Introduzione al Corso

Informatica a, AA 2024/2025

Giacomo Boracchi

giacomo.boracchi@polimi.it

16 Settembre 2024

Per Evitare Confusione

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)

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

Chi siamo

Giacomo Boracchi (docente)

- Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano
- homepage: https://boracchi.faculty.polimi.it
- email 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. Alessandro Montenegro (esercitatore)

- email: alessandro.montenegro@polimi.it
- Pagina esercitazioni:
  - https://montenegroalessandro.github.io/InfoA2425/index.html

Chi siamo

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

- email: edoardo.peretti@polimi.it

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

- email: gianenrico.conti@polimi.it

Pagina Laboratorio: **TBD**

Il Corso

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

La Pagina del Corso

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 2.0.1**
- Circa un Mercoledì su due dalle **13:30 alle 15:15 Aula B.2.4**
- Spesso il Venerdì dalle **8:45 alle 11:15 Aula T.2.3**

Le esercitazioni si terranno

- Spesso il Lunedì dalle **13:30 alle 16:15, Aula 2.0.1**
- Circa un Mercoledì su due dalle **13:30 alle 15:15 Aula B.2.4**
- Raramente il Venerdì dalle **8:45 alle 11:15 Aula T.2.3**

Organizzazione, laboratori

I Laboratori si terranno in presenza e partiranno dal 26 Settembre

- Martedì **in Aula 21.0.1 oppure 21.0.2**
- Mercoledì **in Aula 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

boracchi.faculty.polimi.it/teaching/InfoACalendar.htm

Photos TIP Projects DEIB Cineca Insieme Salute GB Scheda intranetRicerca MQ nVidia Grai

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 (mappa)
altri dettagli» copia nel mio calendario

venerdì, 15 settembre

08:45 **Lezione**
Quando ven, 15 settembre, 08:45 – 11:00
Dove Aula 5.1.1 (mappa)
altri dettagli» copia nel mio calendario

lunedì, 18 settembre

13:30 **Lezione**
Quando lun, 18 settembre, 13:30 – 16:00
Dove Aula 5.0.3 (mappa)
altri dettagli» copia nel mio calendario

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

Eventi mostrati nel fuso orario: Ora dell'Europa centrale - Roma

Google Calendario

Lezioni ed Esercitazioni

How to..

Modalità di Insegnamento

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)**

Modalità di Insegnamento

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'in
- 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

I 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.

I Laboratori

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

Sarete divisi in due squadre

- **'Squadra 1' CP DISPARI:**
- **'Squadra**

Il Laboratorio è

- prendere
- **consolida** nti introdotti a lezione.

Non si impara a programmare su carta!

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

I Laboratori, Consigli

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

I testi degli esercizi di laboratorio verranno pubblicati sul sito del docente già da 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

I Laboratori

Vi invitiamo ad installare l'ambiente di programmazione

- Installare Code::Blocks per il C, versione con compilatore (http://www.codeblocks.org/)
- Potete anche usare alternative come Xcode (utenti iOS) o DevC++
- 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 oppure https://boracchi.faculty.polimi.it/teaching/InfoA/Install_code_blocks%20_win11.pdf

- Potrete usare quelli del laboratorio, accedendo con le vostre credenziali a **virtualdesktop**. Trovate qui tutte le indicazioni https://www.ict.polimi.it/software/virtual-desktop-software-for-study-and-teaching/?lang=en
- Guardate sempre il calendario per trovare gli orari e l'aula del vostro turno.

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?»

Richieste via mail

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.

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.
- **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

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

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 [illegible] 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

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

Bibliografia

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

Algoritmo

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

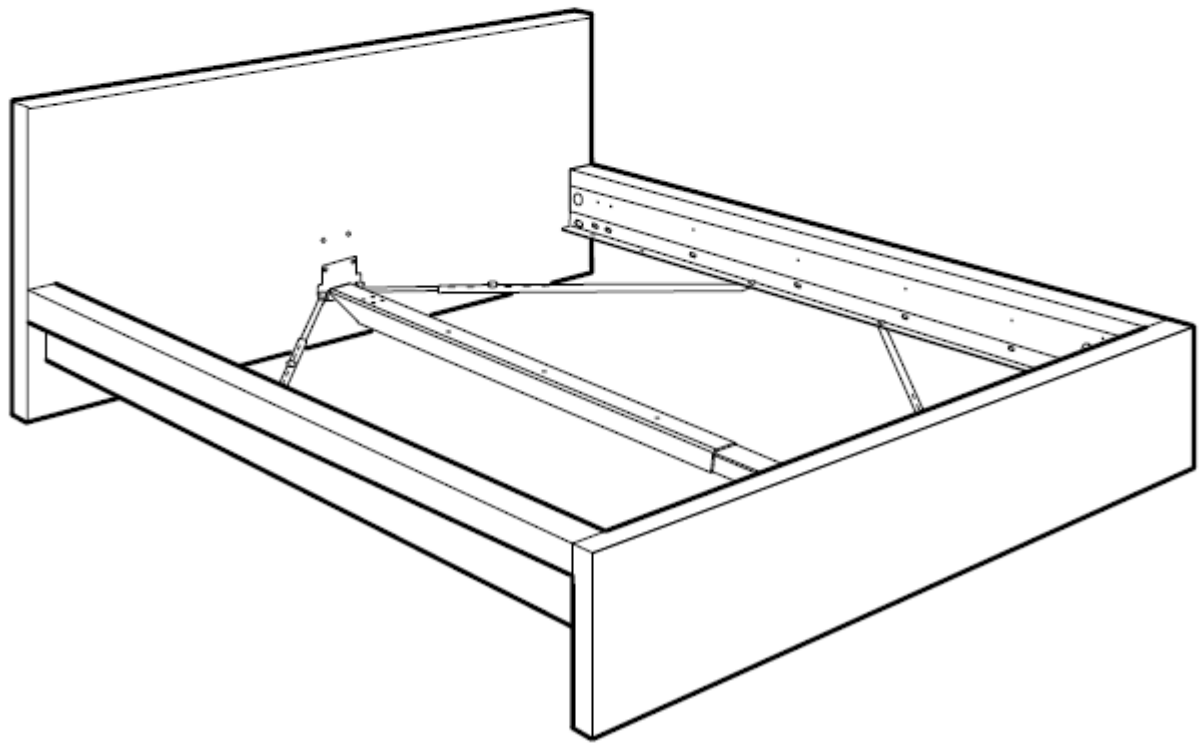
Le operazioni devono:

- essere **comprensibili** senza ambiguità
- essere **eseguibili** da uno strumento automatico: l'esecutore (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 vedermo un esempio di algoritmo in cui vi sarà capitato di essere esecutori

MALM



Design and Quality
IKEA of Sweden

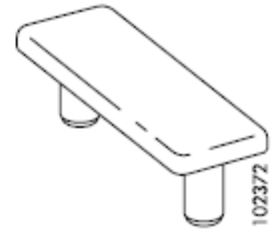
12x

20x/22x

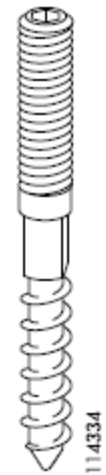
8x

12x

8x



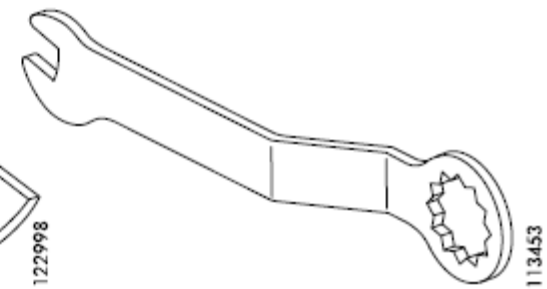
6x



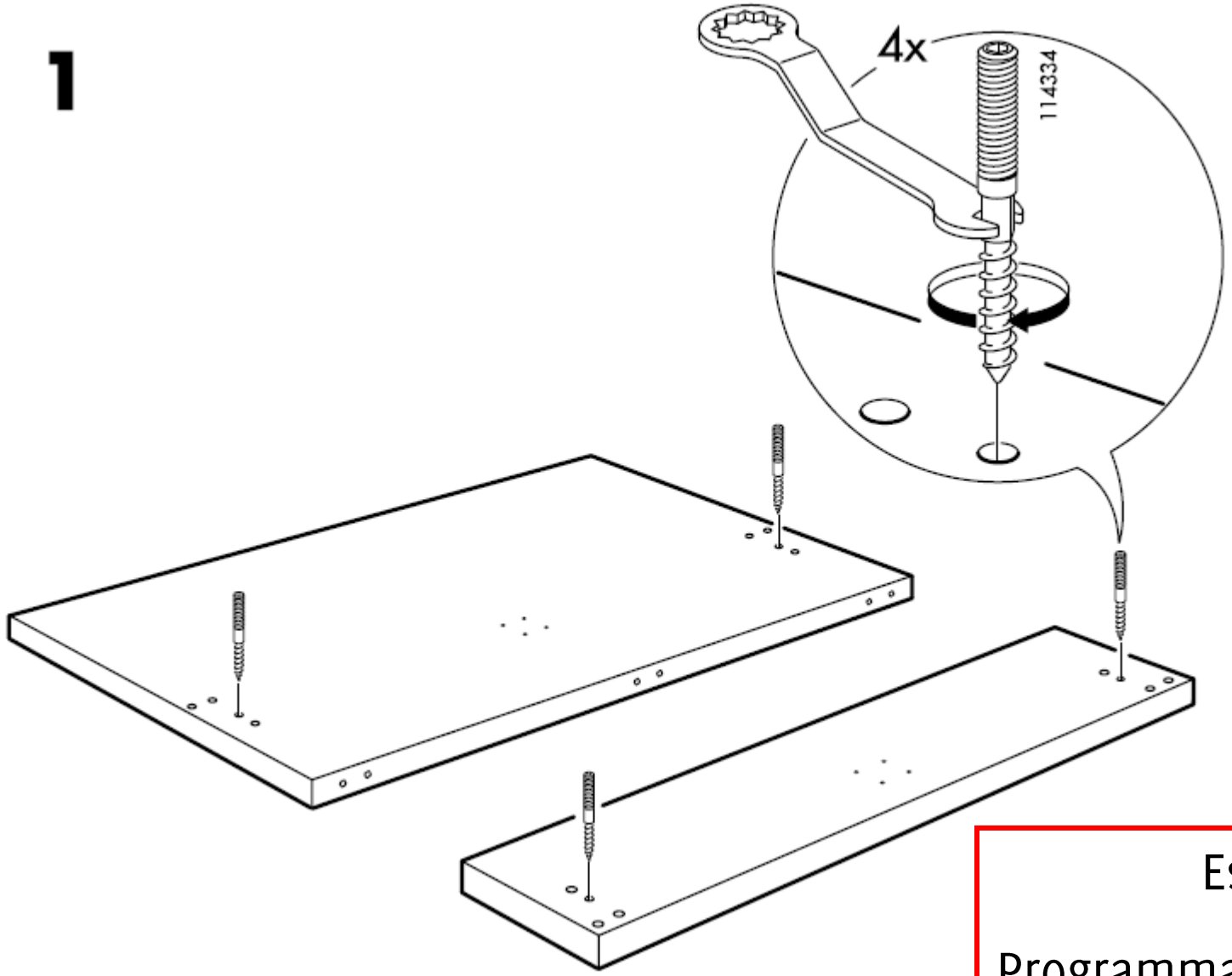
4x

4x

4x



1x

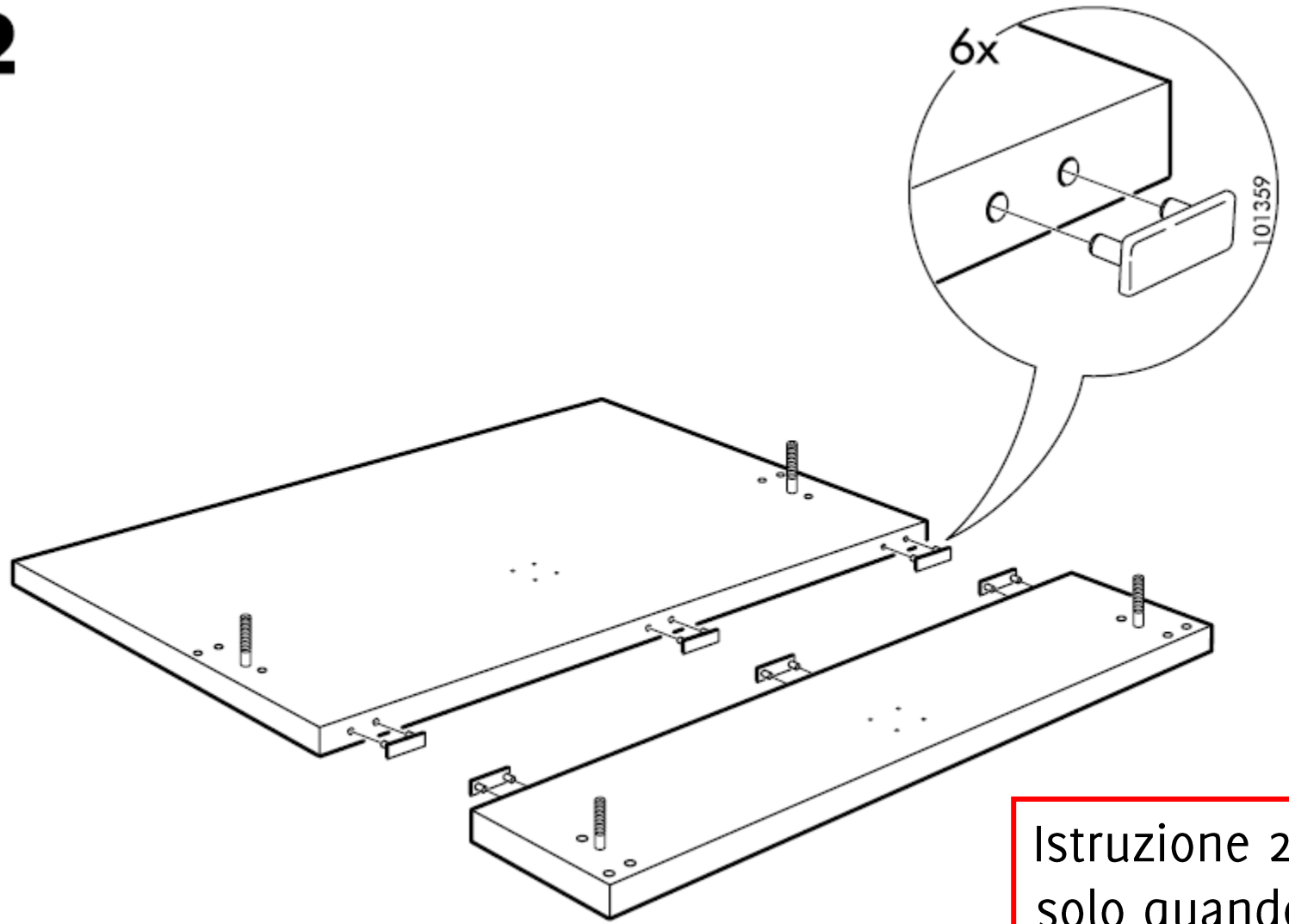


Esecutore: Voi

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

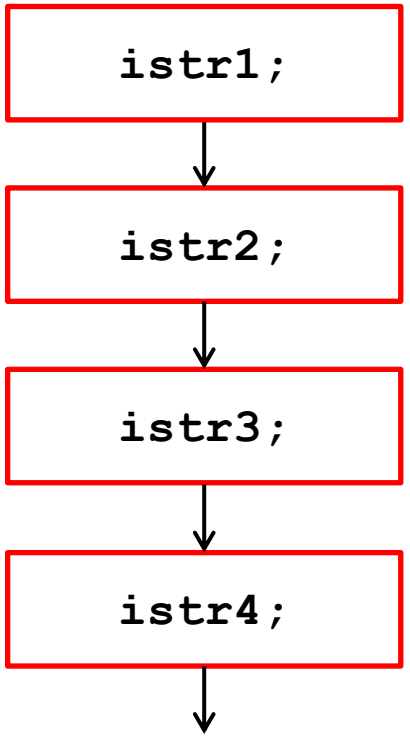
La Sequenzialità

Esempio: algoritmo per andare in università

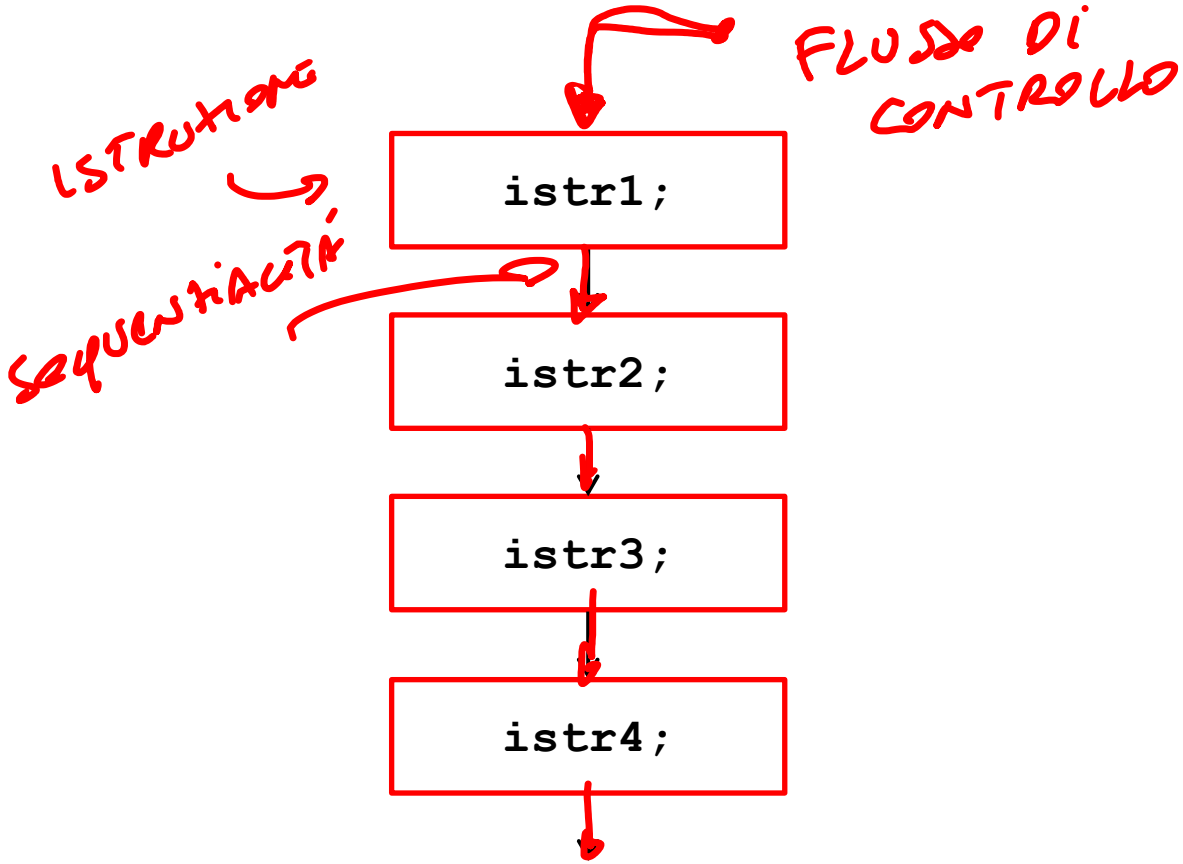
1. Mi alzo quando suona la sveglia
2. Mi dirigo verso la cucina
3. Mangio e bevo un caffè
4. Mi lavo
5. Mi vesto
6. Esco
7. Corro

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

La sequenzialità



La sequenzialità

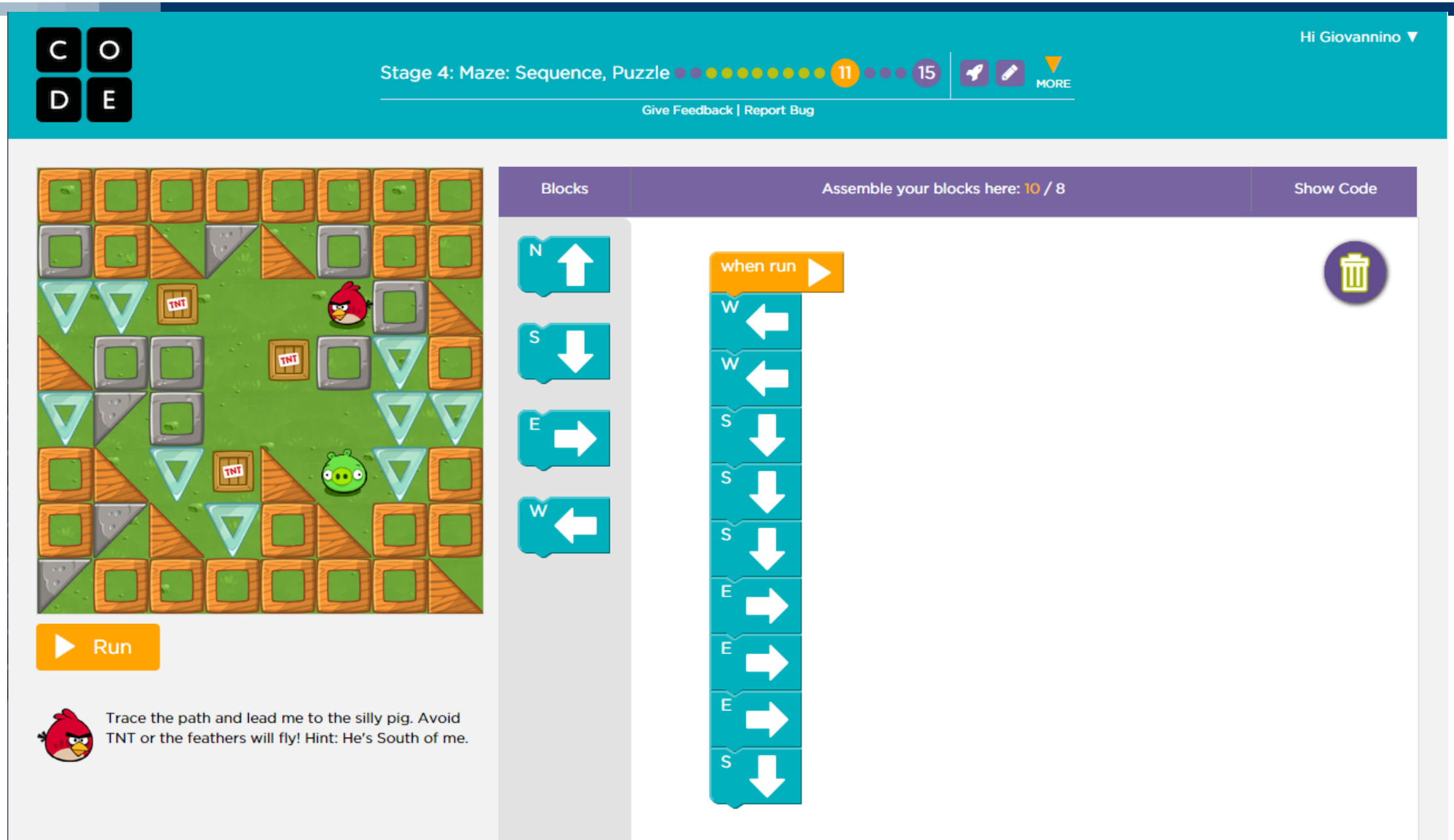



```
istr1;
istr2;
istr3;
istr4;
...
```

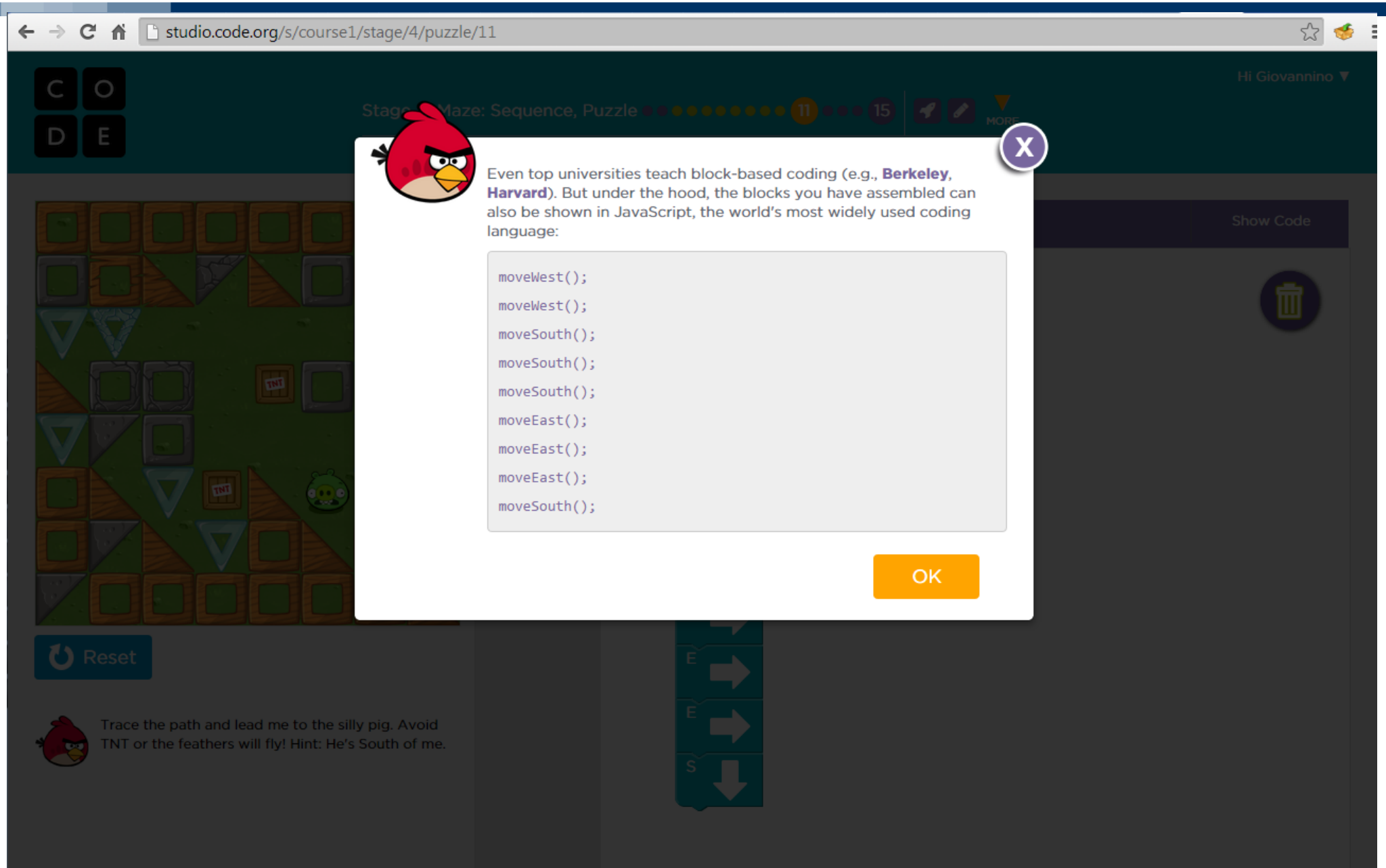
Le istruzioni vengono eseguite **dalla prima all'ultima**.

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

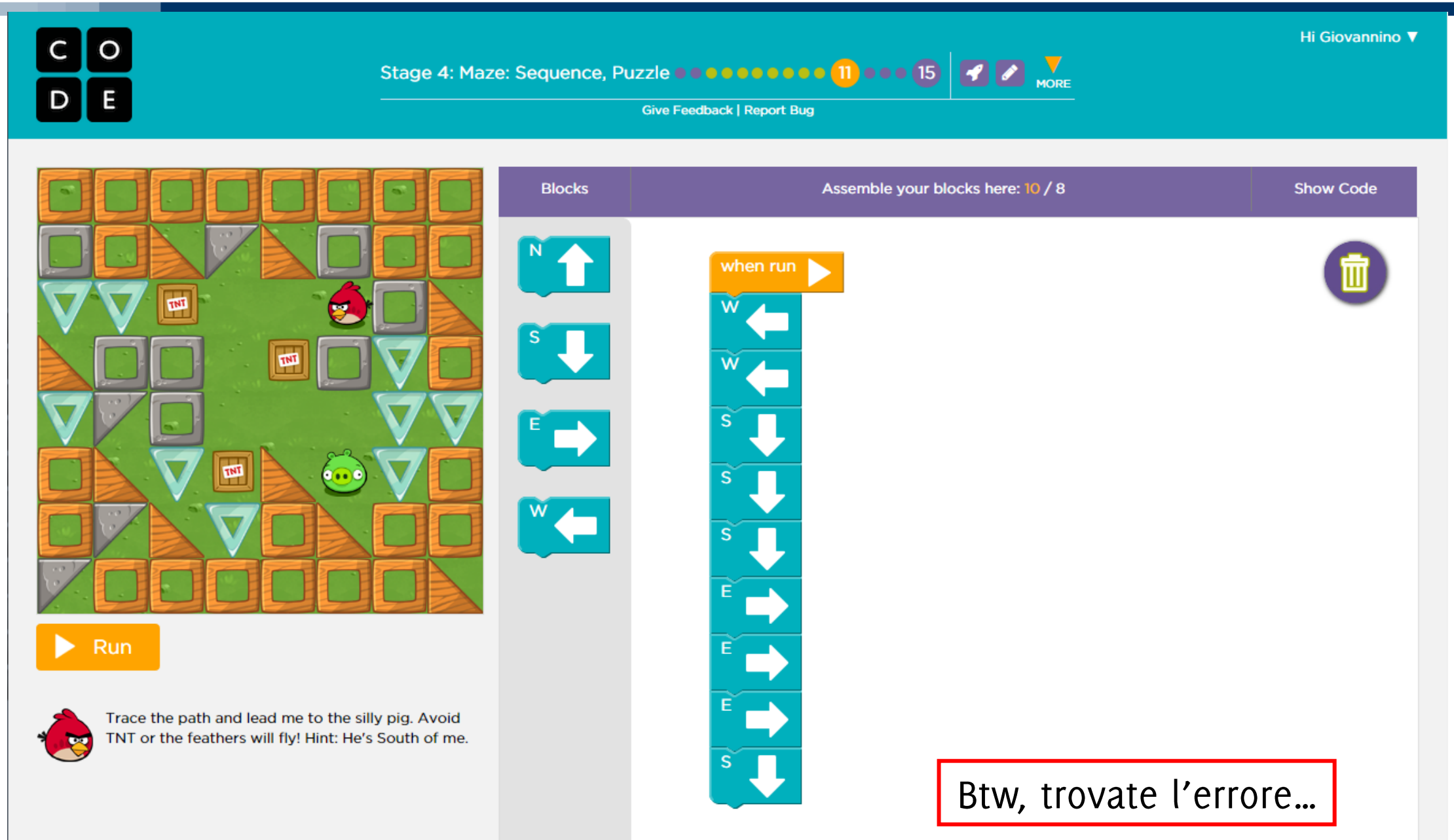
La sequenzialità per i bimbi su code.org



La sequenzialità per i bimbi su code.org



La sequenzialità per i bimbi su code.org



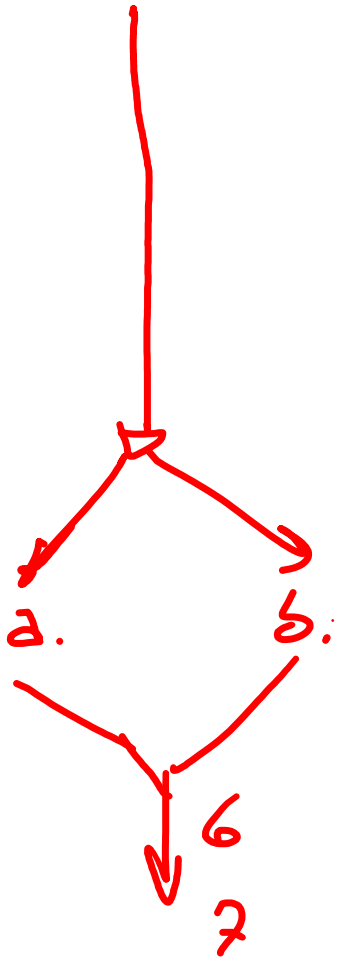
Costrutto Condizionale

Esempio: algoritmo per andare in Università

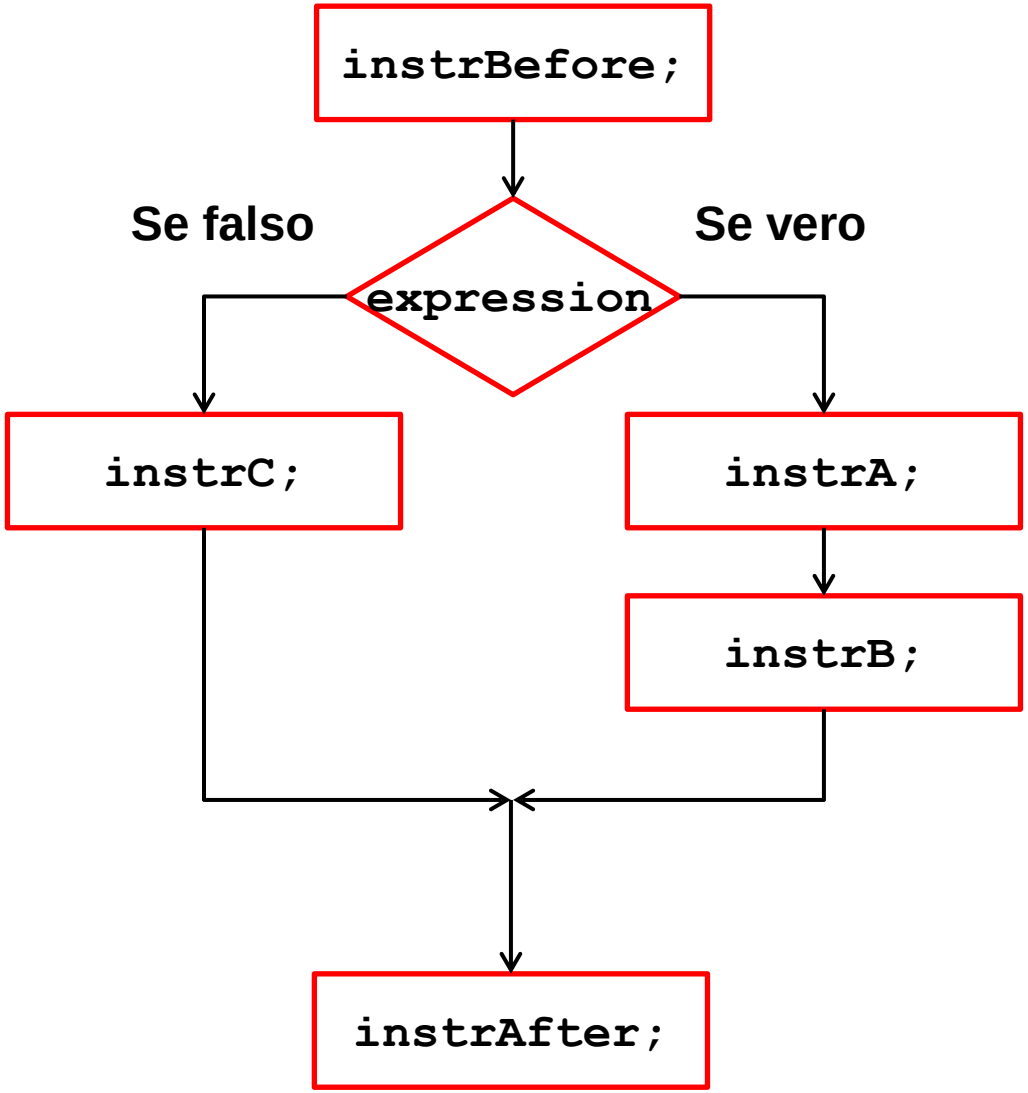
1. Mi alzo quando suona la sveglia
2. Mi dirigo verso la cucina
3. Mangio e bevo un caffè
4. Mi lavo e mi vesto
5. **Se** devo mangiare fuori
   a. prendo il pranzo

   Altrimenti

   b. prendo uno snack per metà mattina
6. Esco
7. Corro

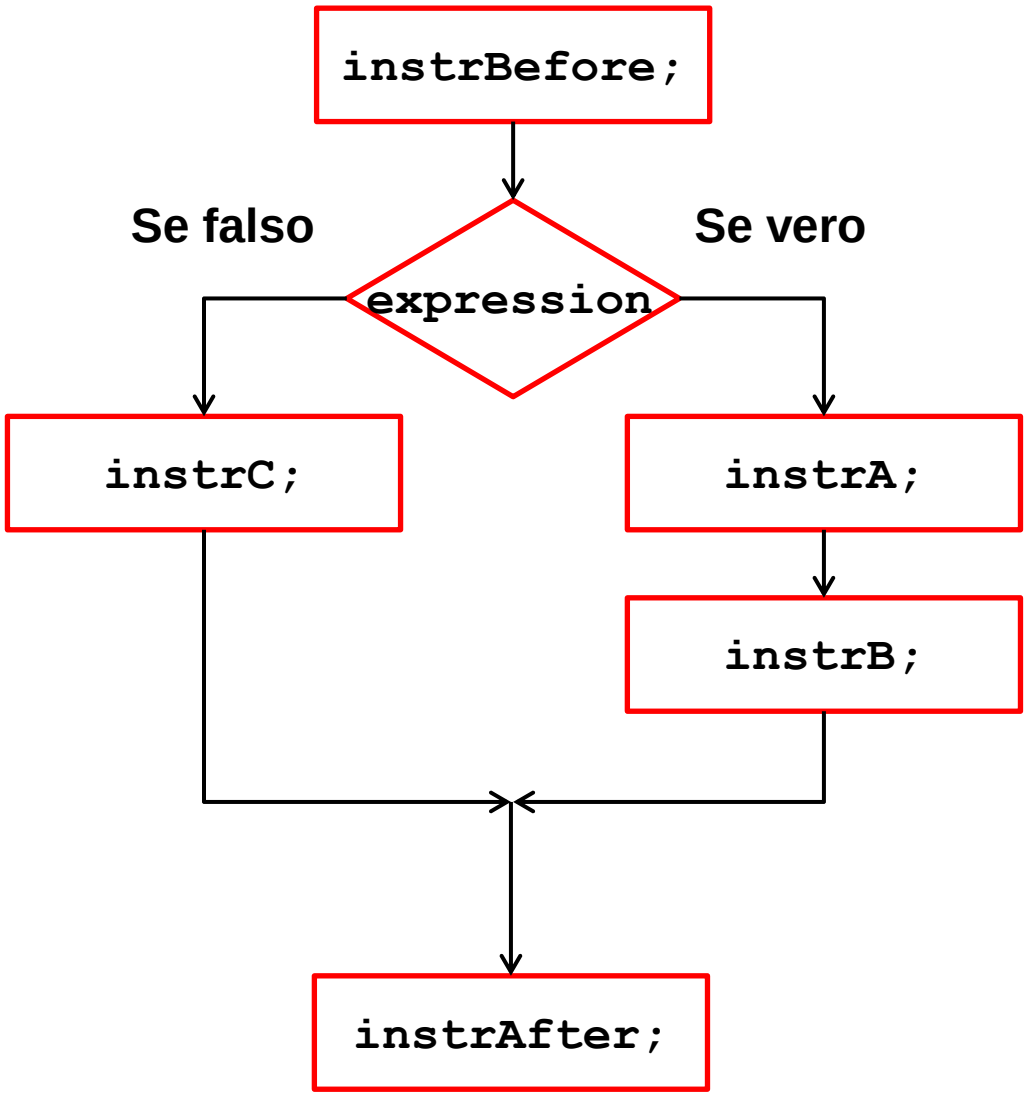


Costrutto Condizionale:



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

Costrutto Condizionale:

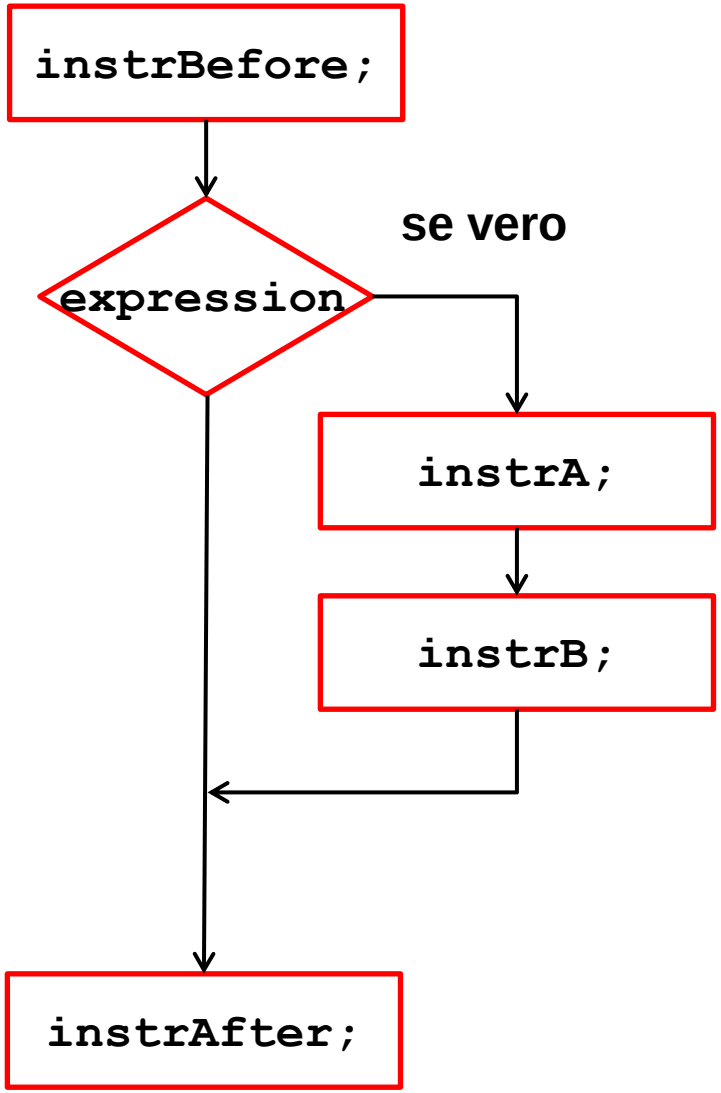


Dopo aver eseguito **`instrBefore`** si valuta **`expression`**

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

Altrimenti eseguo **`instrC;`**

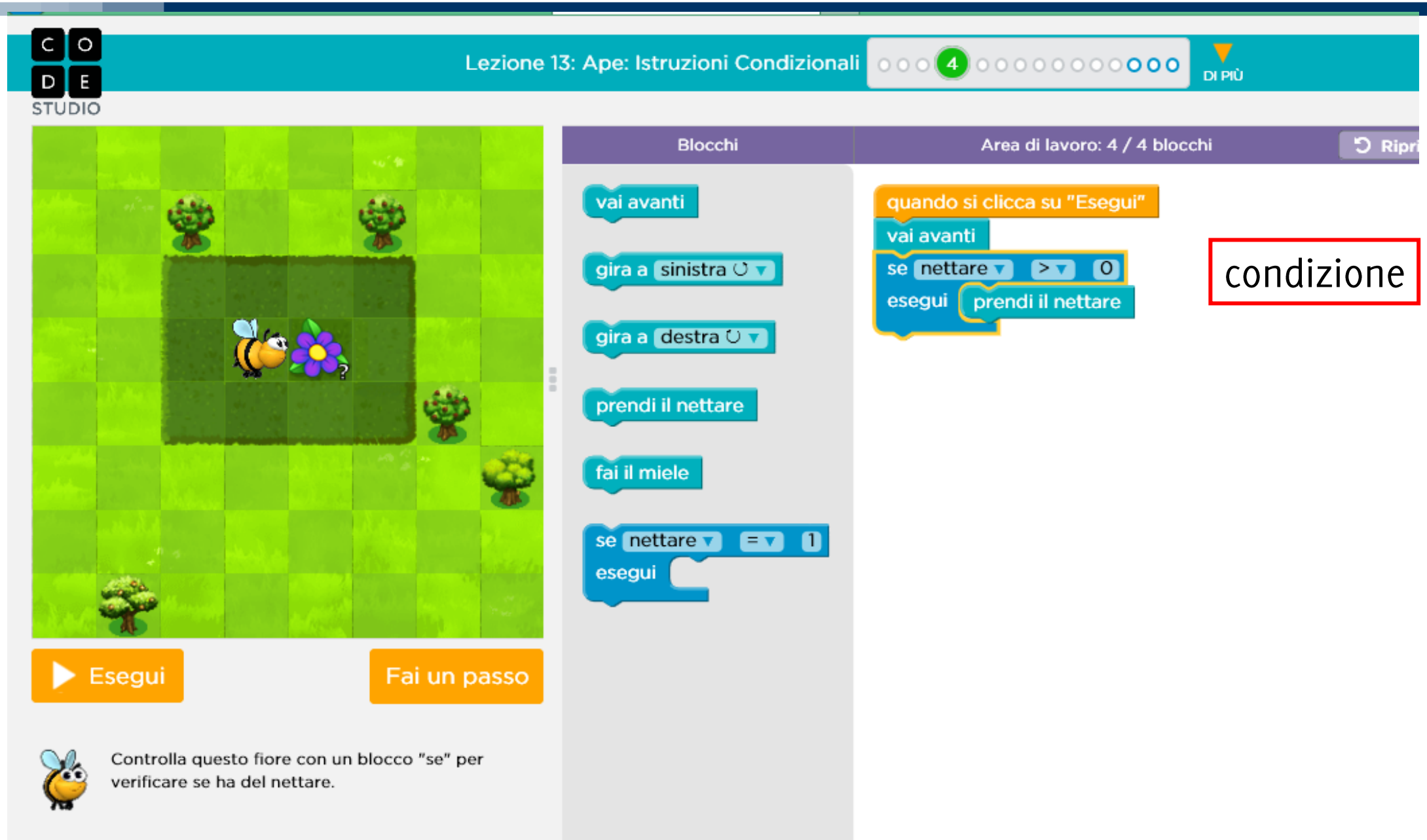
Costrutto Condizionale:



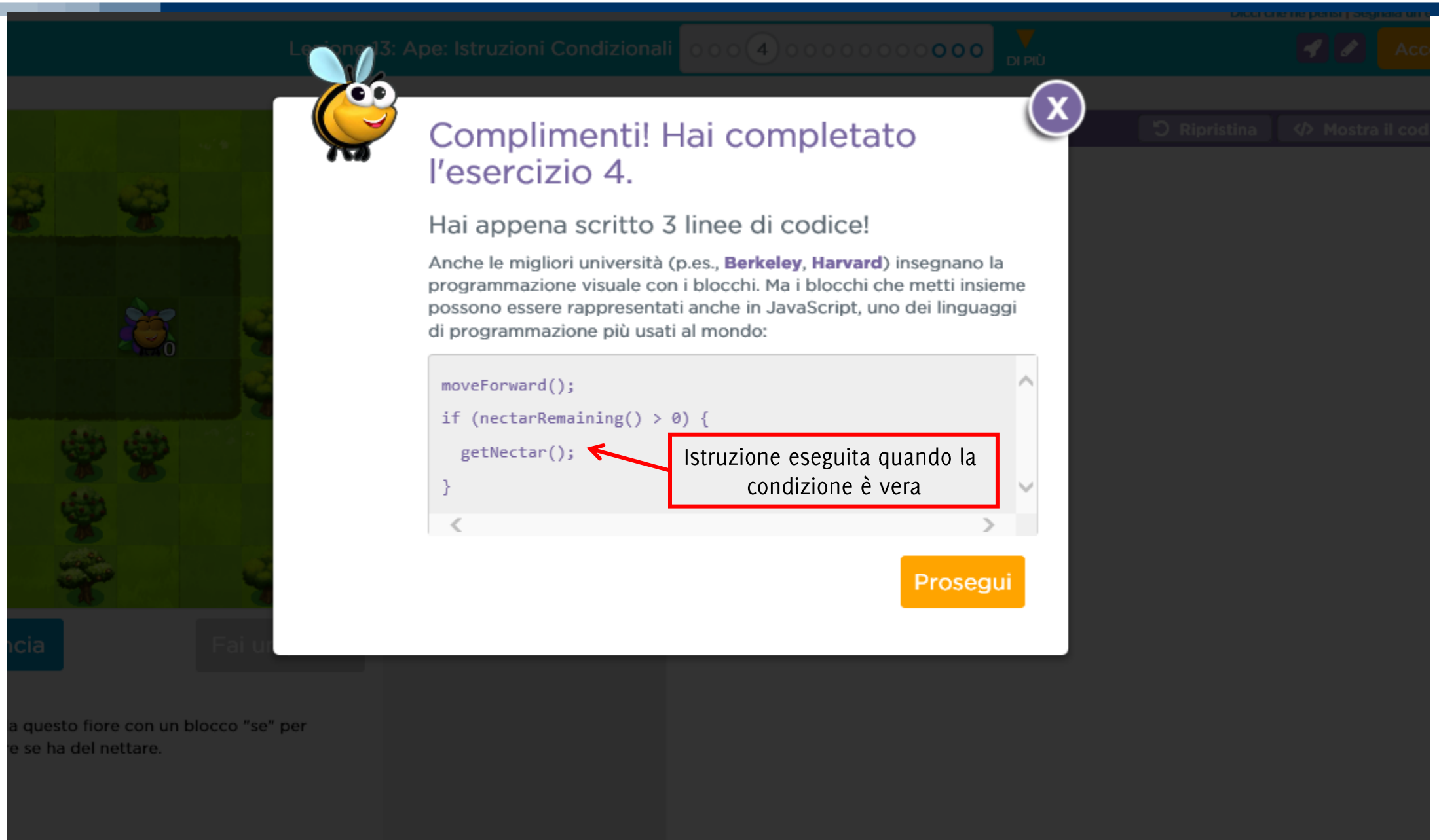
Non è necessario prevedere azioni specifiche nel caso in cui **`expression`** fosse falsa

if (condizione)
→ ISTR se vera
[else] opzionale

Il costrutto condizionale su code.org



Il costrutto condizionale su code.org



Esempio: algoritmo per andare in Università

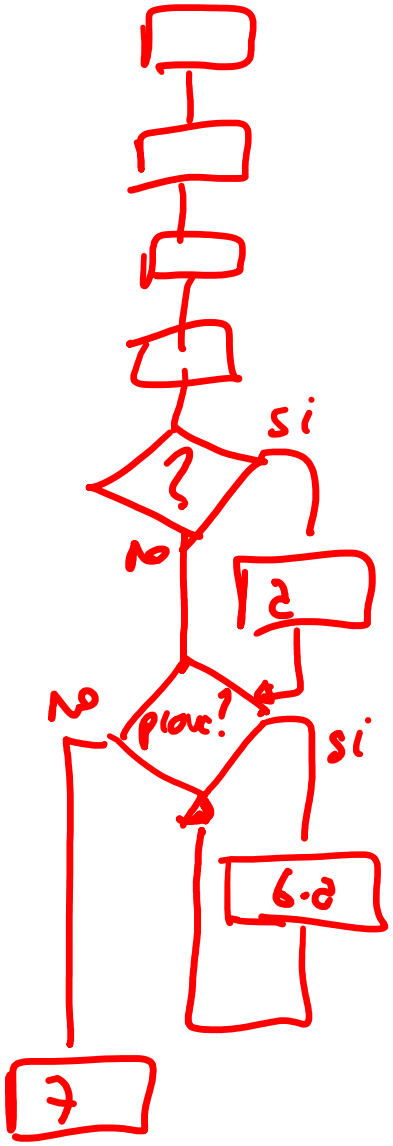
1. Mi alzo quando suona la sveglia
2. Mi dirigo verso la cucina
3. Mangio e bevo un caffè
4. Mi lavo e mi vesto
5. **Se** piove
    a. Prendo l'ombrello
6. Esco
7. Corro

Costrutto Iterativo

Esempio: algoritmo per andare in Università

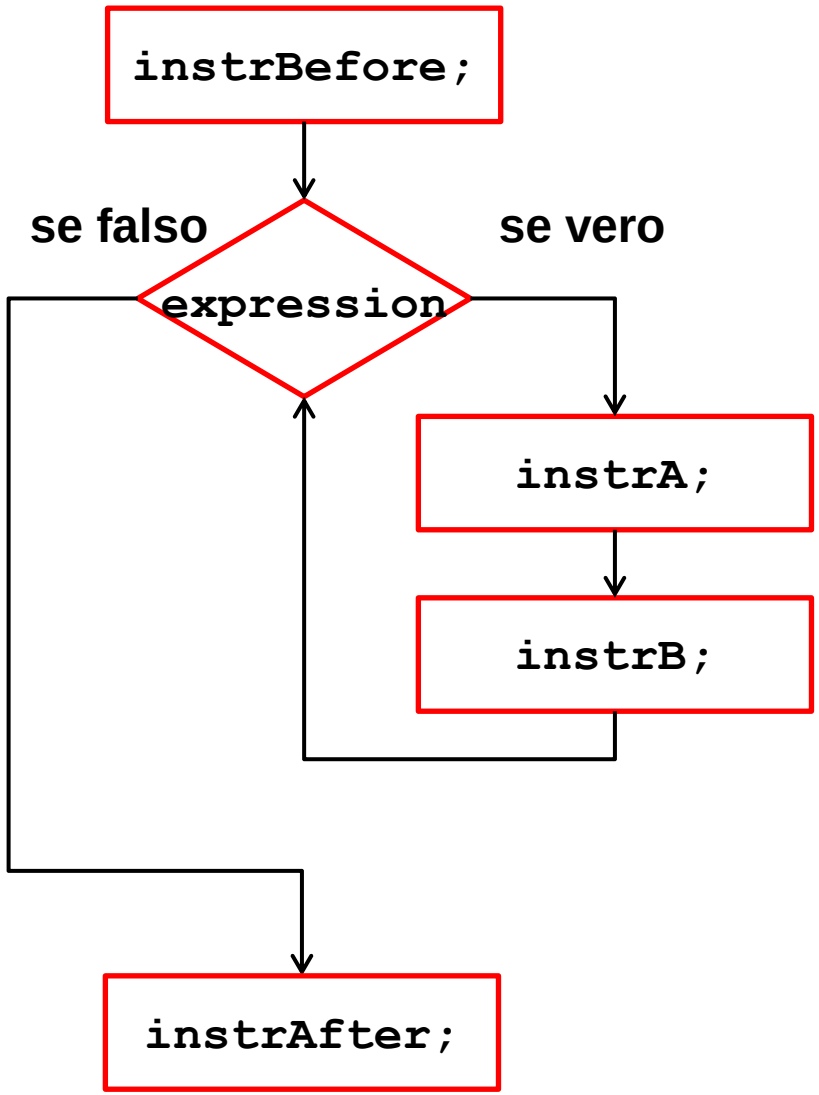
1. Mi alzo quando suona la sveglia
2. Mi dirigo verso la cucina
3. Mangio e bevo un caffè
4. Mi lavo e mi vesto
5. **Se** c'è il laboratorio di informatica
    a. Prendo il laptop
6. **Ripeti**: piove ancora?
    a. Aspetta a casa 5 minuti
7. Vado in stazione
8. Sali sul treno
9. Arrivi a lezione, wow

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

Costrutto Iterativo



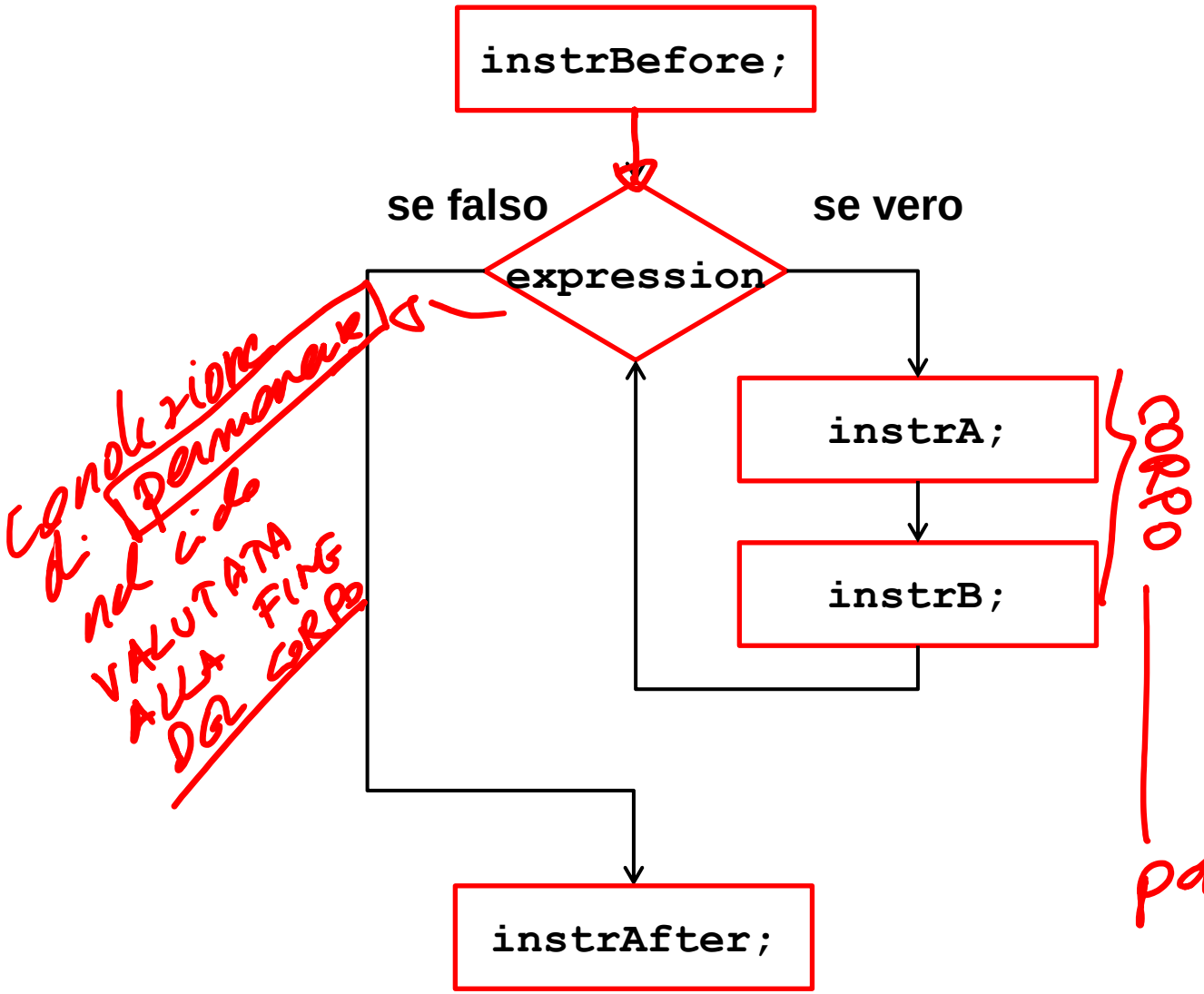
Se **`expression`** è vera eseguo il ramo di istruzioni contenente **`instrA;`** e **`instrB;`** (corpo del ciclo)

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

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

`Expression` rappresenta quindi la condizione di permanenza nel ciclo

Costrutto Iterativo



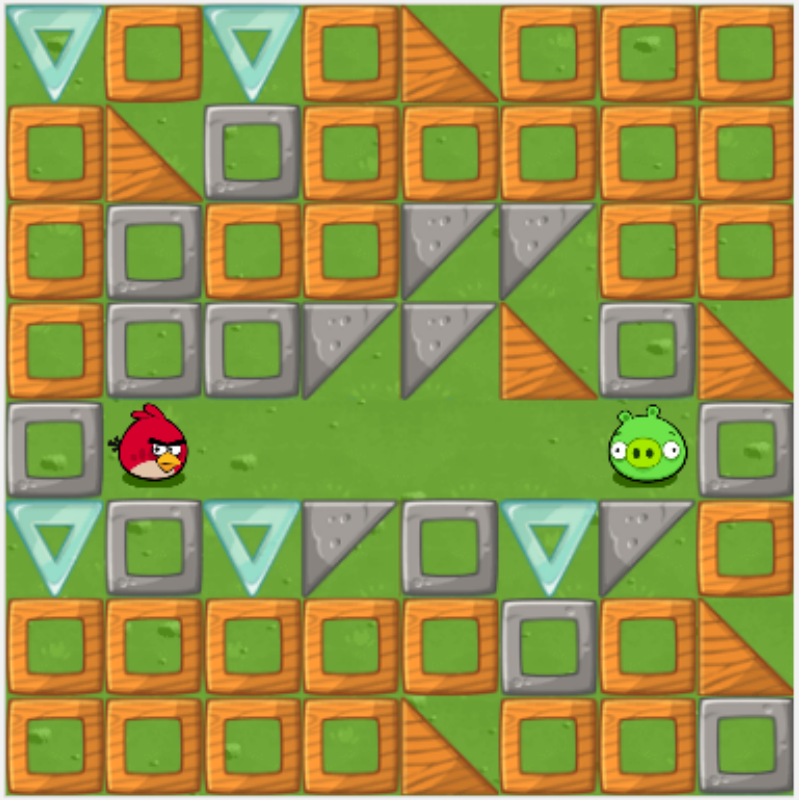
condizione di permanenza nel ciclo VALUTATA ALLA FINE DEL CORPO

CORPO

potrebbe non venir mai eseguito

Il costrutto iterativo su code.org

STUDIO



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

Blocchi

vai avanti

gira a sinistra

gira a destra

ripeti fino a che

esegui

Area di lavoro: 3 / 3 blocchi

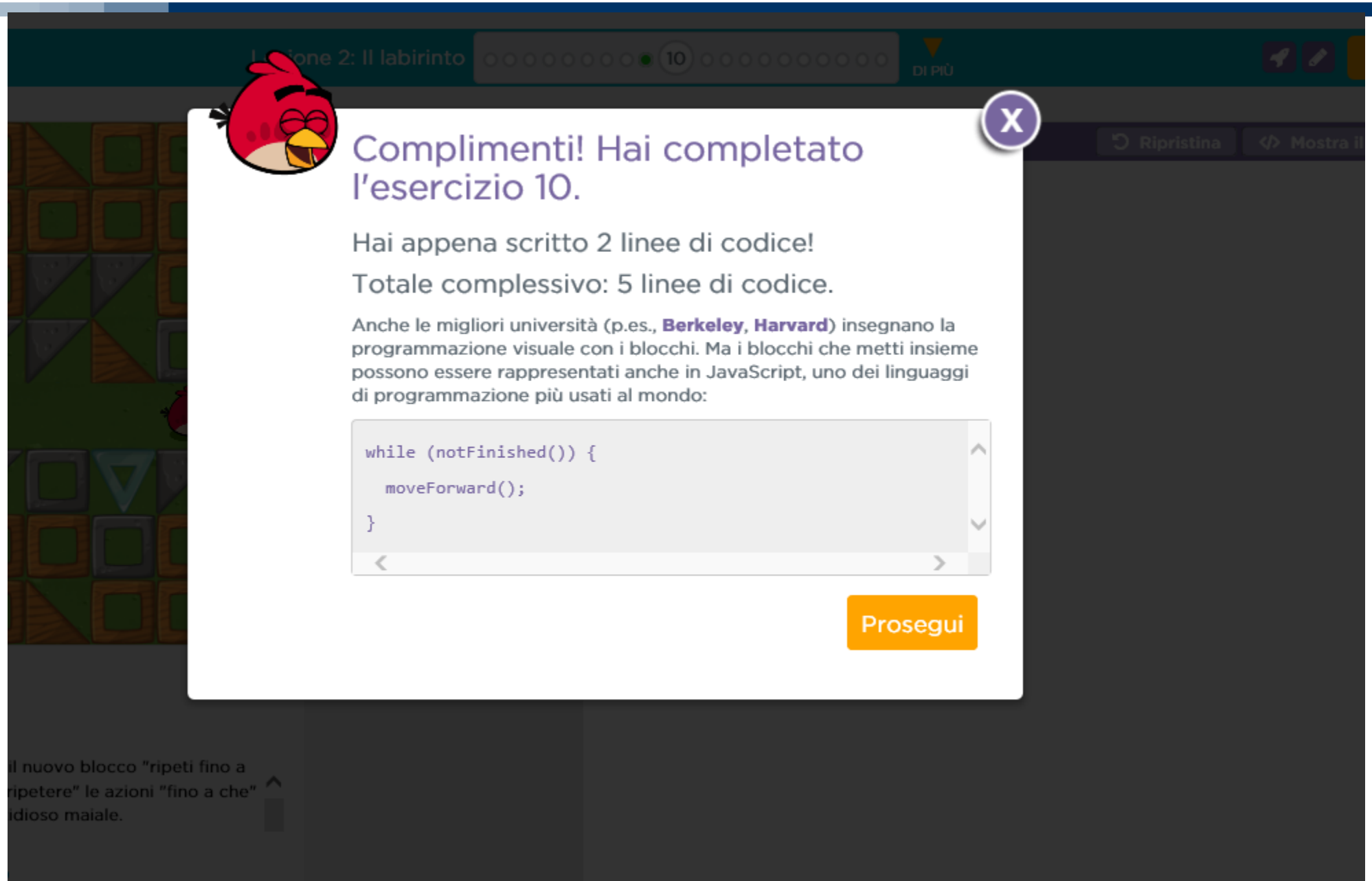
Ripristina

quando si clicca su "Esegui"

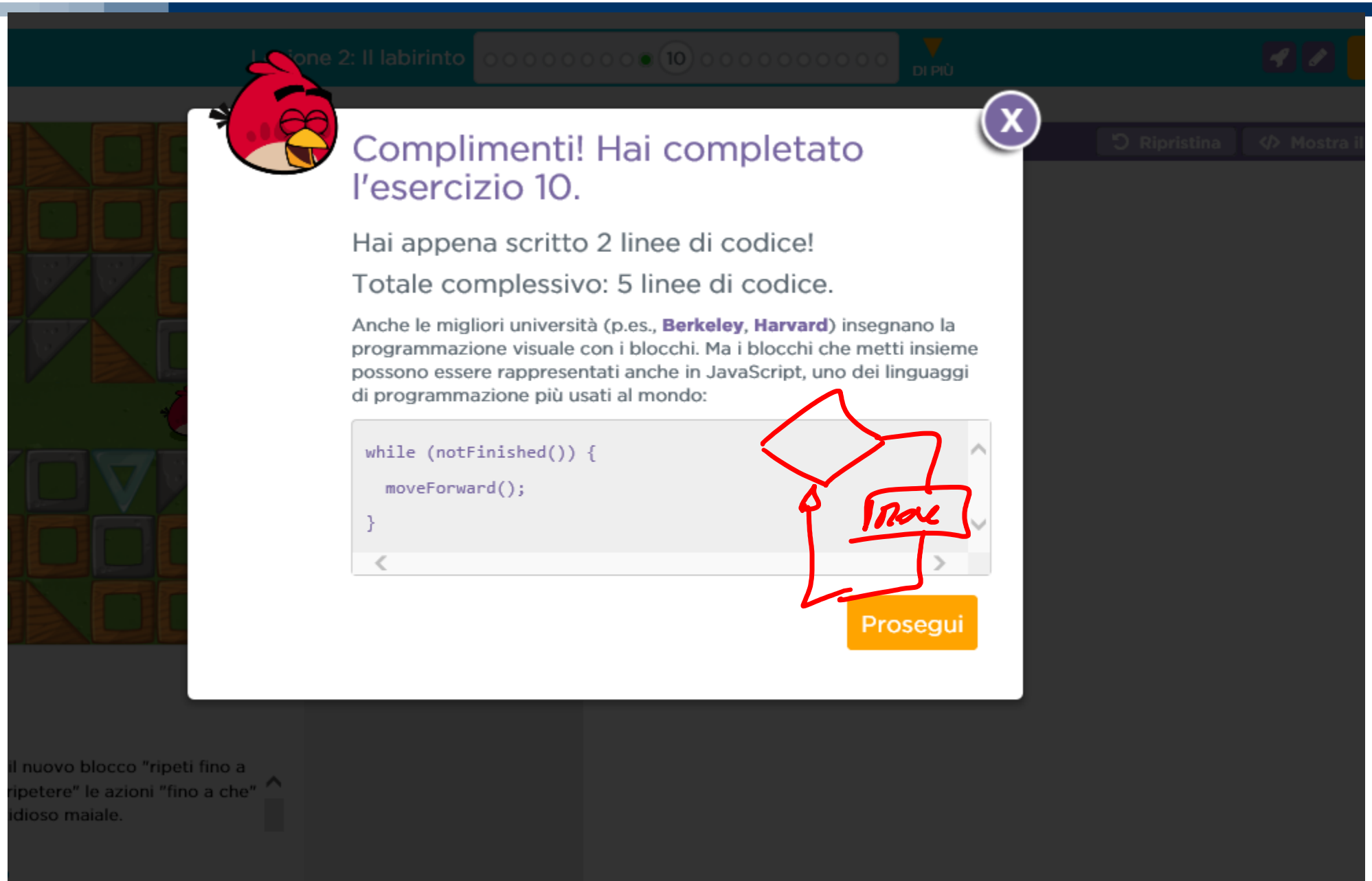
ripeti fino a che

esegui vai avanti

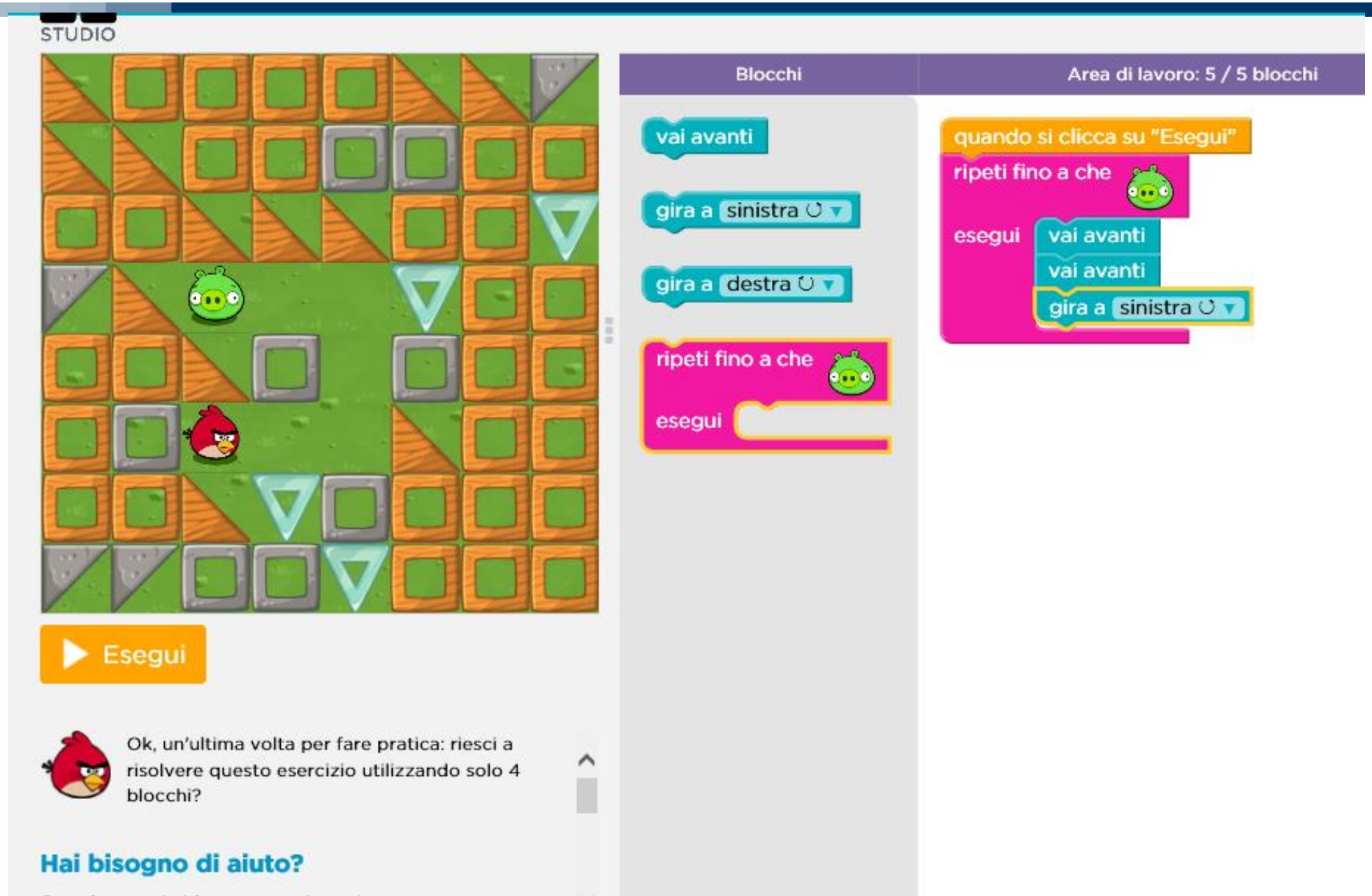
Il costrutto iterativo su code.org



Il costrutto iterativo su code.org

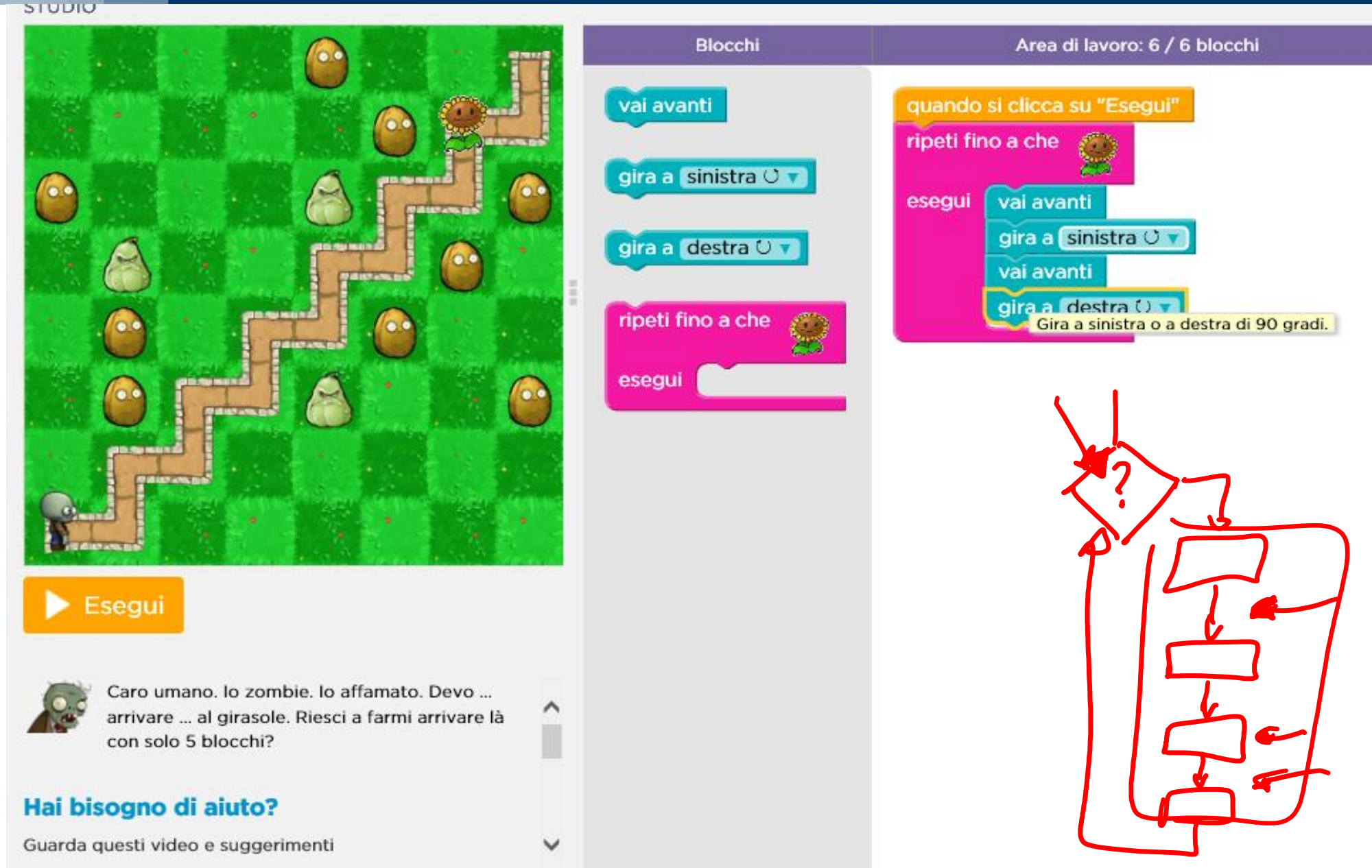


Il costrutto iterativo su code.org



Il costrutto iterativo su code.org

Il costrutto iterativo su code.org



Il costrutto iterativo su code.org

Il costrutto iterativo e condizionale su code.org

STUDIO

Esegui

Qui il nuovo blocco "se" viene usato per farmi 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.

Hai bisogno di aiuto?

Blocchi

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

Area di lavoro: 4 / 5 blocchi

- 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 iterativo e condizionale su code.org

STUDIO

Esegui

Qui il nuovo blocco "se" viene usato per farmi 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.

Hai bisogno di aiuto?

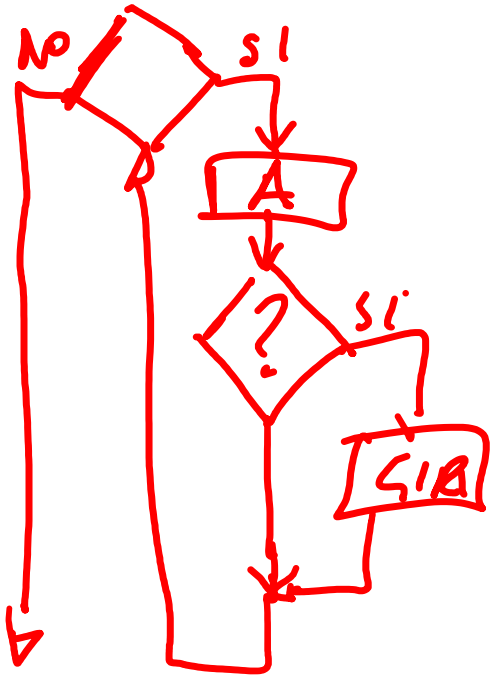
Blocchi

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

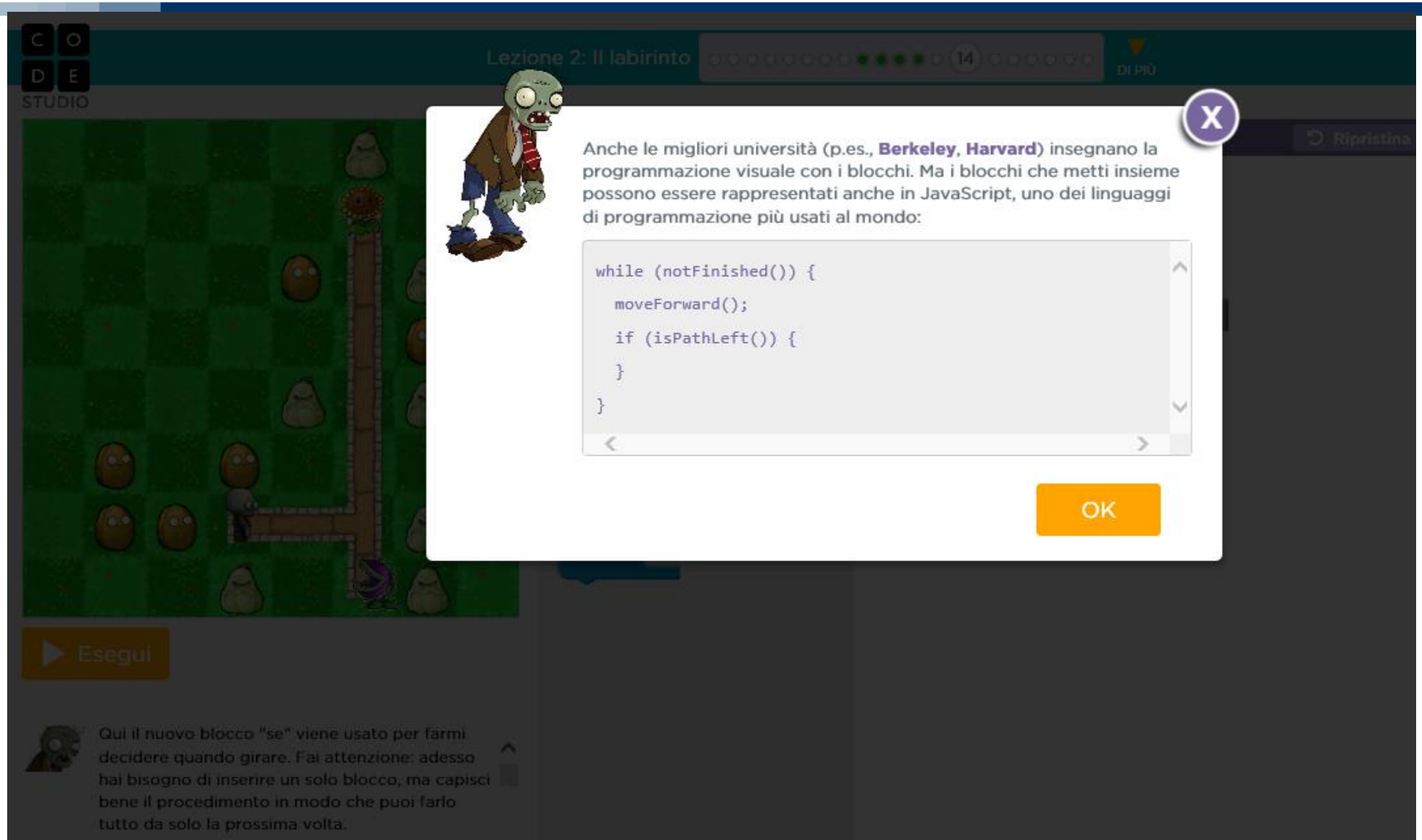
Area di lavoro: 4 / 5 blocchi

- 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



Esempi di Algoritmi

Il Pallottoliere

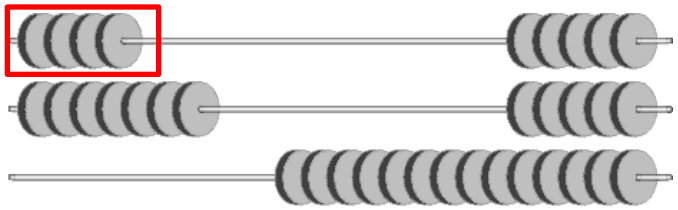
Supponiamo di avere un bambino che sa contare ma non sa fare operazioni aritmetiche.

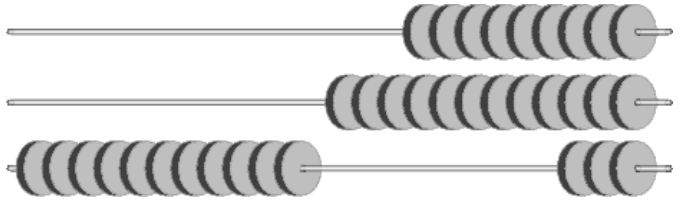
Scriviamo le istruzioni per utilizzare il pallottoliere (utilizzando un corstrutto iterativo)

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

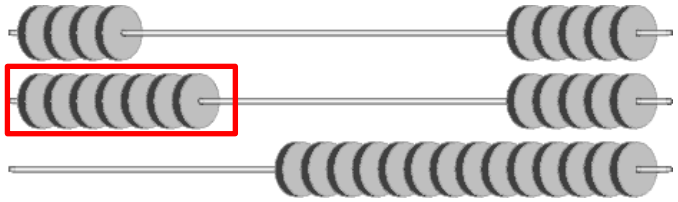


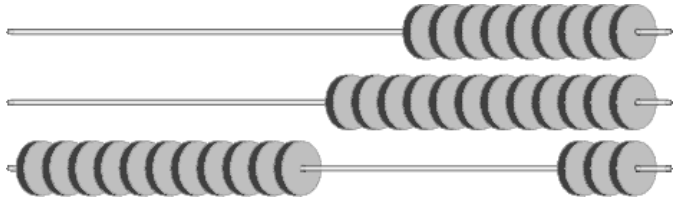


Algoritmo per sommare col pallottoliere

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

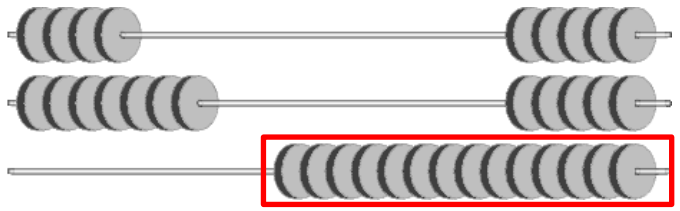


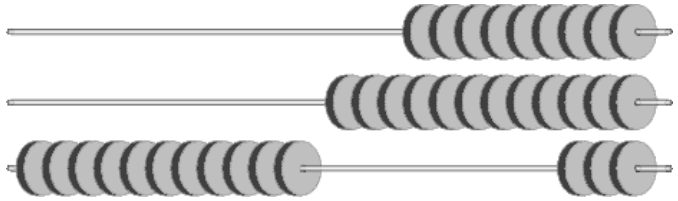


Algoritmo per sommare col pallottoliere

Supponiamo che:

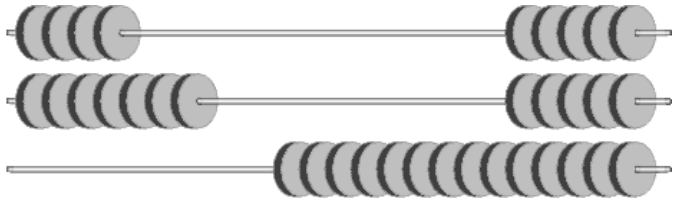
- Primo addendo sia riportato sulla prima riga a sinistra
- Secondo addendo sia sulla seconda riga a sinistra
- Risultato deve apparire nella terza riga (e che all'inizio questa abbia tutte le palline a destra).
- Operiamo con numeri «piccoli», non c'è bisogno del riporto

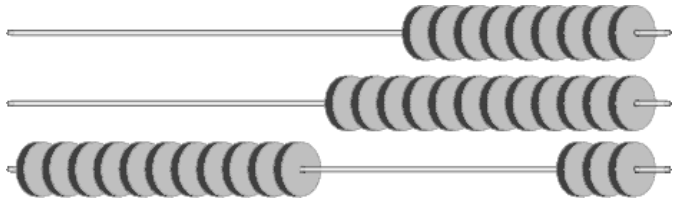




Algoritmo del pallottoliere

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.





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

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

AUTORE / I:
GHEZZI CARLO,
JAZAYERI MEHDI.

TITOLO:
PROGRAMMING LANGUAGE CONCEPTS.
1981.

SCAFFALE 35
POSIZIONE 21

Algoritmo: Ricerca

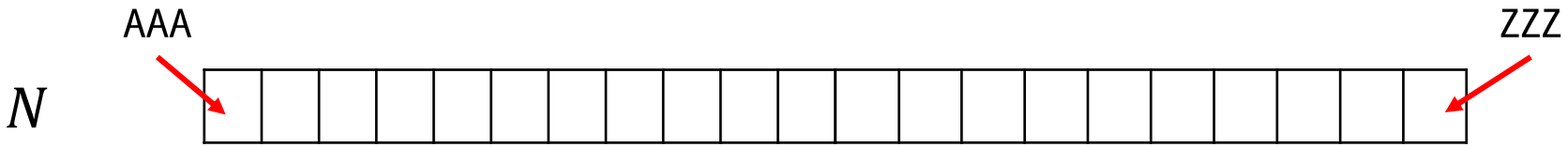
Non tutti gli esecutori sono in grado di «cercare», e comunque la ricerca può essere fatta in diversi modi

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

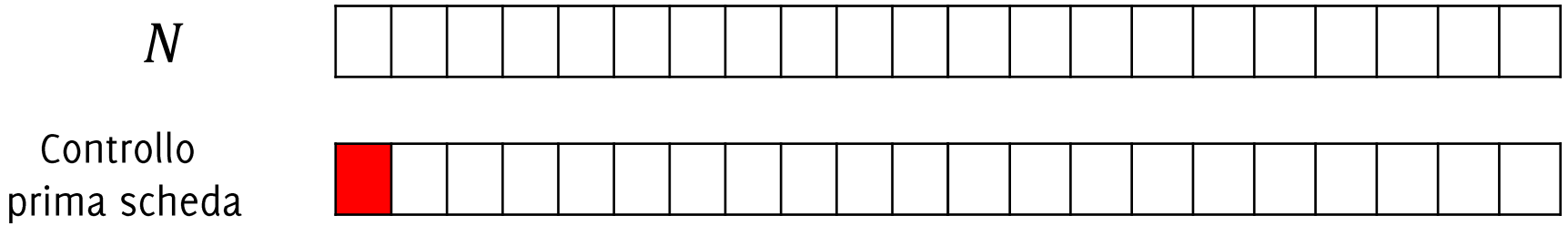
Algoritmo: Ricerca

Assumendo che l'archivio abbia N schede:



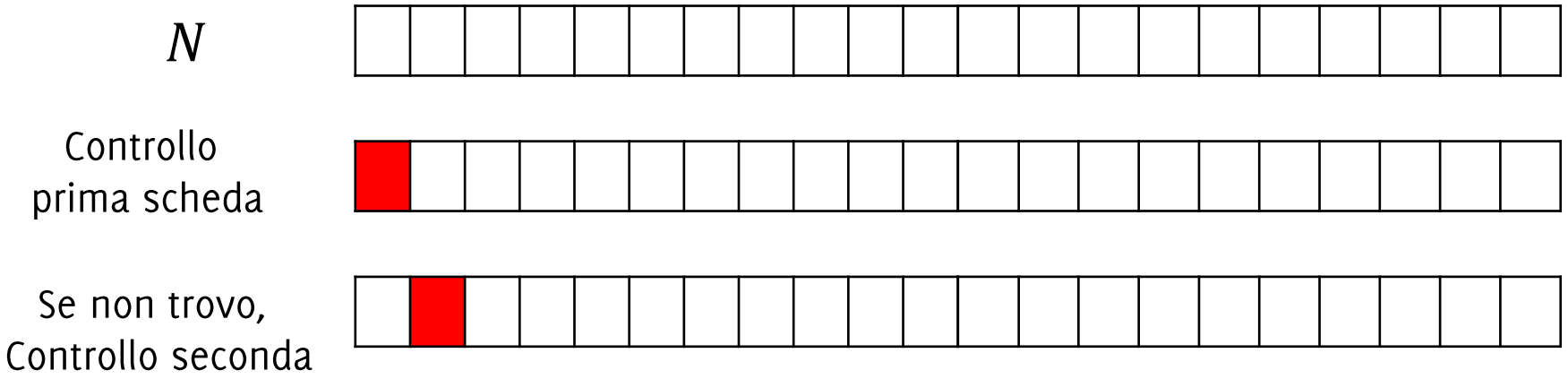
Algoritmo: Ricerca

Assumendo che l'archivio abbia N schede:



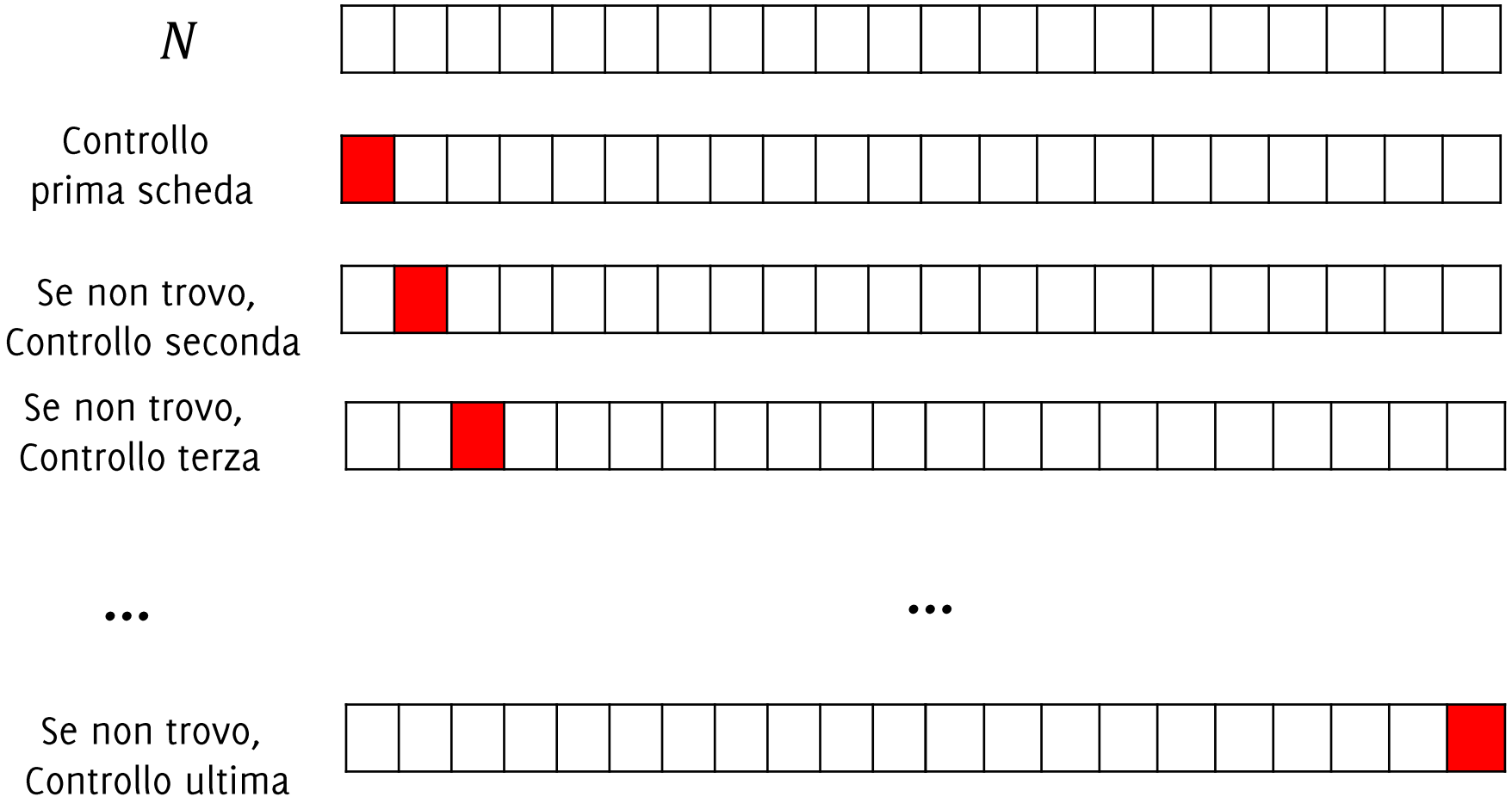
Algoritmo: Ricerca

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

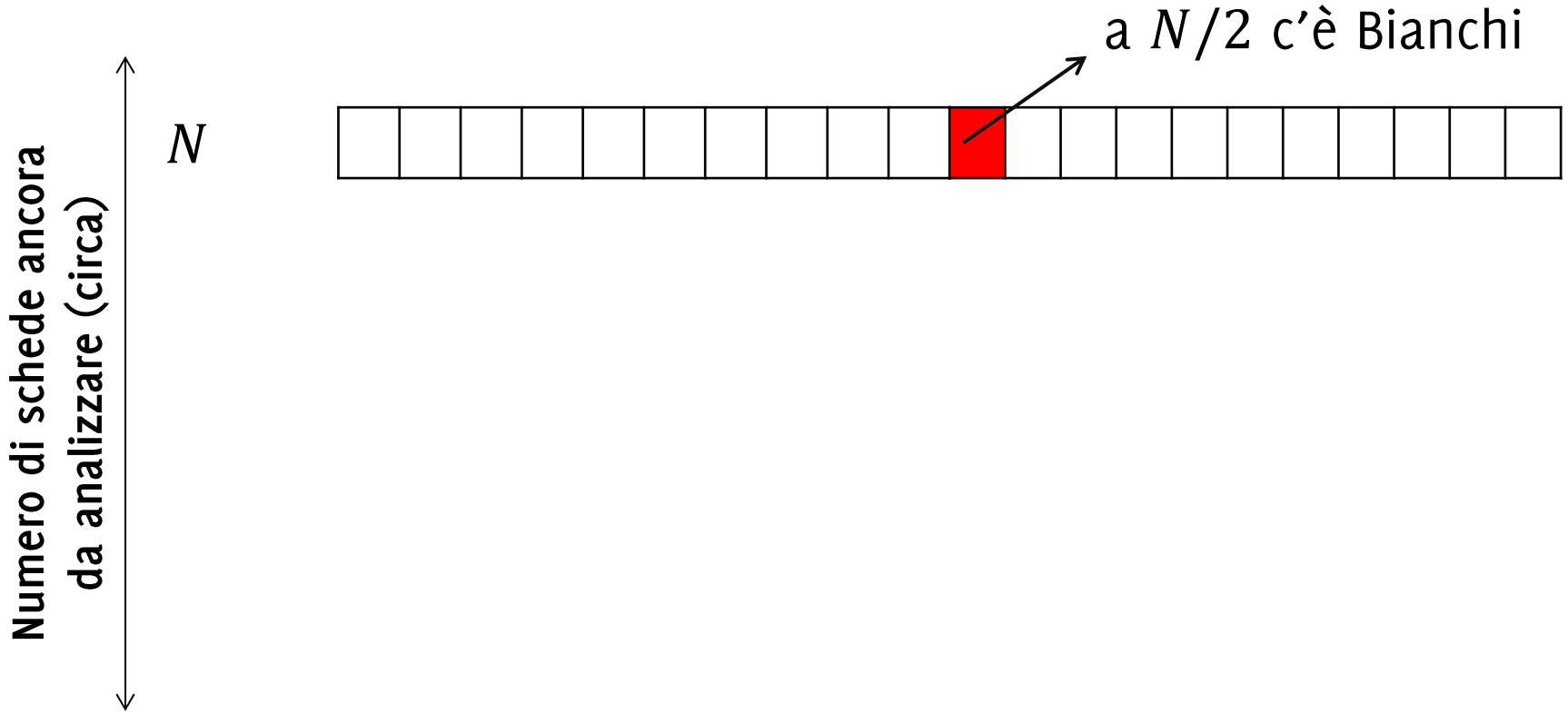
altrimenti,

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).

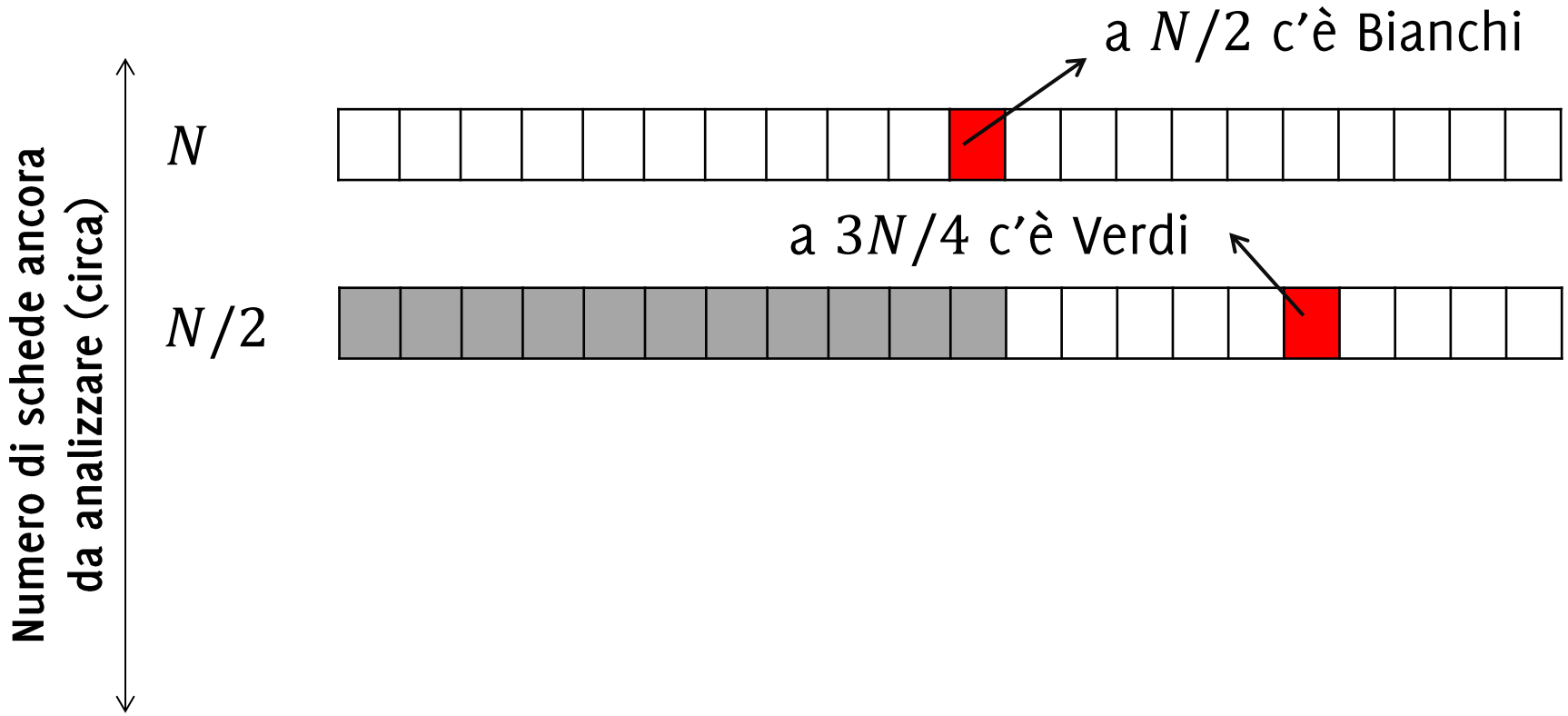
Algoritmo: Ricerca tra elementi ordinati

es: Ricerca dell'autore ROSSI



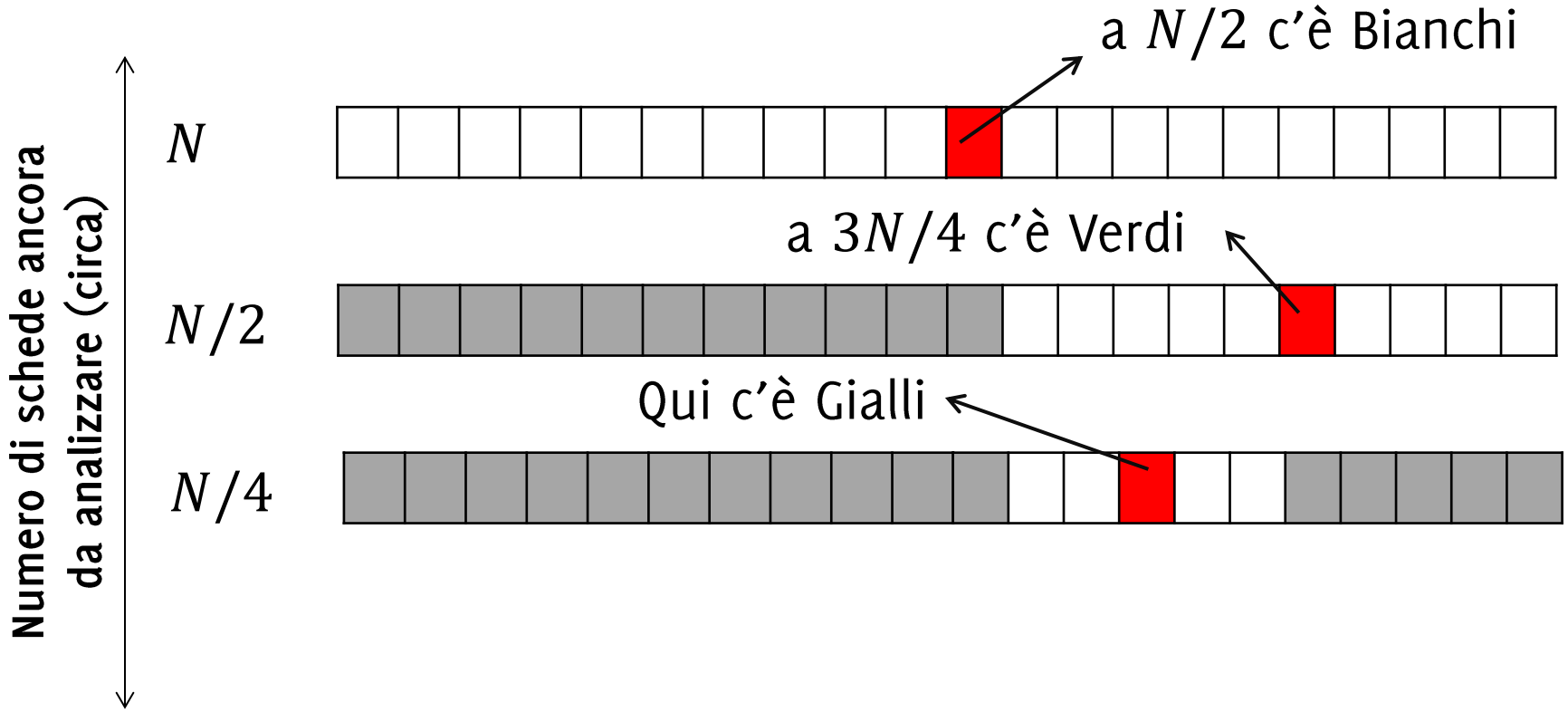
Algoritmo: Ricerca tra elementi ordinati

es: Ricerca dell'autore ROSSI



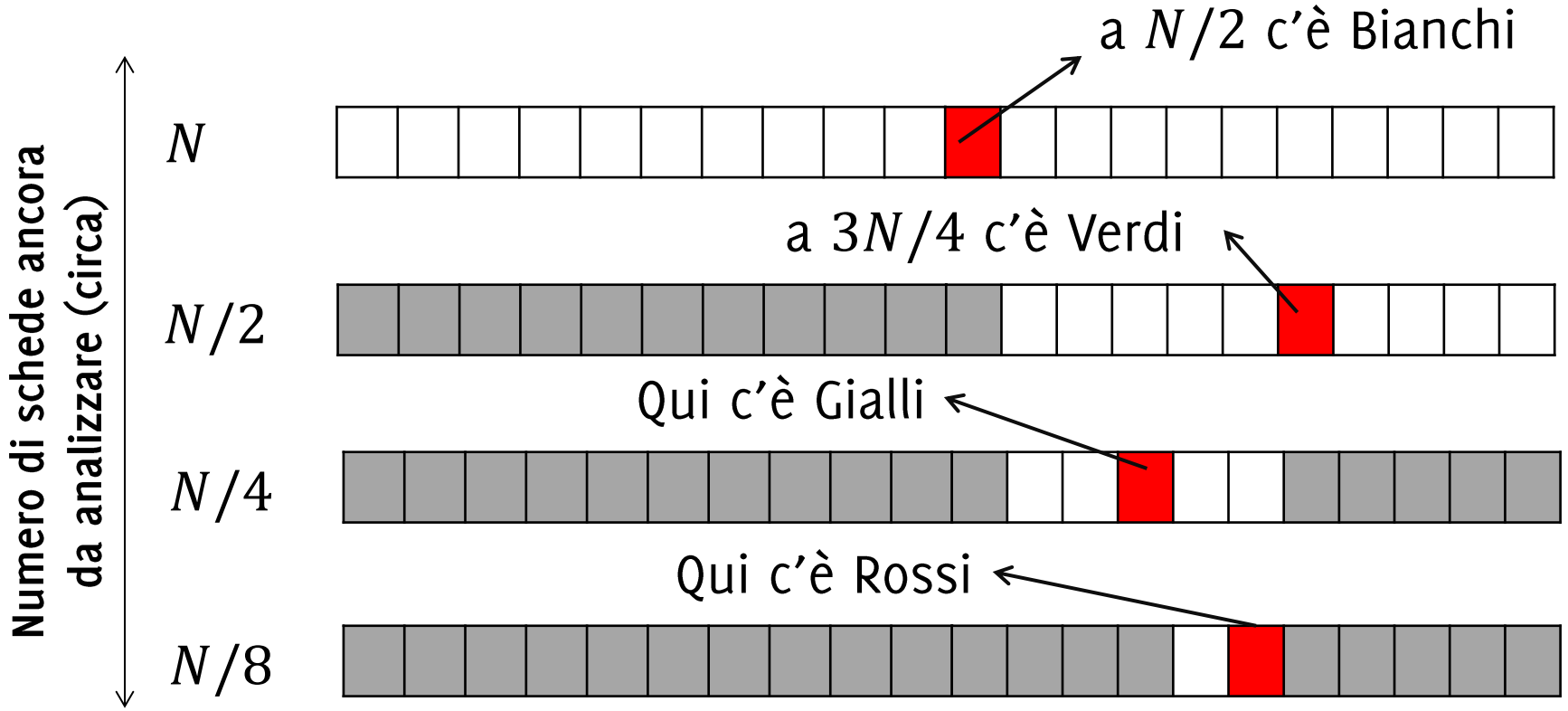
Algoritmo: Ricerca tra elementi ordinati

es: Ricerca dell'autore ROSSI



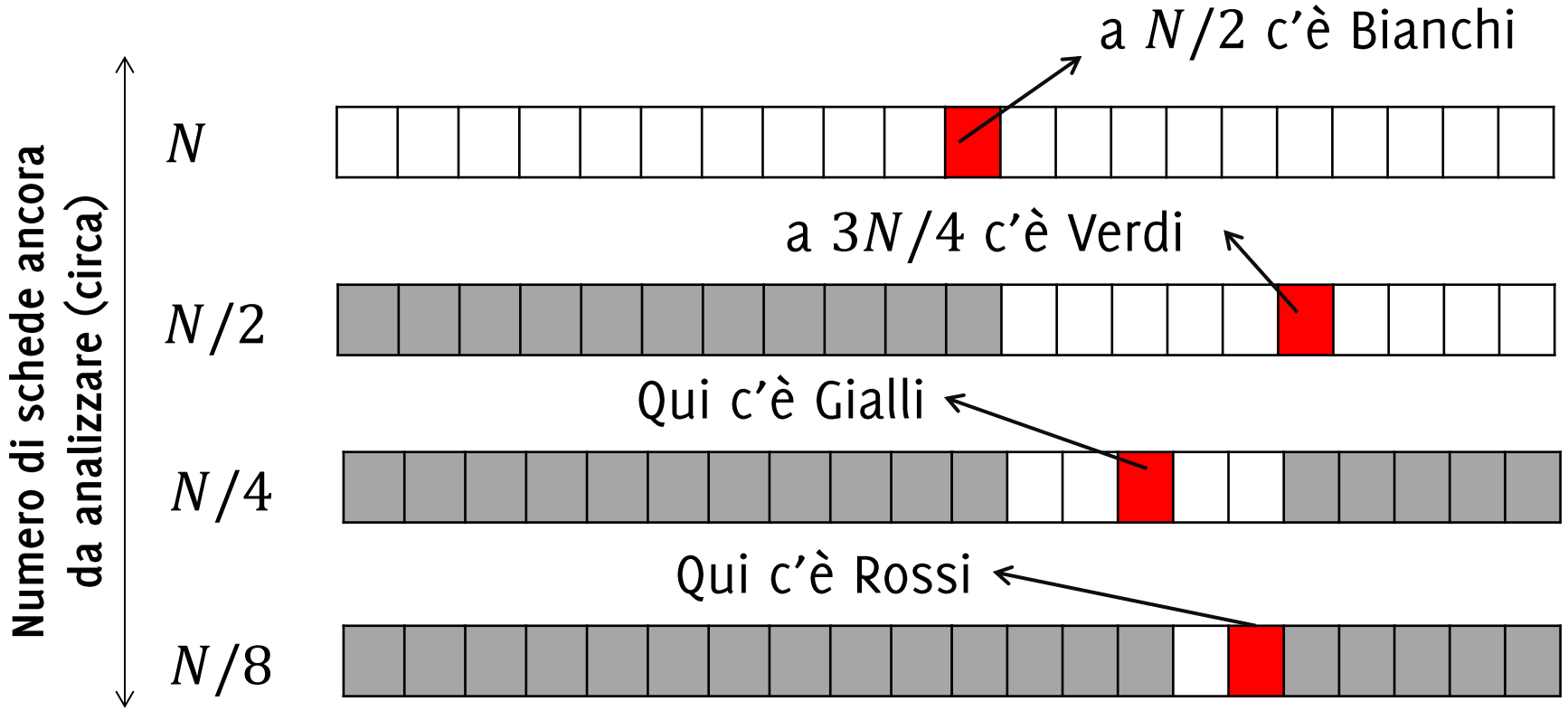
Algoritmo: Ricerca tra elementi ordinati

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

I Diagrammi di Flusso

Linguaggi per esprimere gli algoritmi

Abbiamo due tipi di linguaggi

Semi-formali: specifiche iniziali, ancora **intelligibili solo all'essere umano**

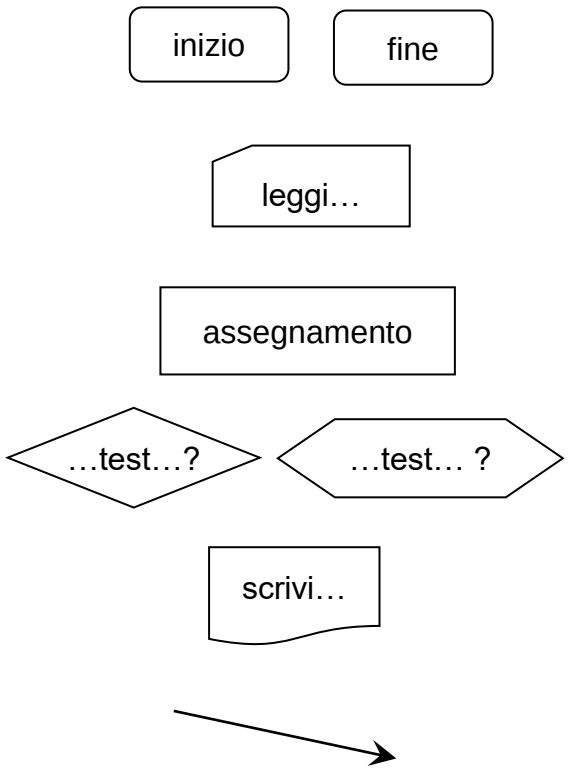
- **Pseudo-codice**
- **Diagrammi di flusso**

Formali: **intelligibili anche alla macchina,** diventano programmi che possono essere eseguiti

- **linguaggi di programmazione**

Diagrammi di Flusso

Detti anche flow chart o schemi a blocchi, sono composti da:



- Blocchi di inizio/fine dell'esecuzione
- Blocco di input (inserimento dati)
- Blocco esecutivo (istruzioni)
- Blocco condizionale
- Blocco di output (stampa dati)
- Flusso di controllo delle operazioni

Le variabili

Inseriamo **le variabili** nei diagrammi di flusso

- Concettualmente identiche ai «foglietti» degli esempi precedenti
- Corrispondono ad uno spazio cui ha accesso l'esecutore, dove poter registrar valori numerici.
- E' possibile leggere, scrivere e sovrascrivere un valore ad una variabile
- Hanno un nome che le identifica
- Hanno anche un tipo, che definisce l'insieme di valori e le operazioni ammissibili

L'istruzione di assegnamento

L'istruzione

$$X \leftarrow X + Y$$

Corrisponde alle seguenti istruzioni

1. Si calcola il valore dell'espressione a destra della freccia (in questo caso $X + Y$)
2. Si scrive il risultato nella variabile a sinistra della freccia (in questo caso X)

Algoritmo 1:

Scrivere l'algoritmo che:

1. Richiede all'utente un numero N
2. calcola la somma dei primi N numeri naturali
3. Stampa il risultato

P.S. **non usare la formula**

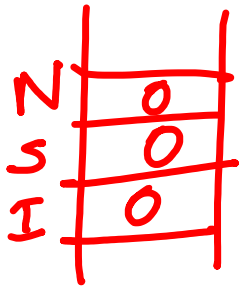
$$\sum_{0}^{N} x = \frac{N}{2}(N+1)$$

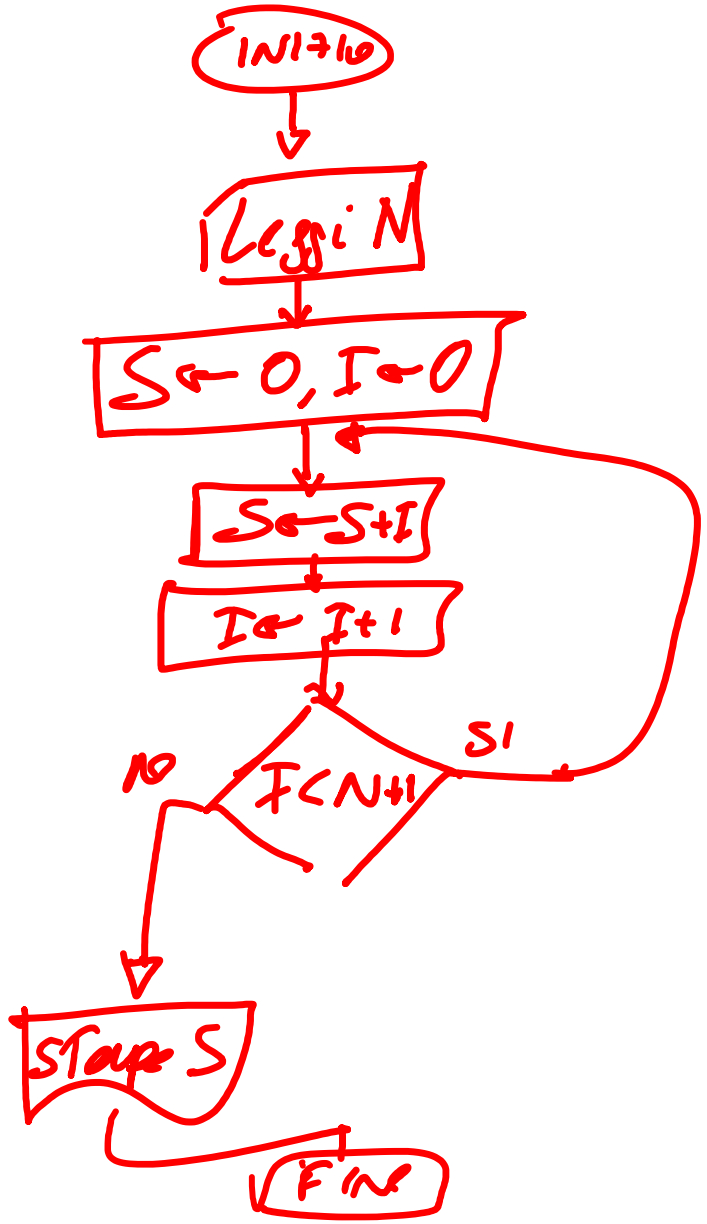
ma progettare una soluzione iterativa

Algoritmo 1:

Scrivere l'algoritmo che:

1. Richiede all'utente un numero $N \in \mathbb{N}$
2. calcola la somma dei primi N numeri naturali
3. Stampa il risultato





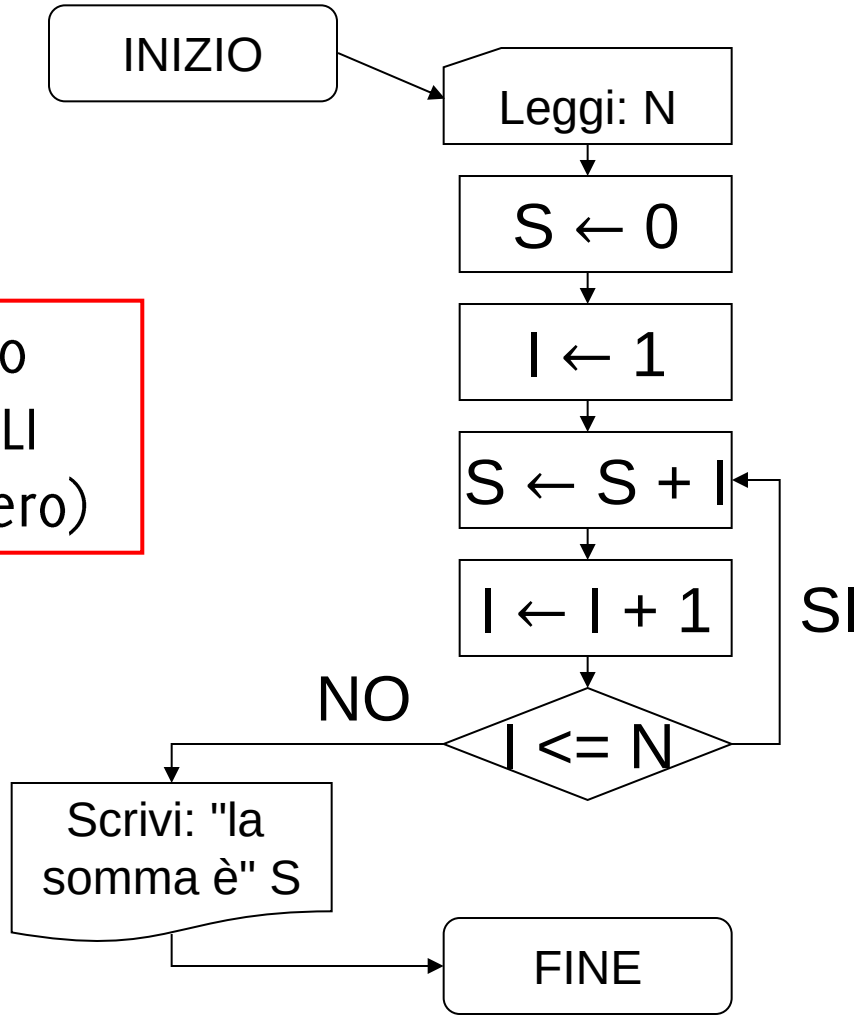
P.S. **non usare la formula**

$$\sum_{0}^{N} x = \frac{N}{2}(N+1)$$

ma progettare una soluzione iterativa

Somma dei primi N numeri naturali

I simboli S, I e N sono definiti come VARIABILI NUMERICHE (di tipo intero)

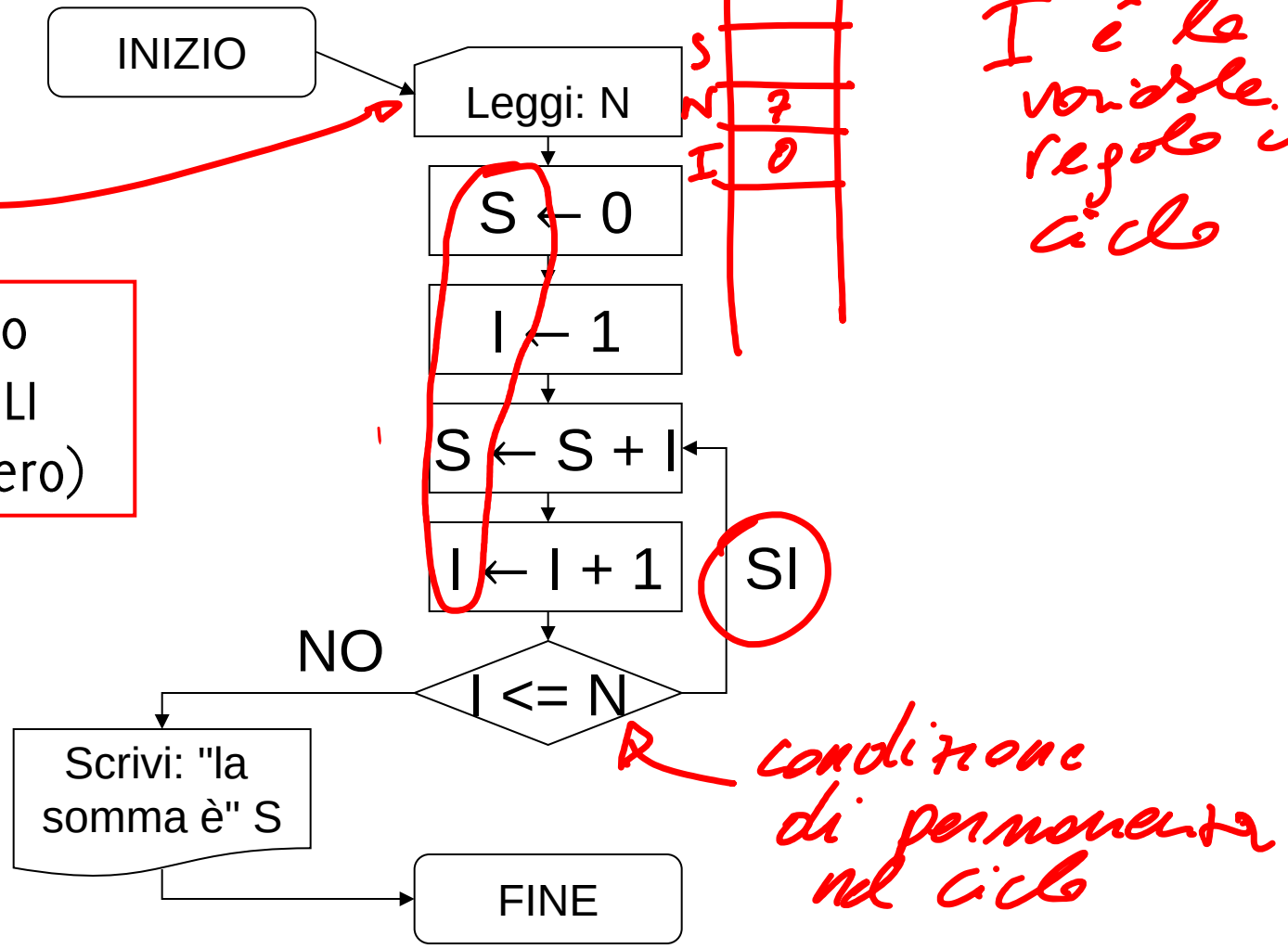


Somma dei primi N numeri naturali

Si assume N > 0

la N non viene modificata dopo

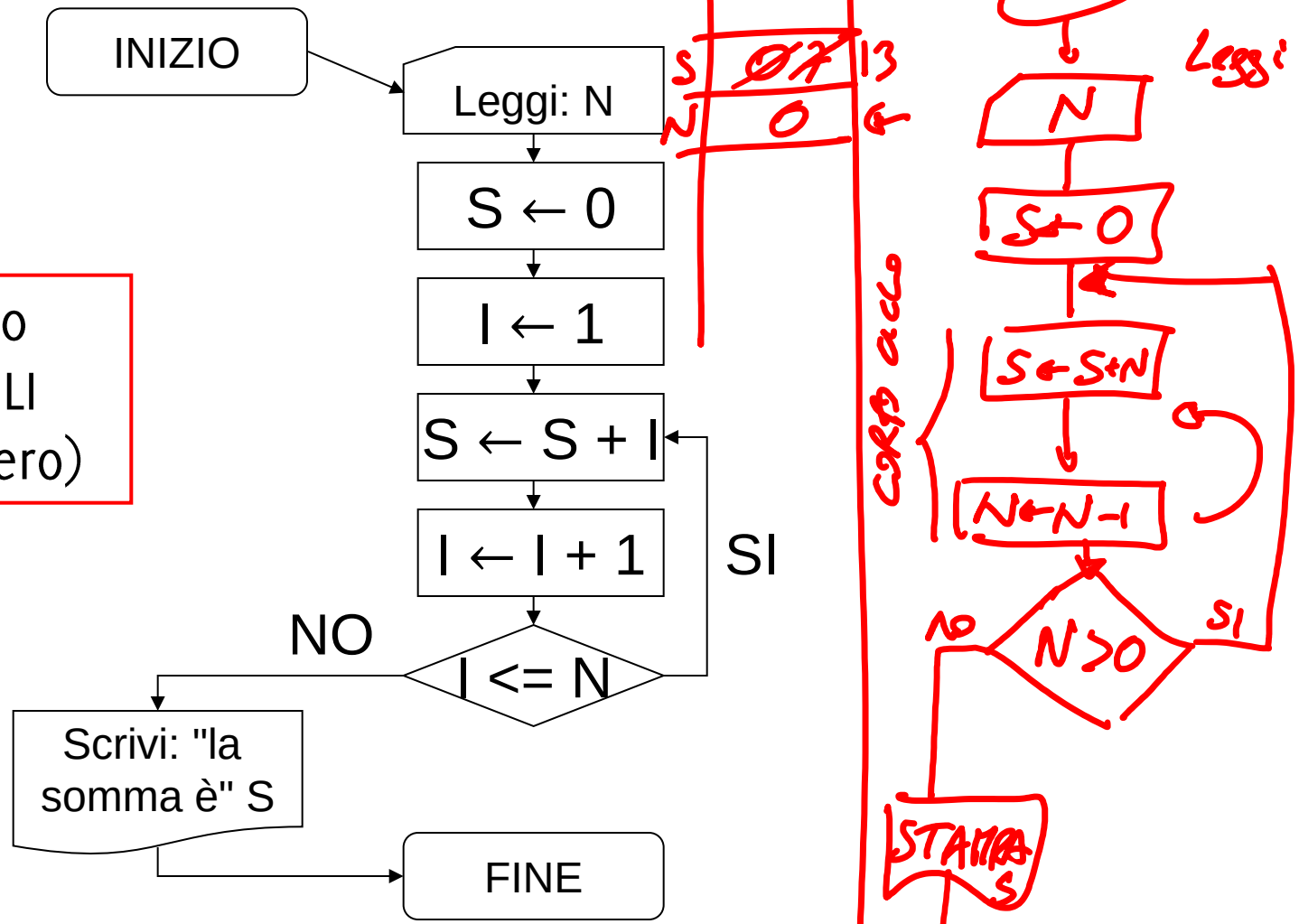
I simboli S, I e N sono definiti come VARIABILI NUMERICHE (di tipo intero)



I è la variabile di regolo il ciclo

R condizione di permanenza nel ciclo

Somma dei primi N numeri naturali



I simboli S, I e N sono definiti come VARIABILI NUMERICHE (di tipo intero)

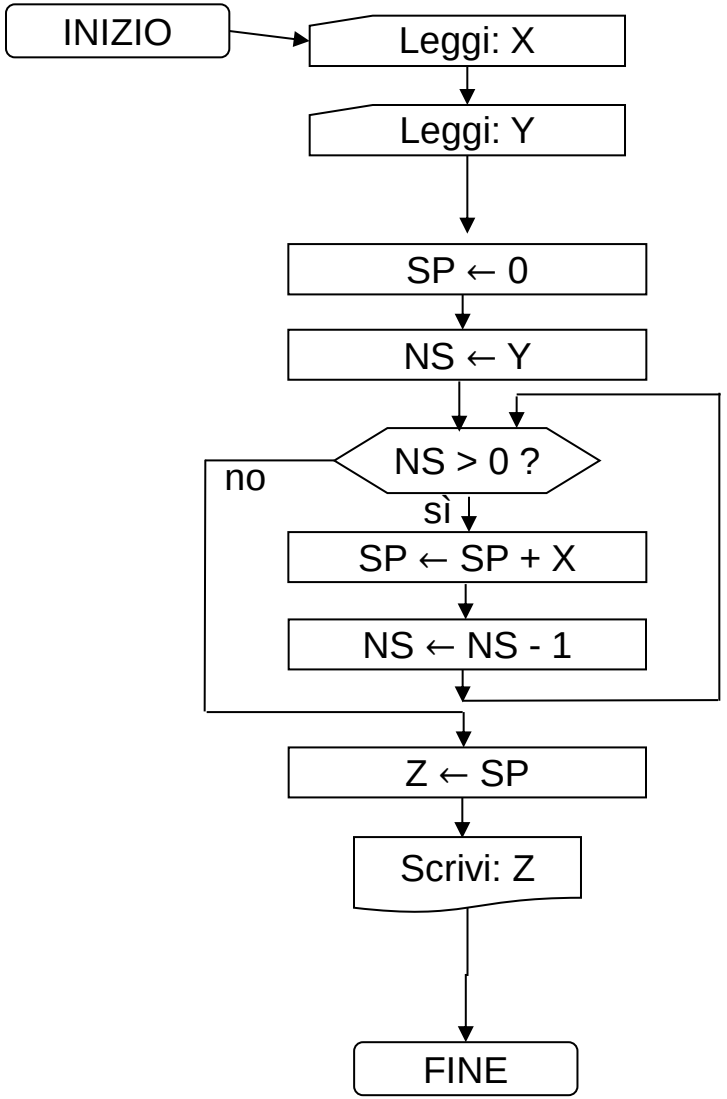
S ← 0 + 7
S ← S + N
N ← 7 - 1
N ← N - 1 ∅

Prodotto per somme ripetute

Scrivere un programma che calcola il prodotto di due numeri X,Y inseriti dall'utente.

Il programma può far solo uso dell'operazione di somma

Prodotto per somme ripetute

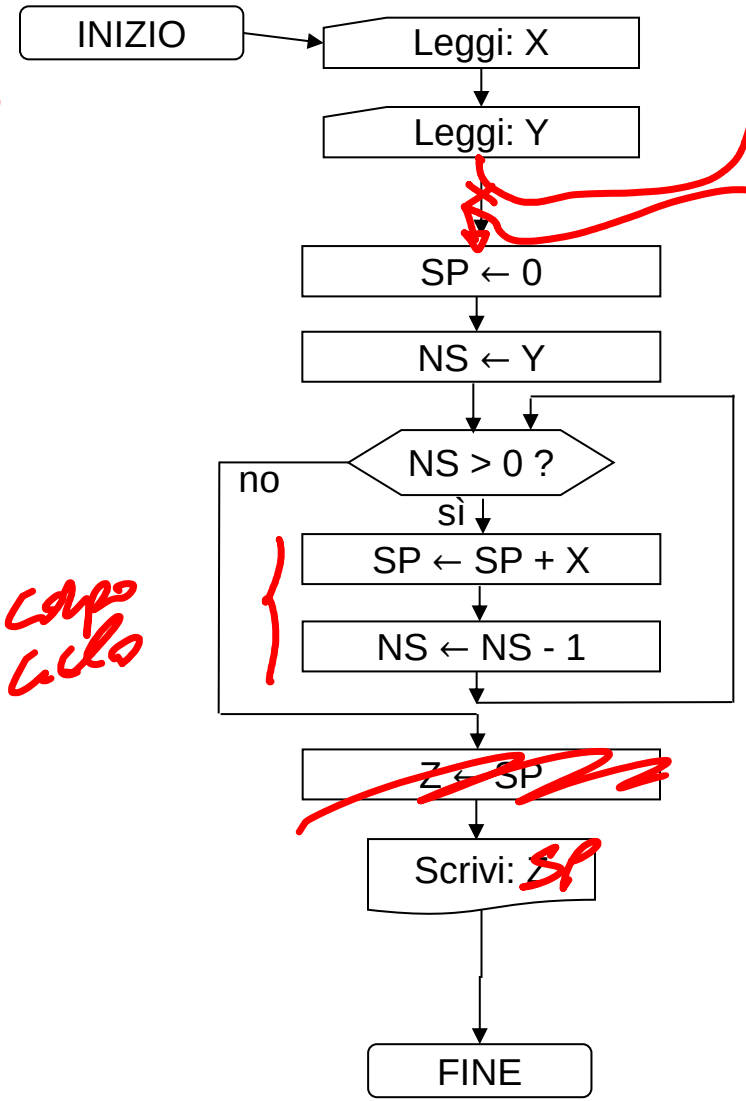


Legenda:

NS: numero somme

SP: somma parziale

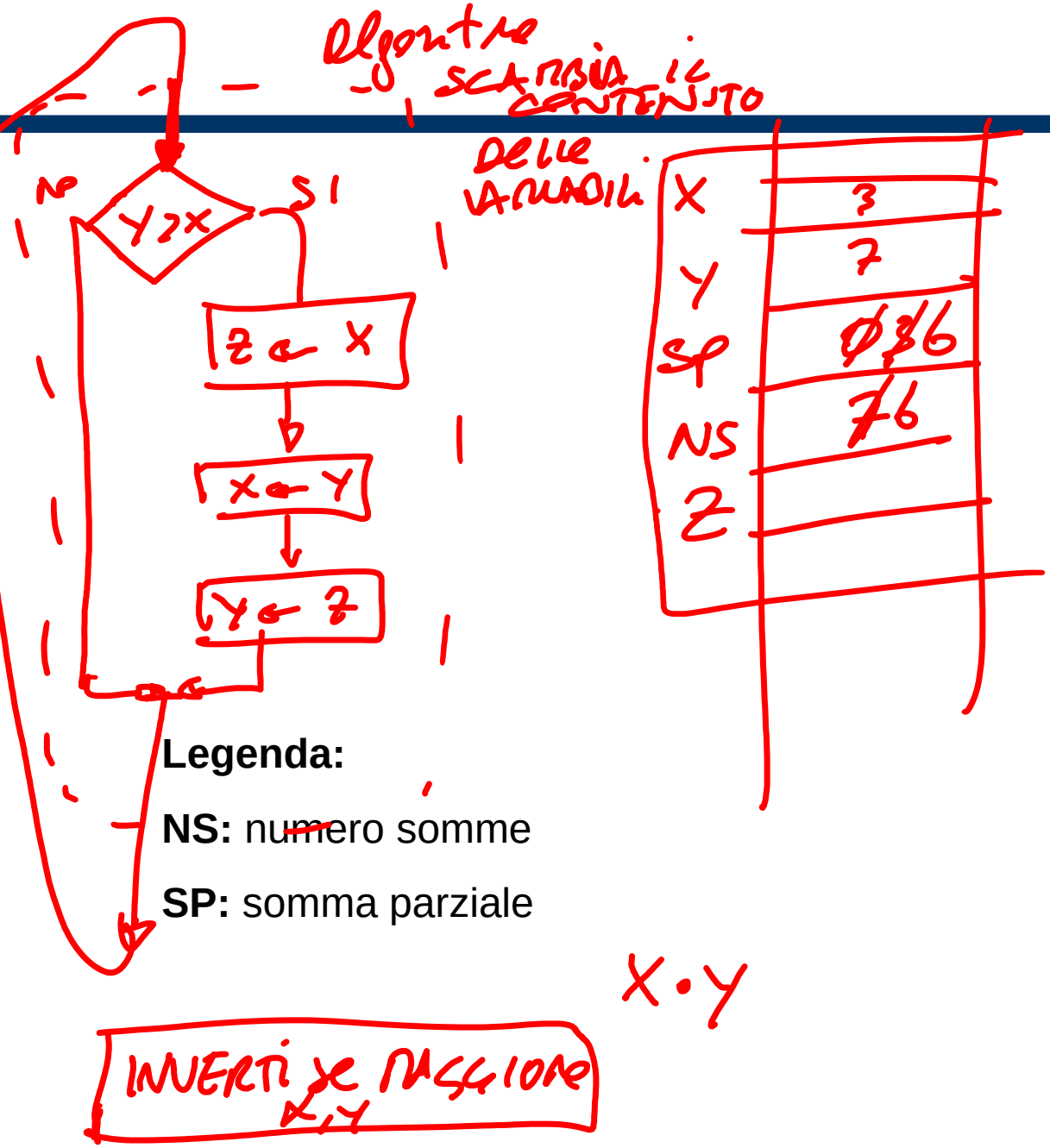
Prodotto per somme ripetute



sommo X Y-volte e lo accumulo in SP

corpo ciclo

3*7



Algoritmo scambia il contenuto delle variabili

Legenda:

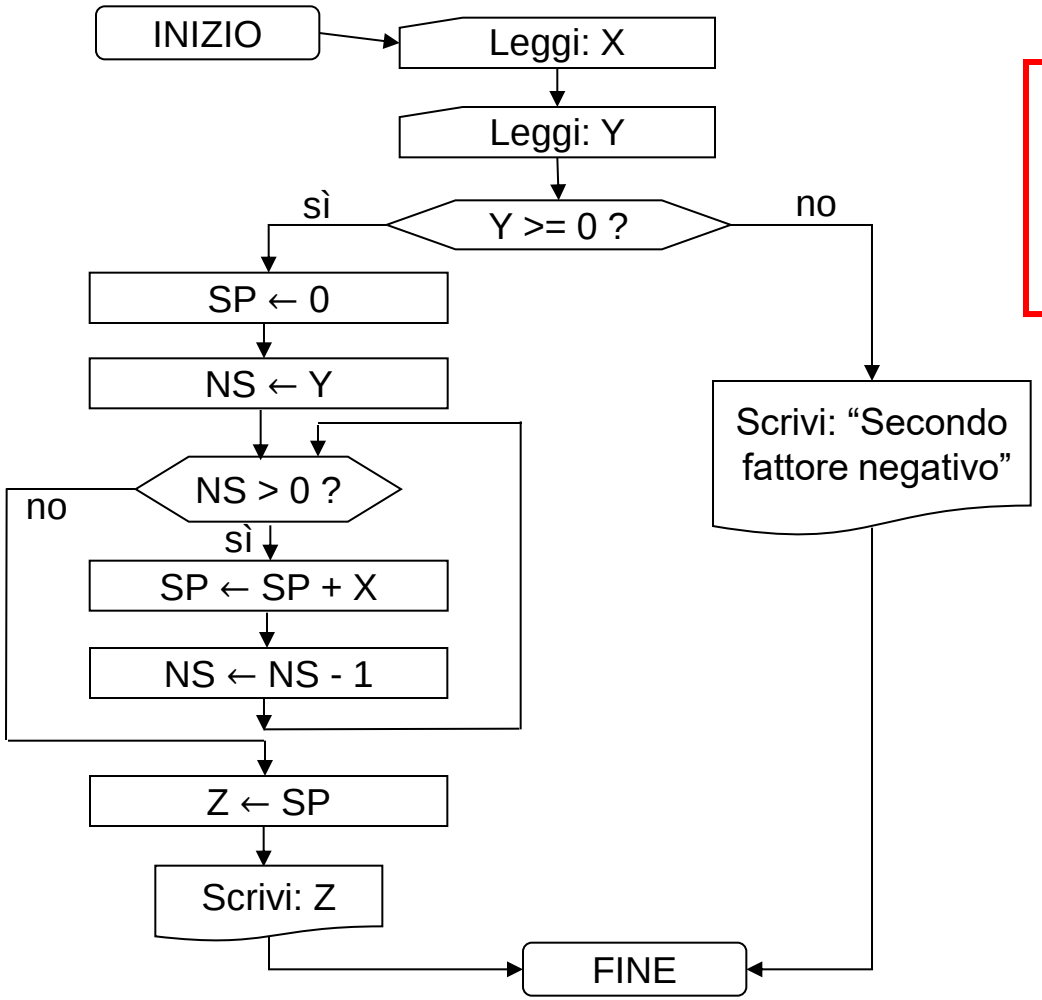
NS: numero somme

SP: somma parziale

X·Y

INVERTI SEQUENZA X,Y

Prodotto per somme ripetute



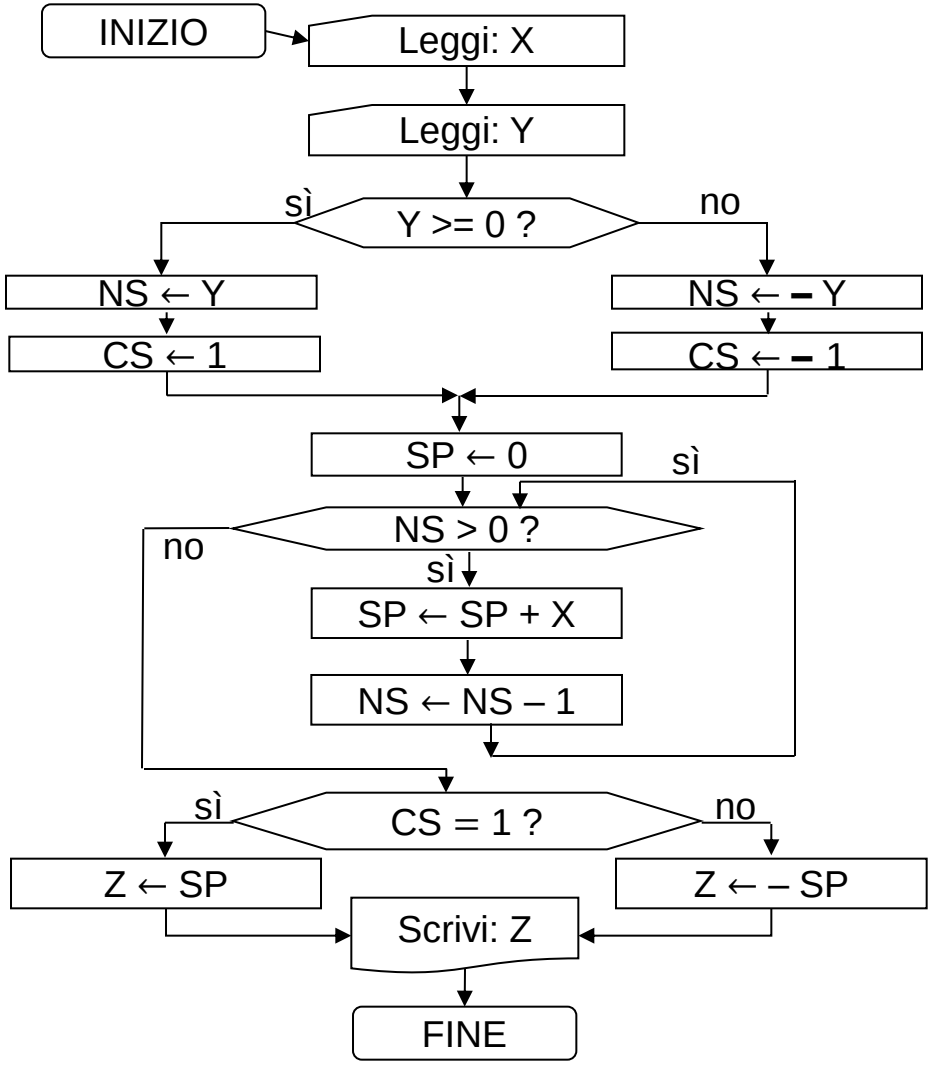
Ipotesi: l'algoritmo non calcola il prodotto nei casi in cui Y è < 0

Legenda:

NS: numero somme

SP: somma parziale

Prodotto per somme ripetute



L'algoritmo calcola il prodotto in ogni caso, anche con fattori negativi

Legenda:

NS: numero somme

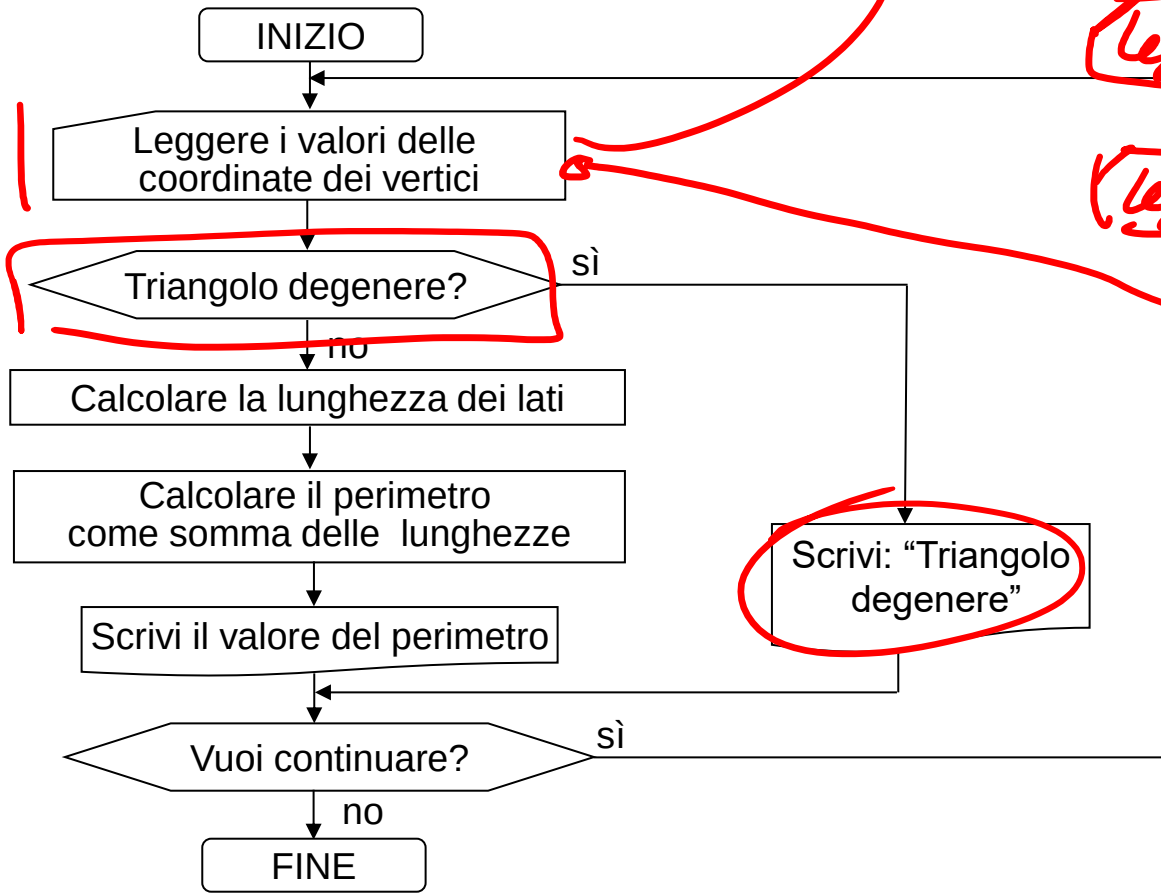
SP: somma parziale

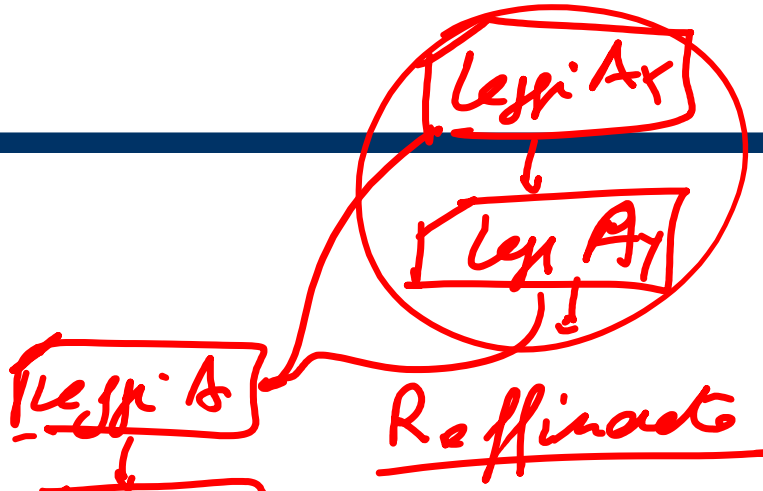
CS: coefficiente segno

Triangoli non degeneri e perimetro

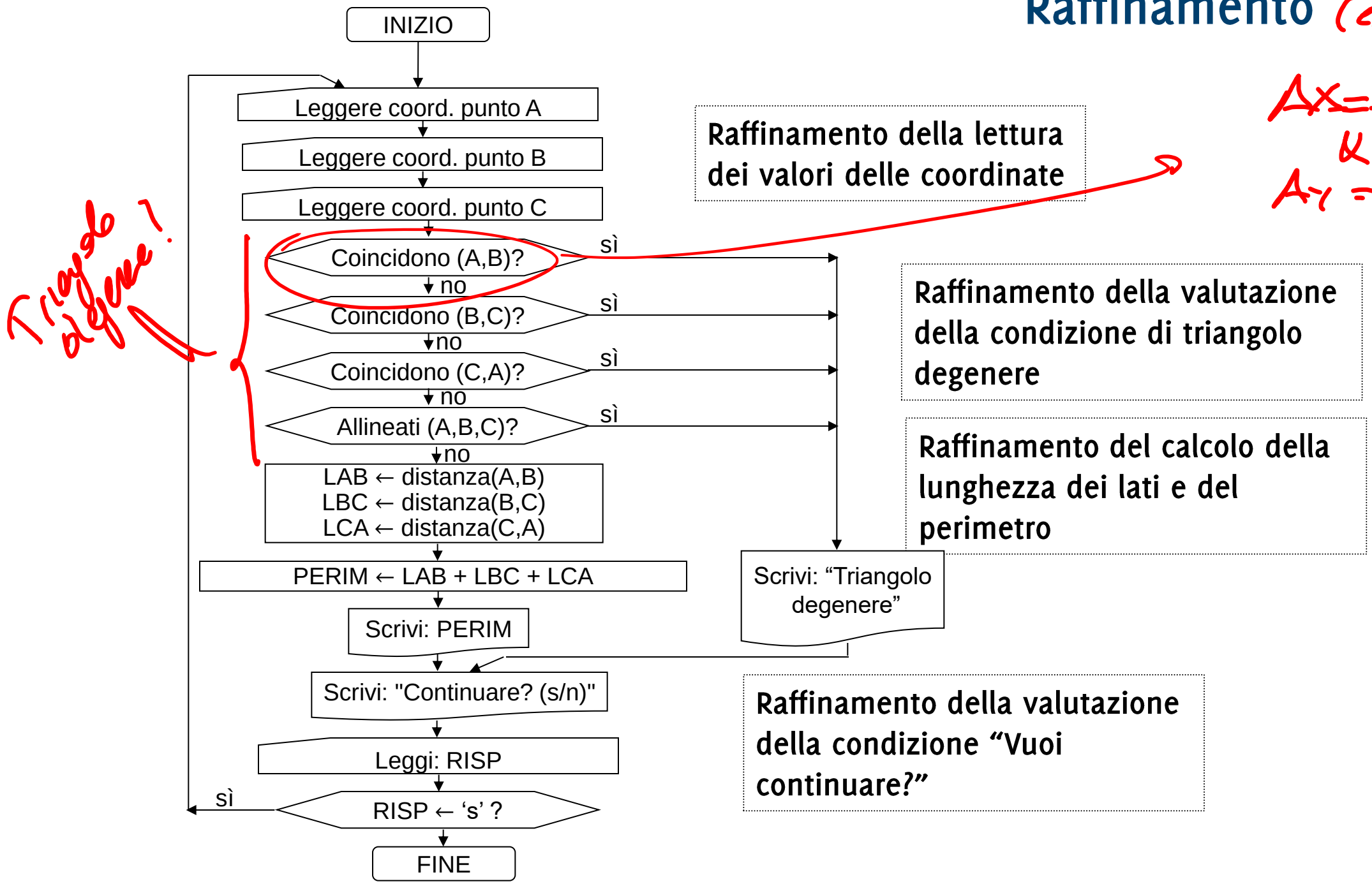
Problema: date le coordinate di tre punti, riconoscere se sono i vertici di un triangolo non degenere, e nel caso calcolarne il perimetro. Ripetere la sequenza di istruzioni se richiesto.

Triangoli non degeneri e perimetro





Raffinamento (espansione)



Concetto di sottoprogramma

Operazioni elementari: **direttamente eseguibili** dall'esecutore

Direttive **complesse**: devono essere **raffinate** ed espresse in termini di **operazioni elementari**

Raffinamento di direttive complesse: realizzabile a parte rispetto all'algoritmo principale

Le direttive complesse possono essere considerate come sottoproblemi da risolvere con un algoritmo dedicato

Sottoprogrammi: descrizioni di questi algoritmi "accessori"

Direttive Complesse: sono chiamate dai sottoprogrammi all'interno dei programmi principali

Vantaggi nell'impiego dei sottoprogrammi

Chiarezza del programma principale

- Tutti i dettagli sono descritti nei sottoprogrammi
- Il programma principale descrive la struttura di controllo generale

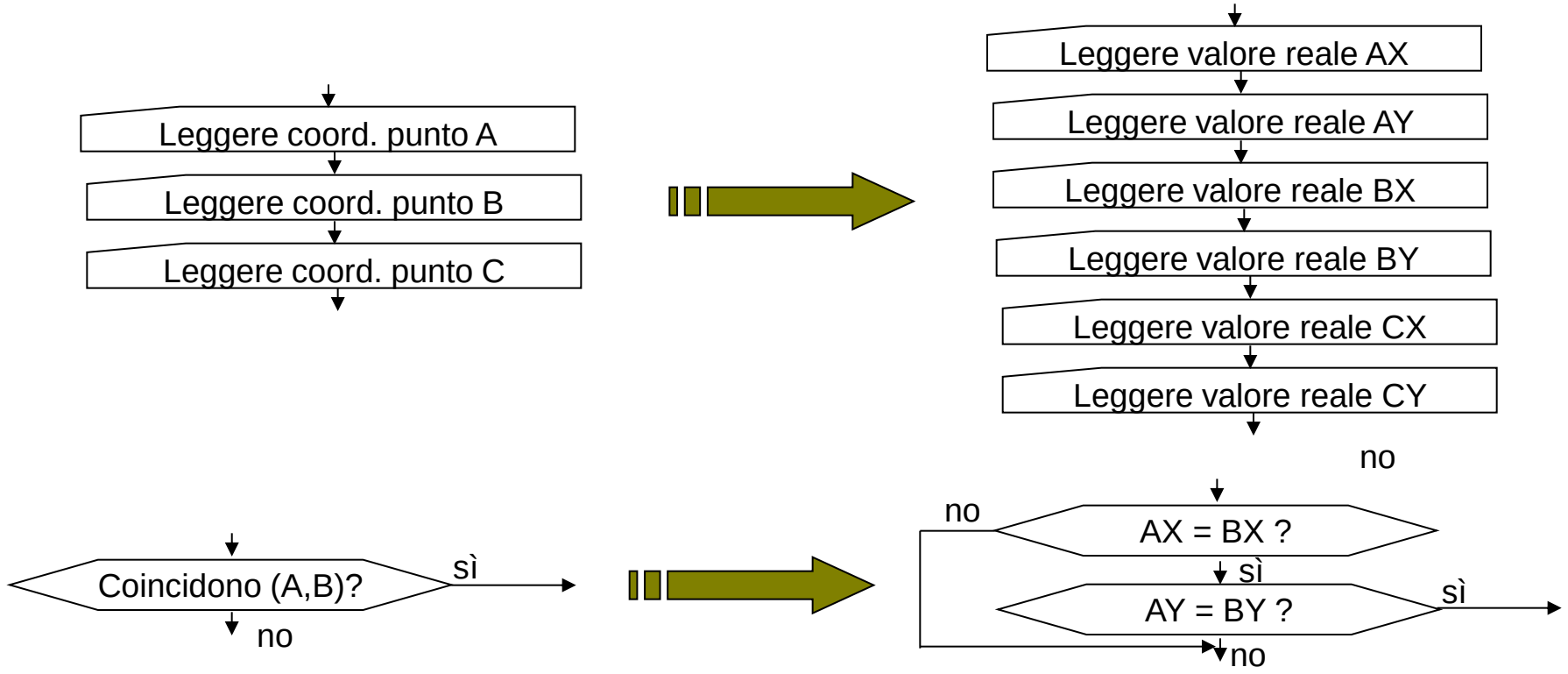
Si evitano ripetizioni

- Alcuni sottoproblemi devono essere affrontati più volte nella soluzione di un problema principale
- il sottoprogramma può essere richiamato tutte le volte che sia necessario

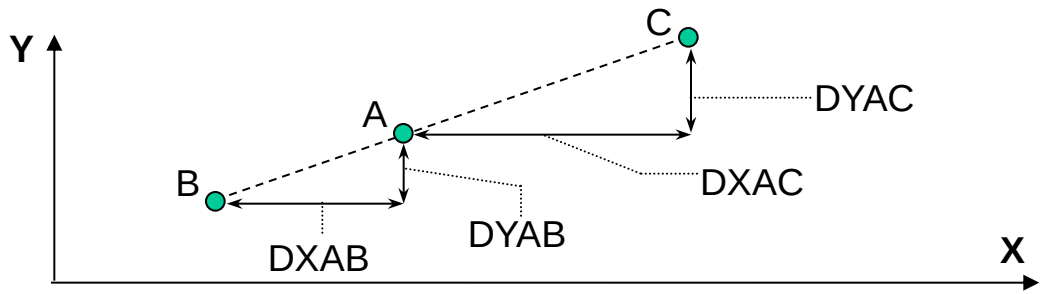
Disponibilità di "sottoprogrammi" prefabbricati

- Sottoproblemi ricorrenti hanno soluzioni svilupapte da programmatori esperti, raccolti nelle cosiddette "librerie" di sottoprogrammi

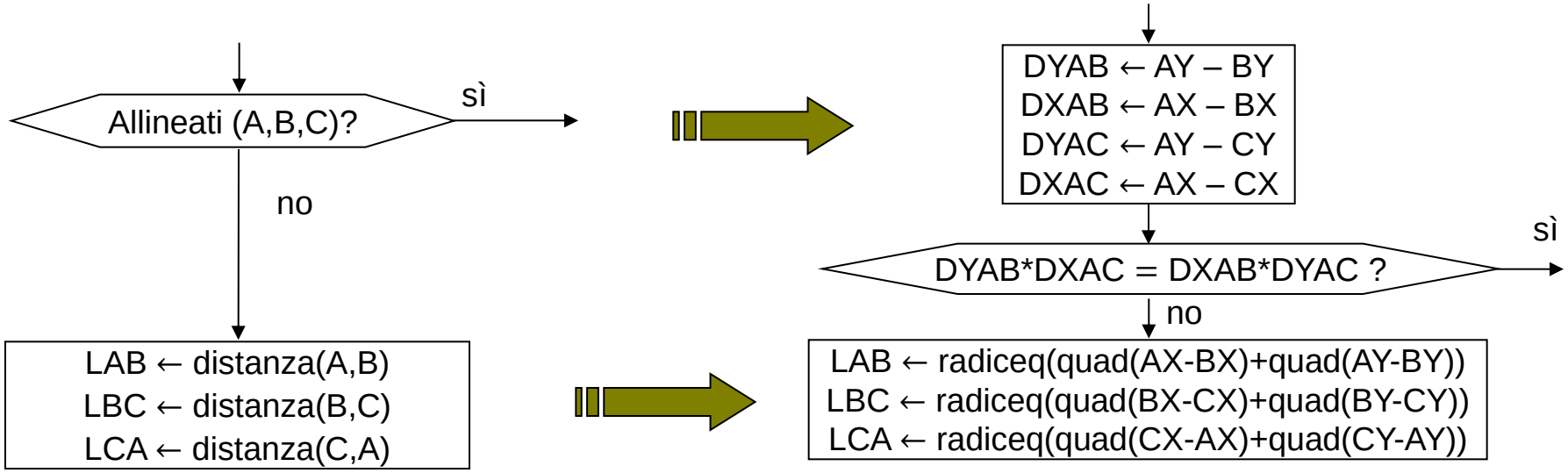
Espansione delle direttive complesse



Raffinamento



Se A, B, C sono allineati, vale la proporzione DYAB : DXAB = DYAC : DXAC



quad(N) indica N*N

radiceq(N) indica $\sqrt{N}$

I Programmi

Linguaggi di programmazione (formali, per la codifica)

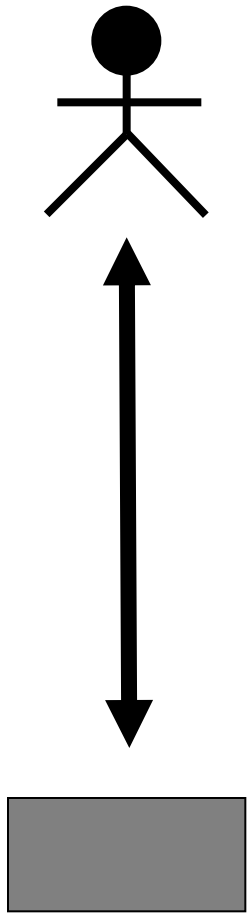
Consentono di scrivere gli algoritmi sotto forma di programmi eseguibili dal calcolatore

- Codificare gli algoritmi

Sono suddivisi in:

- Linguaggi di alto livello
  - linguisticamente più vicini al linguaggio naturale
- Linguaggi assembler
  - più vicini al codice macchina

Il concetto di “livello” dei linguaggi di programmazione



Il programmatore

alto

*Il **livello** del linguaggio*

basso

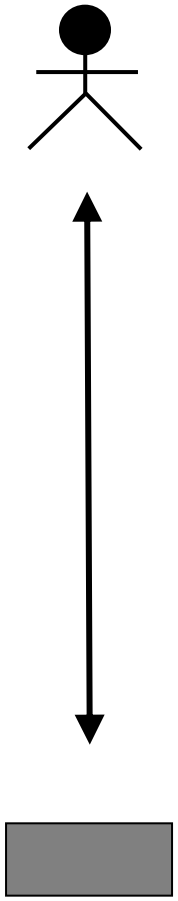
La macchina hardware

Esempi

Linguaggio C

Linguaggio assembler

Linguaggio macchina



TOT=PAGA+STRAORD;

LOAD PAGA
ADD STRAORD
STORE TOT

0100001111
1100111001
0110001111

La “Babele” dei linguaggi

Problemi di comunicazione e compatibilità

Opportunità di specializzazione

- Inizialmente si usava direttamente il linguaggio della macchina
- Nella seconda metà degli anni '50, il linguaggio si alza di livello
  - Si usano programmi, **i compilatori**, che traducono (programmi scritti ne)i linguaggi di più alto livello nel linguaggio della macchina
  - Opportunità: **traduzioni diverse** dello stesso programma “alto” verso i linguaggi “bassi” di **macchine diverse**

Componenti di un linguaggio

Vocabolario: parole chiave che costituiscono il linguaggio

- riconosciute dal parser (analizzatore lessicale)

Sintassi: regole per comporre i simboli del vocabolario

- Il controllo della sintassi avviene tramite l'analizzatore sintattico

Semantica: significato delle espressioni

- Il controllo della semantica è il più difficile
- Un errore semantico si può rilevare, in genere, solo a tempo di esecuzione

Alcuni linguaggi

I primi e tradizionali linguaggi

- Fortran, Cobol

Linguaggi che non “mimano” l’architettura della macchina

- LISP

Linguaggi speciali

- SQL, per interrogazione di database, …

Linguaggi moderni

- C, C++, … Java, C# …. Python …

Compilatori e Interpreti

I **compilatori** sono programmi che traducono i programmi di alto livello in codice macchina

Gli **interpreti**, invece, ne interpretano direttamente le operazioni, eseguendole

Esempi di linguaggi interpretati

- MATLAB, PYTHON,
- BASIC,
- LISP, PROLOG (usati nell'intelligenza artificiale)

Esempi di linguaggi compilati

- C, C++,
- COBOL, PASCAL, FORTRAN

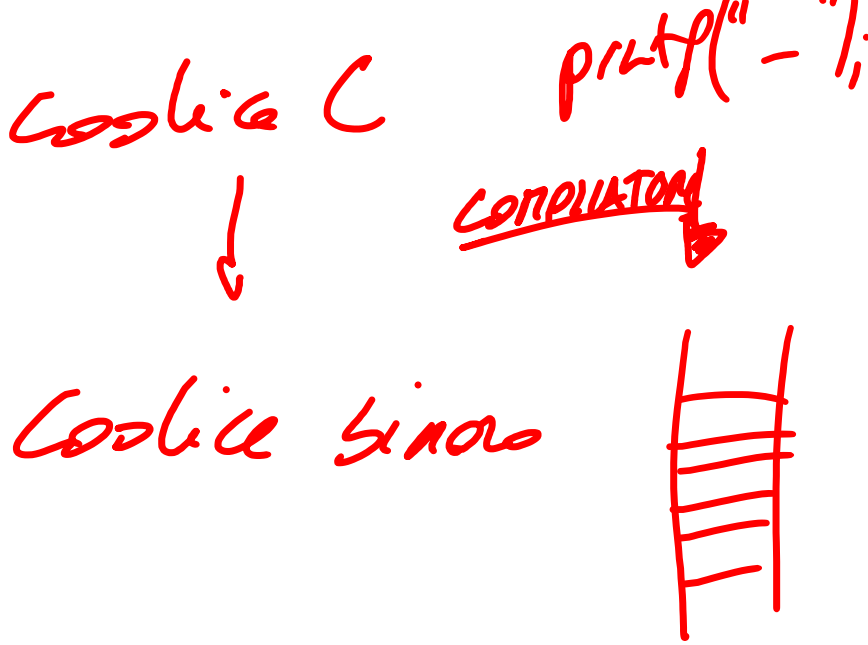
Problemi, Algoritmi, Programmi

Compito dell'informatico è inventare (creare) algoritmi ...

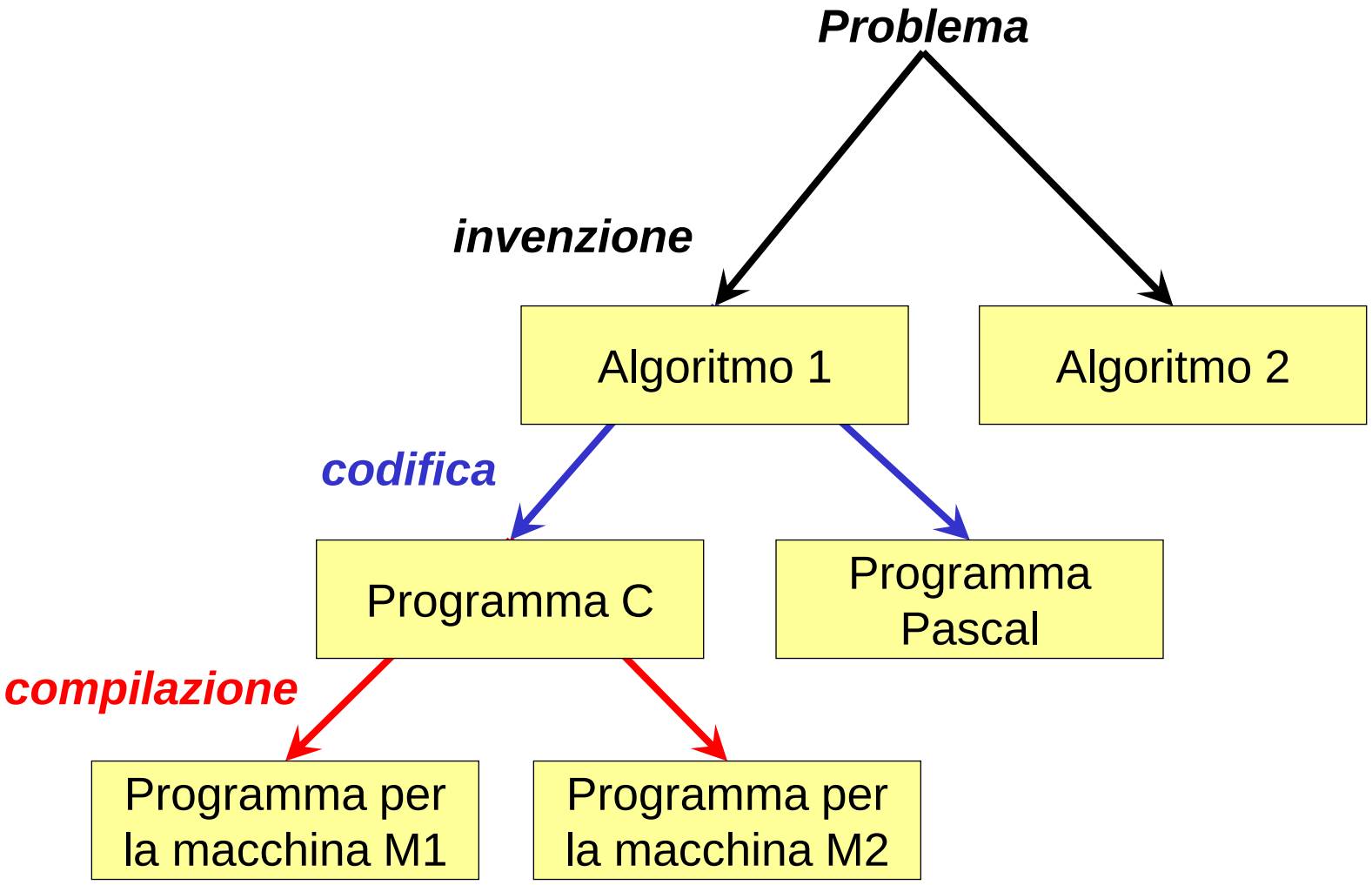
- cioè escogitare e formalizzare le sequenze di passi che risolvono un problema

... e codificarli in programmi

- cioè renderli comprensibili al calcolatore



Problemi, Algoritmi, Programmi



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/ragionevole

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

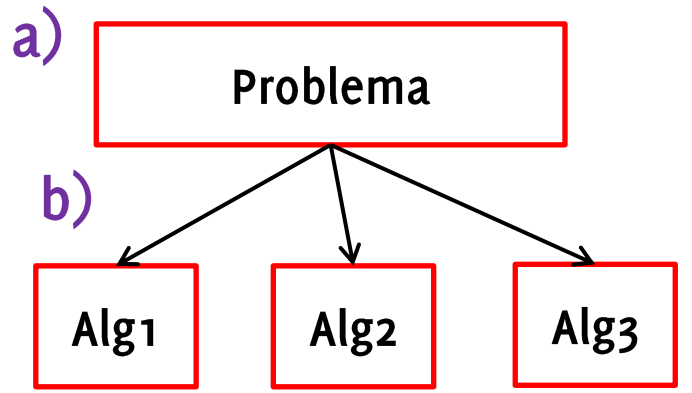
Riepilogando: Come Procedere

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



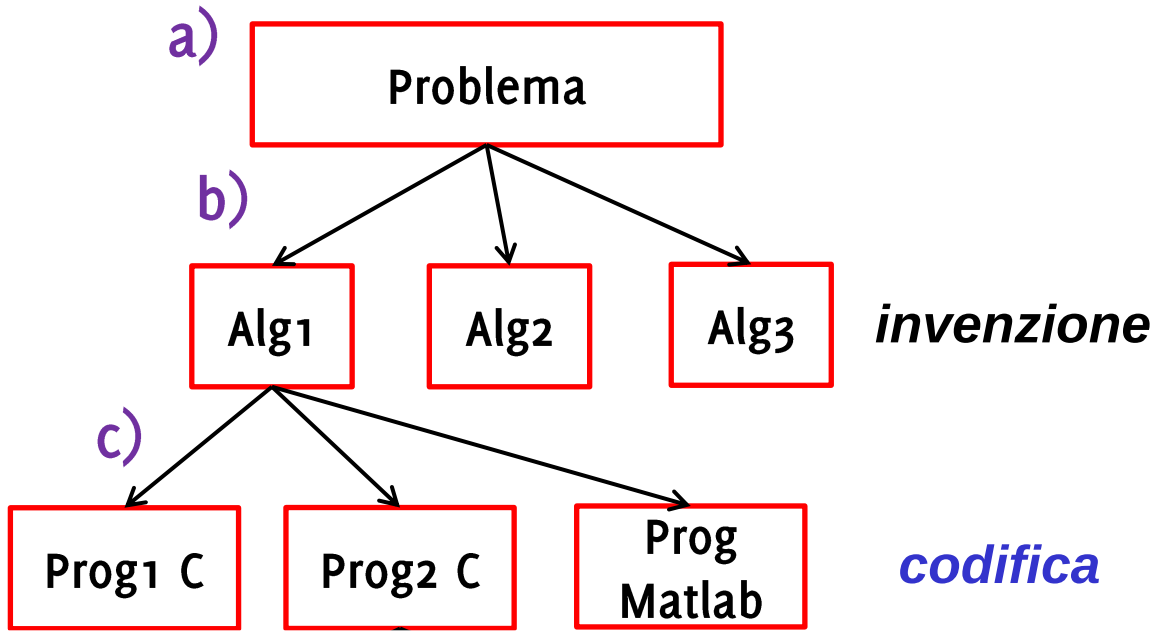
invenzione

Riepilogando: Come Procedere

a) Partirete dall'analisi di un problema

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

c) Codificherete un algoritmo in un linguaggio di programmazione,



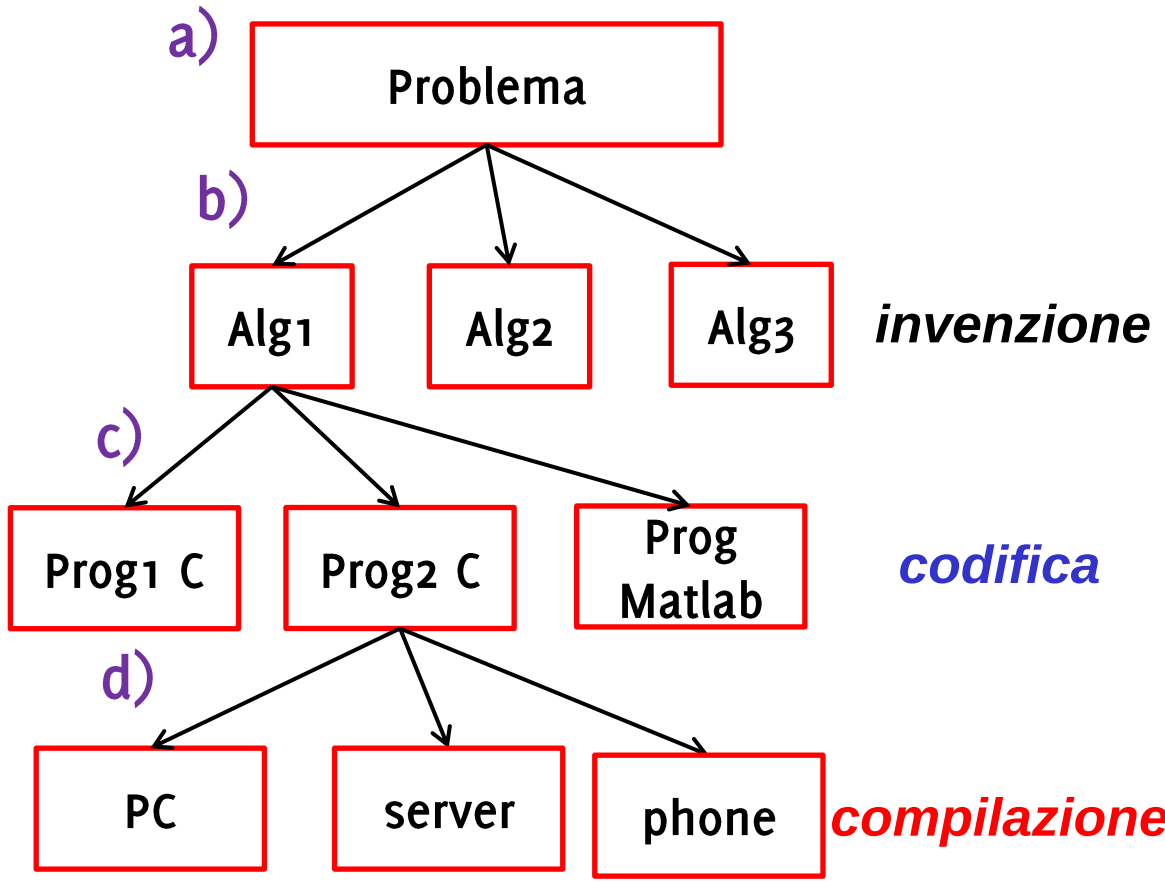
Riepilogando: Come Procedere

a) Partirete dall'analisi di un problema

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

c) Codificherete un algoritmo in un linguaggio di programmazione,

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



Riepilogando: Cosa Fare

a) Dovrete (capire e) definire correttamente il problema

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

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

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