lavoro: 6 / 6 blocchi

vai avanti

gira a sinistra ↶

gira a destra ↷

ripeti fino a che

esegui

quando si clicca su "Esegui"

ripeti fino a che

esegui

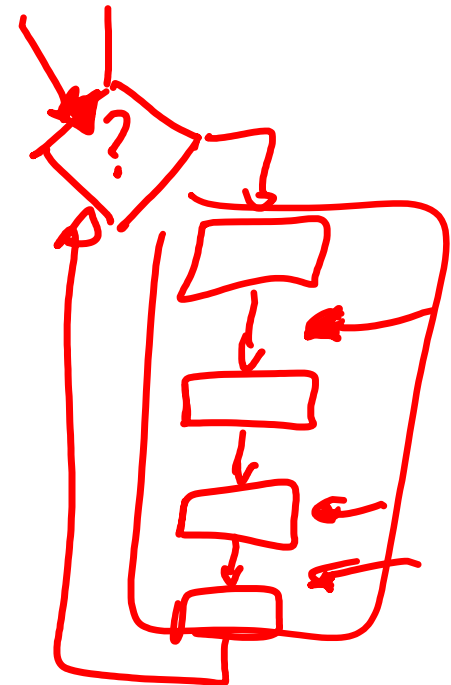
vai avanti

gira a sinistra ↶

vai avanti

gira a destra ↷

Gira a sinistra o a destra di 90 gradi.



# Il costrutto iterativo su code.org

C O  
D E  
STUDIO

Lezione 2: Il labirinto

12

DI PIÙ



Ricomincia



Caro umano, lo zombie, lo affamato. Devi arrivare ... al girasole. Riesci a farmi arrivare con solo 5 blocchi?

Hai bisogno di aiuto?



## Complimenti! Hai completato l'esercizio 12.

Hai appena scritto 5 linee di codice!

Totale complessivo: 14 linee di codice.

Anche le migliori università (p.es., **Berkeley, Harvard**) insegnano la programmazione visuale con i blocchi. Ma i blocchi che metti insieme possono essere rappresentati anche in JavaScript, uno dei linguaggi di programmazione più usati al mondo:

```
while (notFinished()) {  
  moveForward();  
  turnLeft();  
  moveForward();  
  turnRight();  
}
```

<

>

Prosegui




Ripristina



# Il costrutto iterativo e condizionale su code.org

STUDIO





Area di lavoro: 4 / 5 blocchi

Blocchi

- vai avanti
- gira a sinistra
- gira a destra
- ripeti fino a che
- se c'è strada a sinistra

 Esegui

 Qui il nuovo blocco "se" viene usato per farti decidere quando girare. Fai attenzione: adesso hai bisogno di inserire un solo blocco, ma capisci bene il procedimento in modo che puoi farlo tutto da solo la prossima volta.

```
quando si clicca su "Esegui"
  ripeti fino a che 
    esegui
      vai avanti
      se c'è strada a sinistra
        esegui
```


Ripete le azioni incluse, smettendo quando diventa

Hai bisogno di aiuto?

# Il costrutto iterativo e condizionale su code.org



 Esegui


 Qui il nuovo blocco "se" viene usato per farti decidere quando girare. Fai attenzione: adesso hai bisogno di inserire un solo blocco, ma capisci bene il procedimento in modo che puoi farlo tutto da solo la prossima volta.

Blocchi

Area di lavoro: 4 / 5 blocchi

- vai avanti
- gira a sinistra
- gira a destra
- ripeti fino a che
- se c'è strada a sinistra

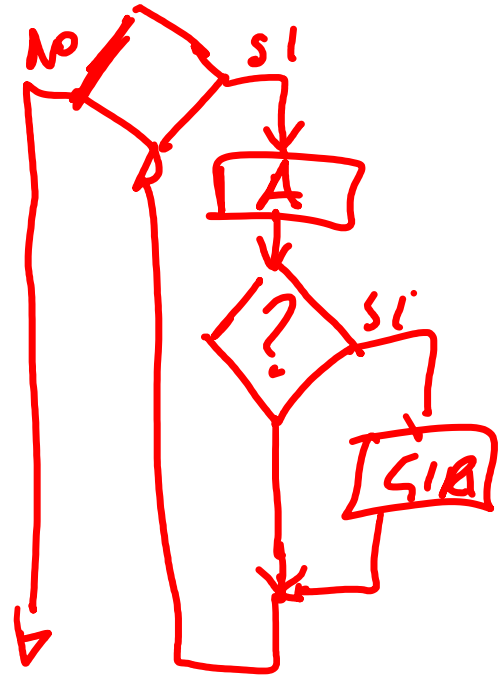
quando si clicca su "Esegui"

ripeti fino a che 

esegui

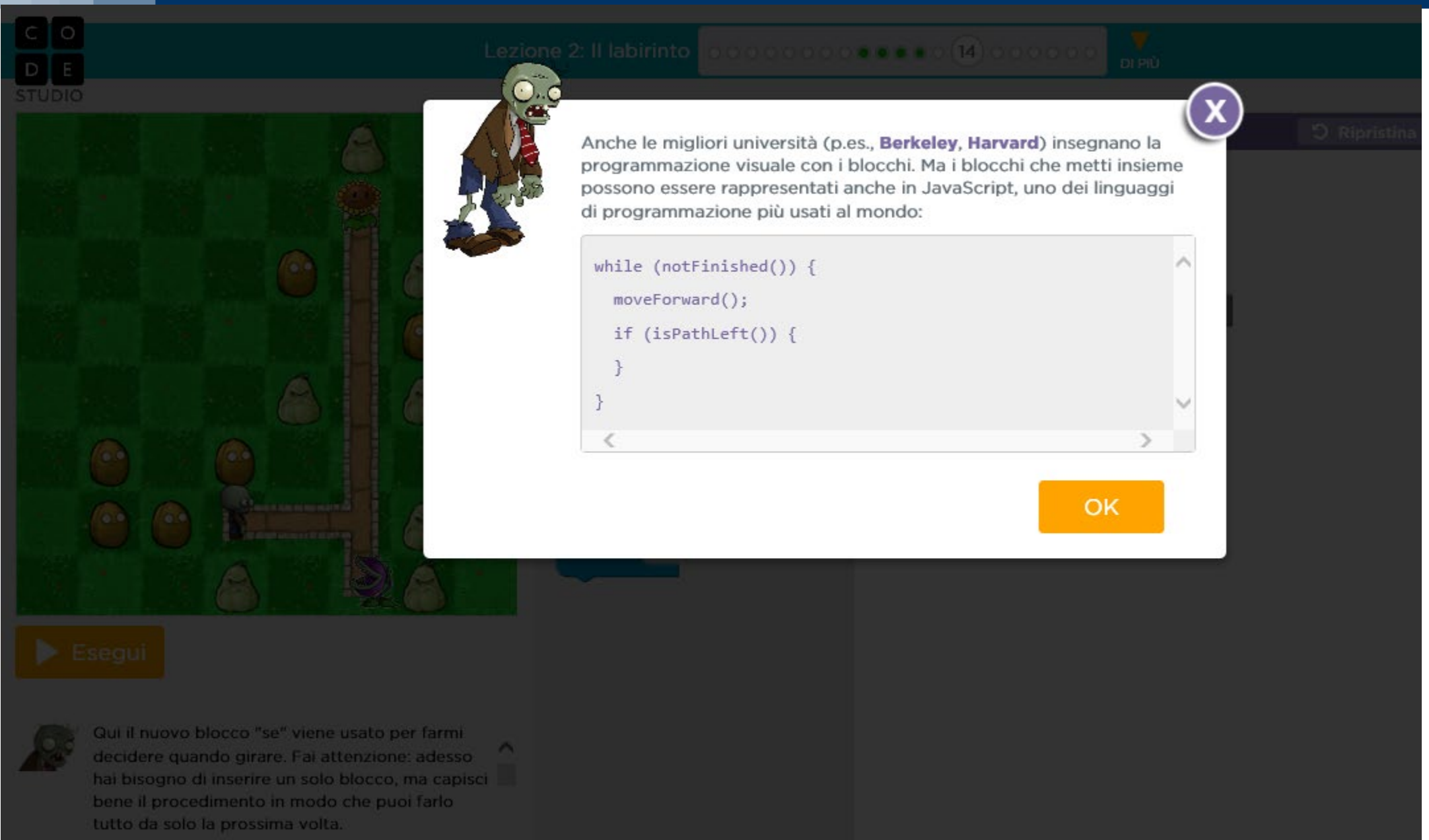
- vai avanti
- se c'è strada a sinistra
- esegui

Ripete le azioni incluse, smettendo quando diventa





# Il costrutto iterative su code.org



Lezione 2: Il labirinto 14 DI PIÙ

STUDIO

Ripristina

Anche le migliori università (p.es., **Berkeley, Harvard**) insegnano la programmazione visuale con i blocchi. Ma i blocchi che metti insieme possono essere rappresentati anche in JavaScript, uno dei linguaggi di programmazione più usati al mondo:

```
while (notFinished()) {  
  moveForward();  
  if (isPathLeft()) {  
  }  
}
```

OK

Esegui

Qui il nuovo blocco "se" viene usato per farti decidere quando girare. Fai attenzione: adesso hai bisogno di inserire un solo blocco, ma capisci bene il procedimento in modo che puoi farlo tutto da solo la prossima volta.

# Esempi di Algoritmi

## Il Pallottoliere

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

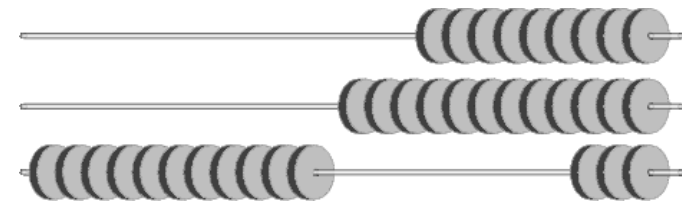
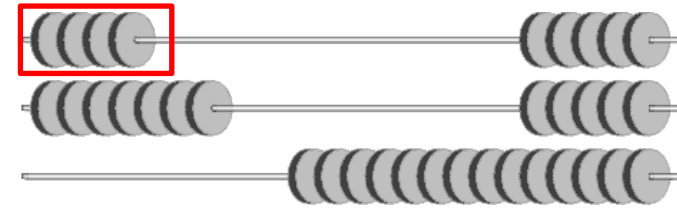
Scriviamo le istruzioni per utilizzare il pallottoliere (utilizzando un costrutto iterativo)



# Algoritmo per sommare col pallottoliere

Supponiamo che:

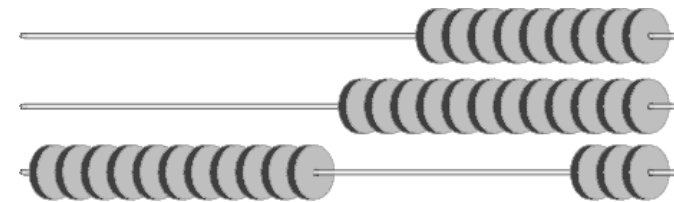
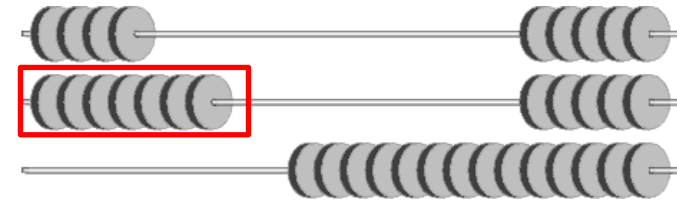
- Primo addendo sia riportato sulla prima riga a sinistra
- Secondo addendo sia sulla seconda riga a sinistra
- Risultato deve apparire nella terza riga (e che all'inizio questa abbia tutte le palline a destra).
- Operiamo con numeri «piccoli», non c'è bisogno del riporto



# Algoritmo per sommare col pallottoliere

Supponiamo che:

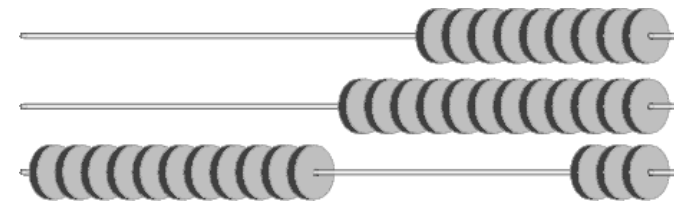
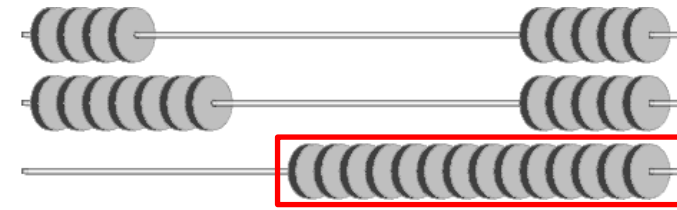
- Primo addendo sia riportato sulla prima riga a sinistra
- Secondo addendo sia sulla seconda riga a sinistra
- Risultato deve apparire nella terza riga (e che all'inizio questa abbia tutte le palline a destra).
- Operiamo con numeri «piccoli», non c'è bisogno del riporto



# Algoritmo per sommare col pallottoliere

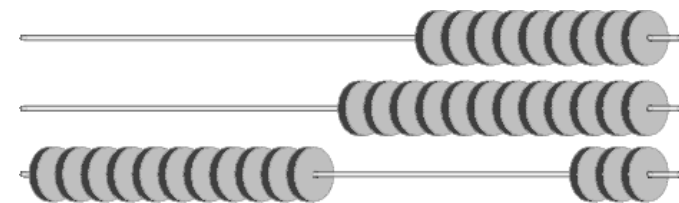
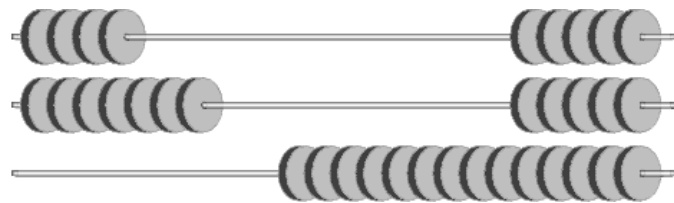
Supponiamo che:

- 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.



# Proprietà fondamentali degli algoritmi

## 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

## 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

## 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.

## 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.

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**

## 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

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

## 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

AUTORE / I:  
GHEZZI CARLO,  
JAZAYERI MEHDI.

TITOLO:  
PROGRAMMING LANGUAGE CONCEPTS.  
1981.

SCAFFALE 35  
POSIZIONE 21

## Algoritmo: Ricerca

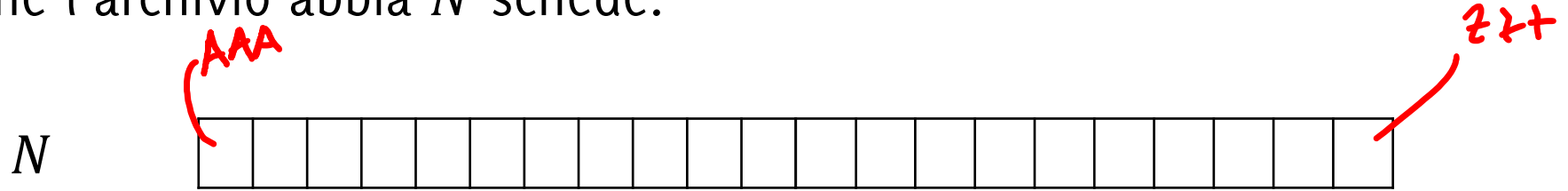
Non tutti gli esecutori sono in grado di «cercare», e comunque la ricerca può essere fatta in diversi modi

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**  $\Rightarrow$  ricerca conclusa con successo
  - **altrimenti**  $\Rightarrow$  ricerca conclusa con insuccesso



# Algoritmo: Ricerca

Assumendo che l'archivio abbia  $N$  schede:

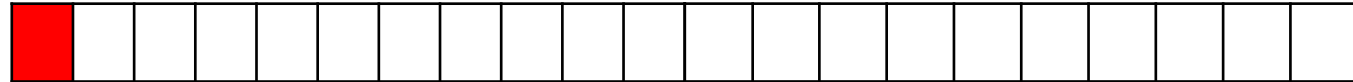


Assumendo che l'archivio abbia  $N$  schede:

$N$



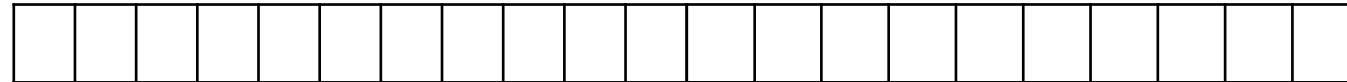
Controllo  
prima scheda



# Algoritmo: Ricerca

Assumendo che l'archivio abbia  $N$  schede:

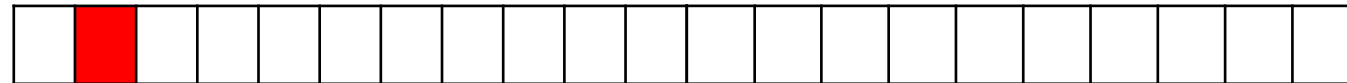
$N$



Controllo  
prima scheda

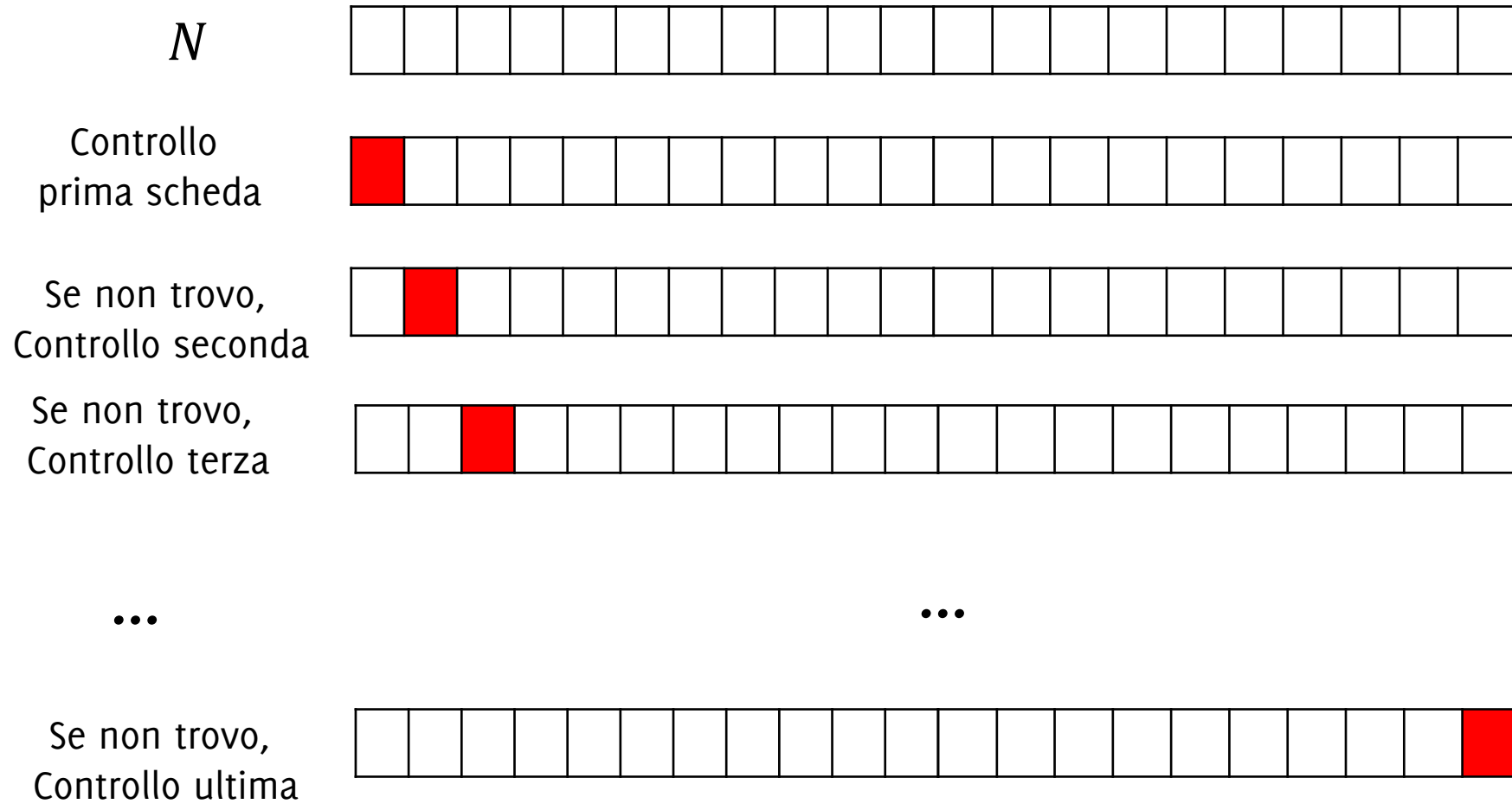


Se non trovo,  
Controllo seconda



# 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**  $\Rightarrow$  ricerca conclusa con successo
  - **altrimenti**  $\Rightarrow$  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

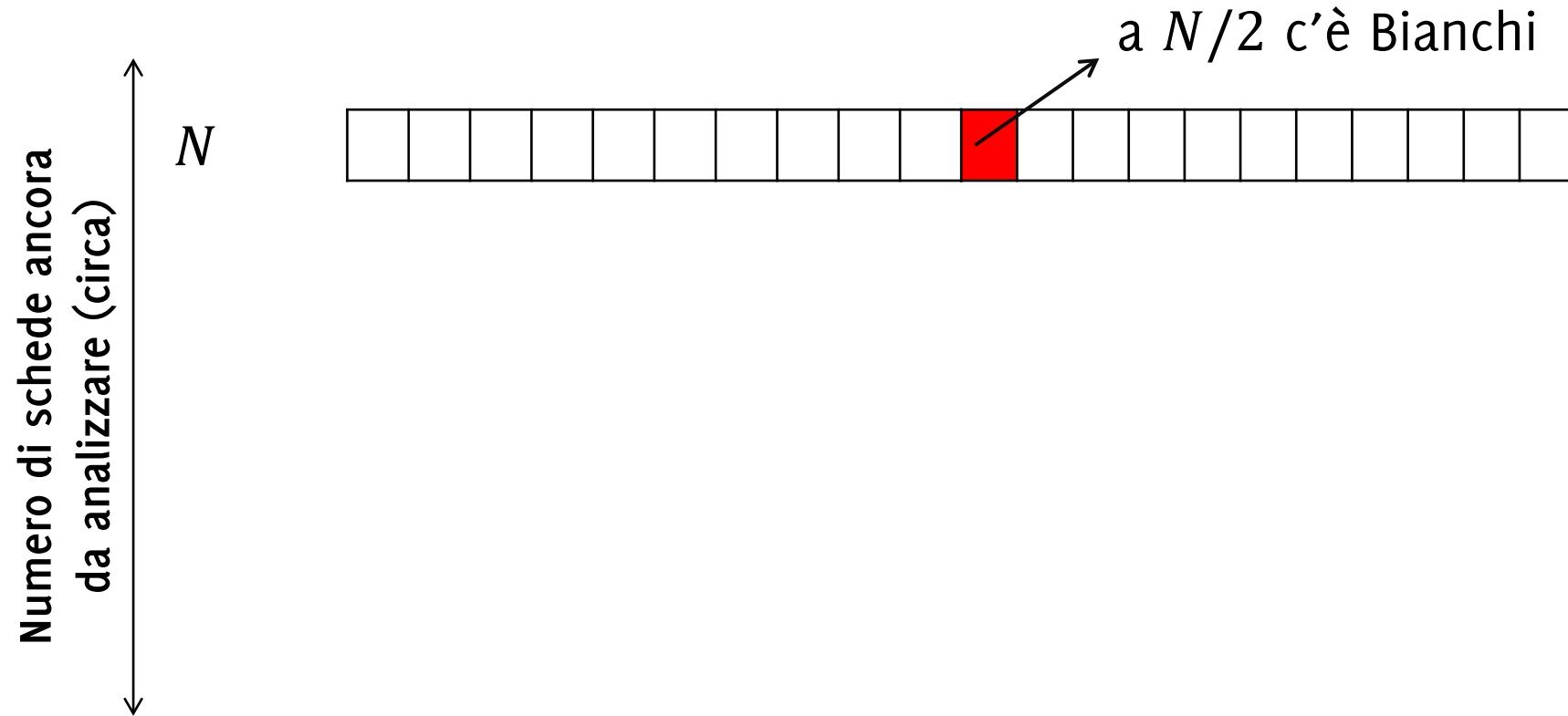
Lo schedario contiene  $N$  schede ordinate in cui cercare

1. prendi la scheda centrale, 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

**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

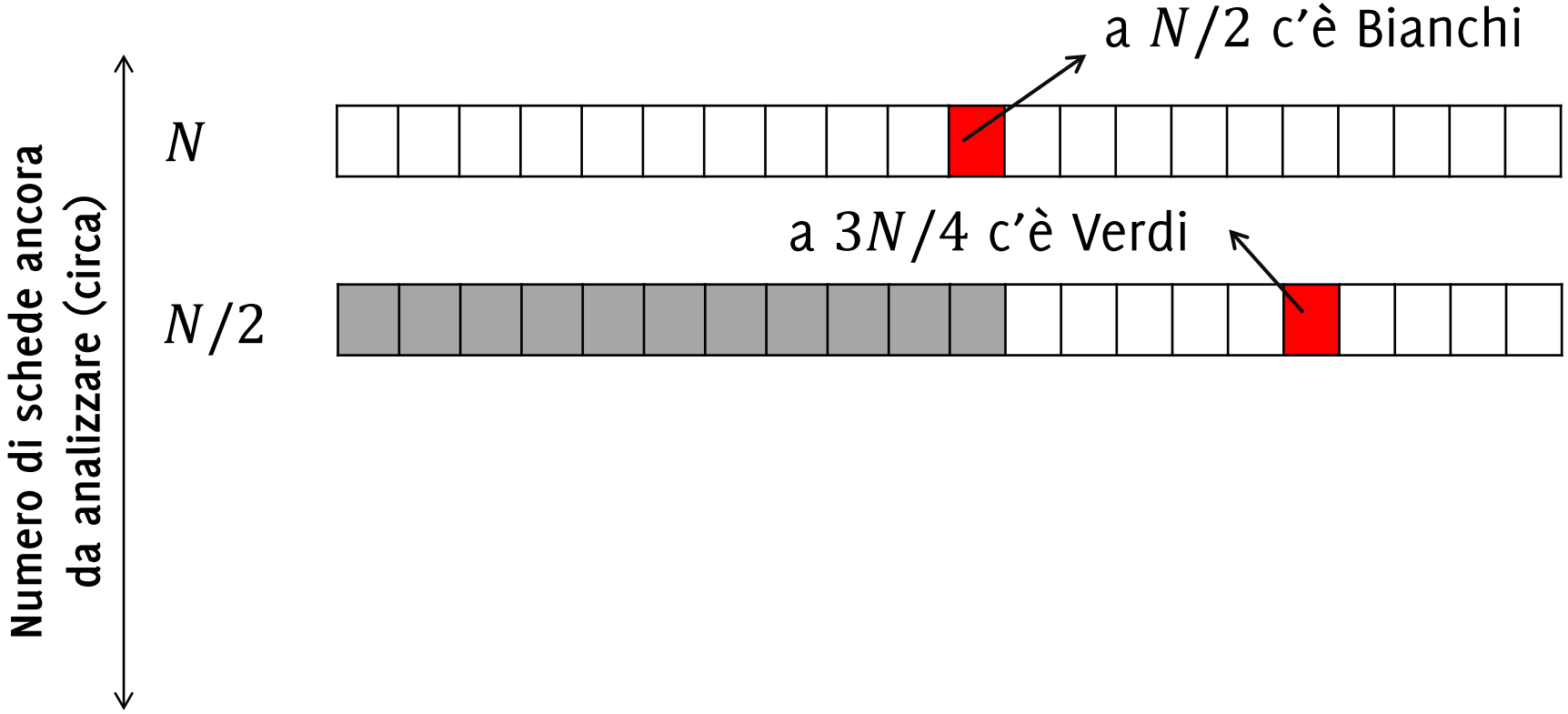
es: Ricerca dell'autore ROSSI





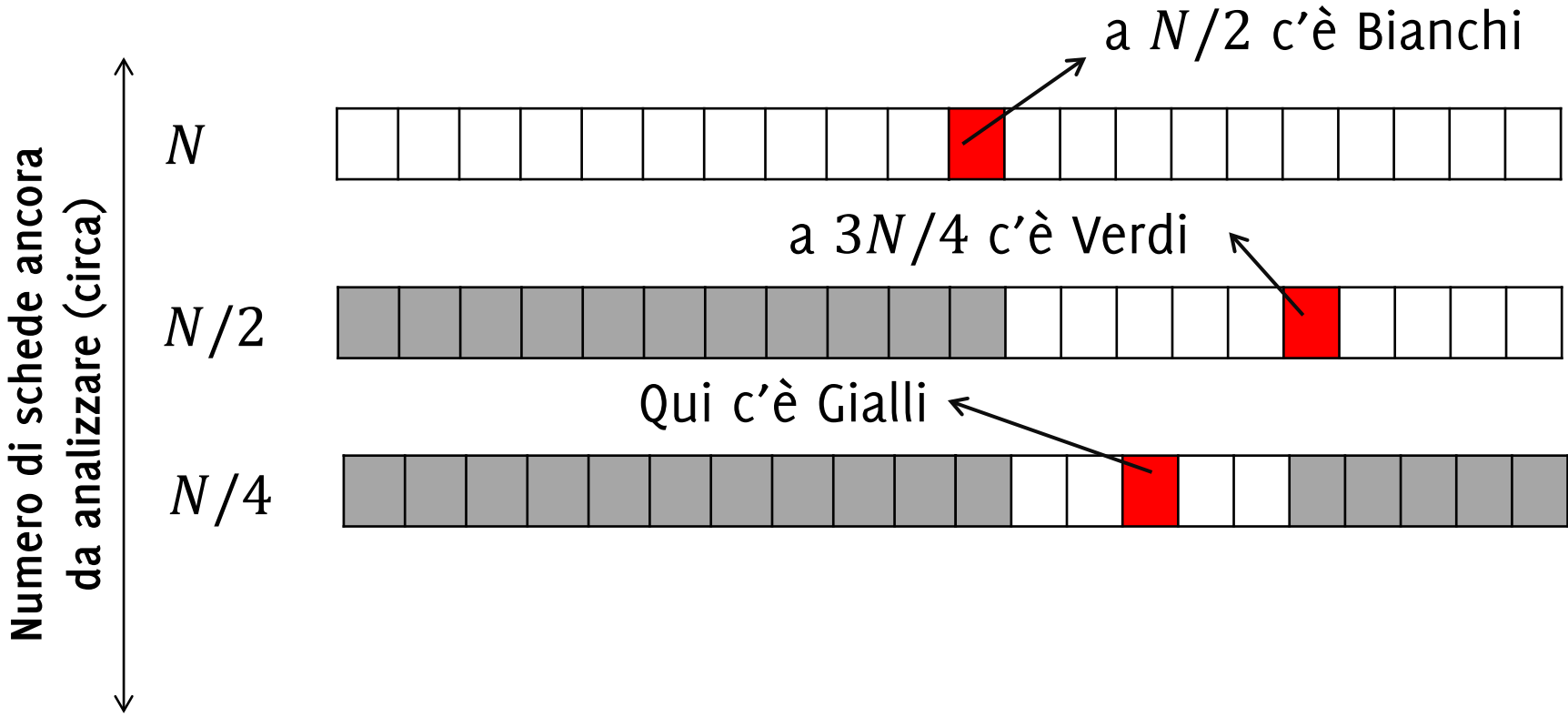
# Algoritmo: Ricerca tra elementi ordinati

es: Ricerca dell'autore ROSSI



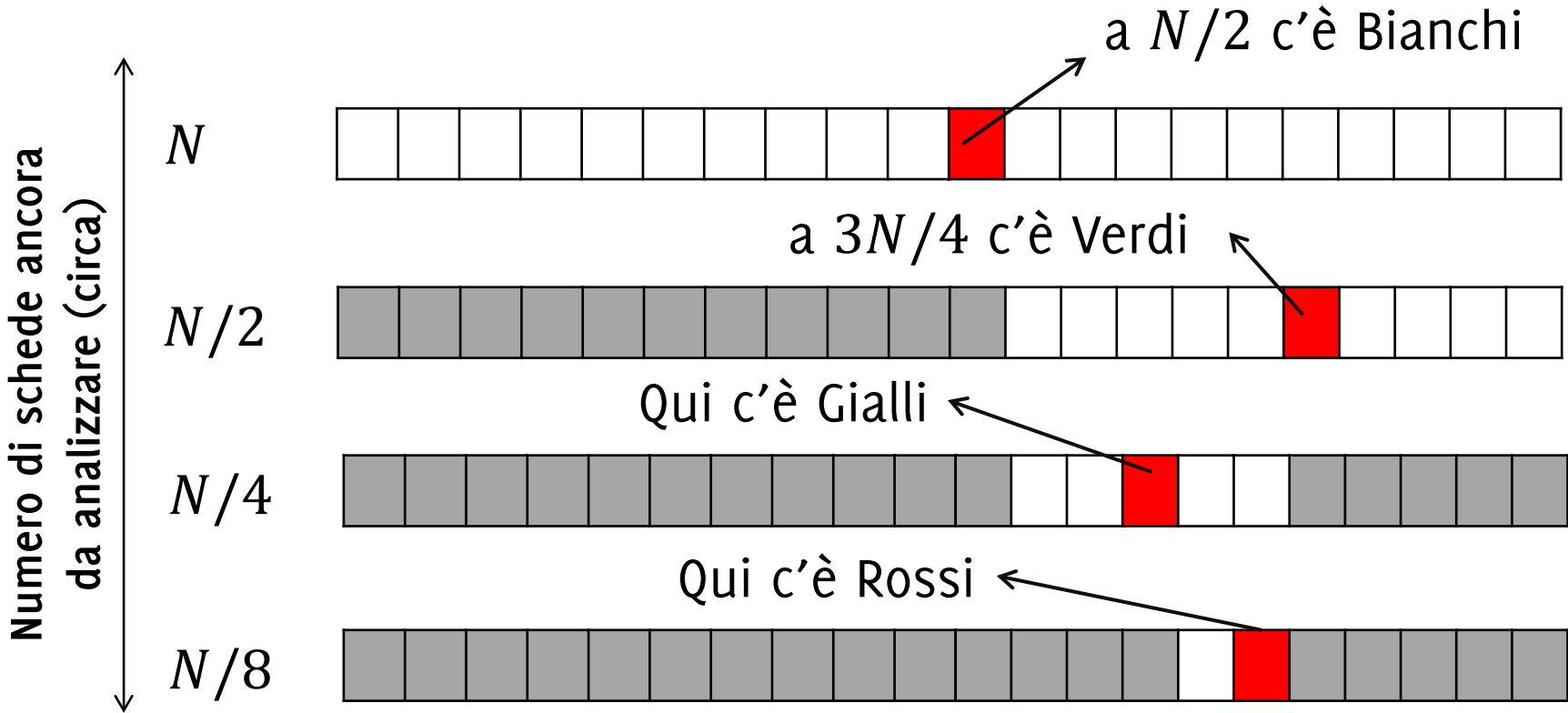
# Algoritmo: Ricerca tra elementi ordinati

es: Ricerca dell'autore ROSSI



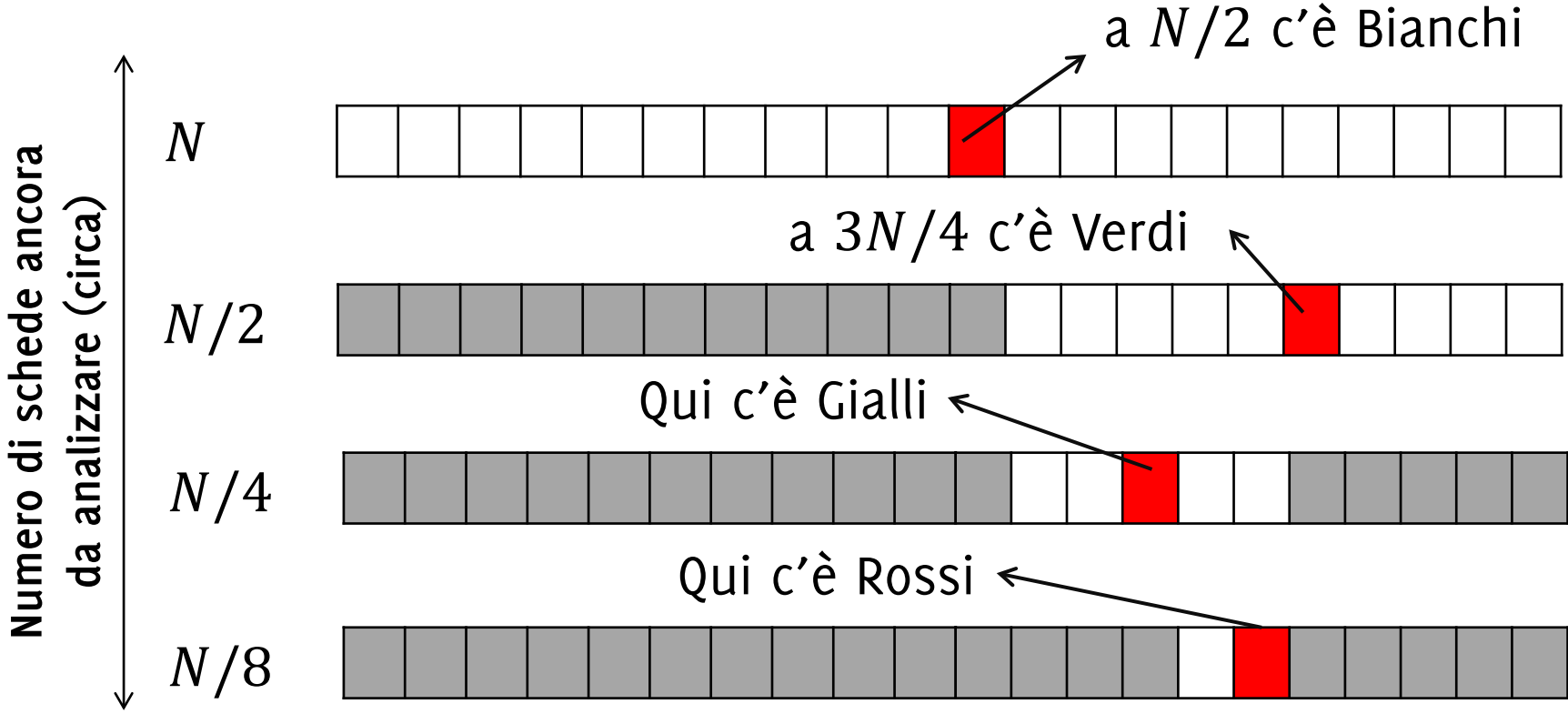
# Algoritmo: Ricerca tra elementi ordinati

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, **finchè** 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