LOGICA, ALGORITMI E PROGRAMMAZIONE STRUTTURATA

Prof R. Bresolin

a.s. 2019 – 2020

La LOGICA

Un aspetto essenziale
del pensiero umano
con profonde influenze
sullo sviluppo delle civiltà

E' possibile darne una definizione precisa?

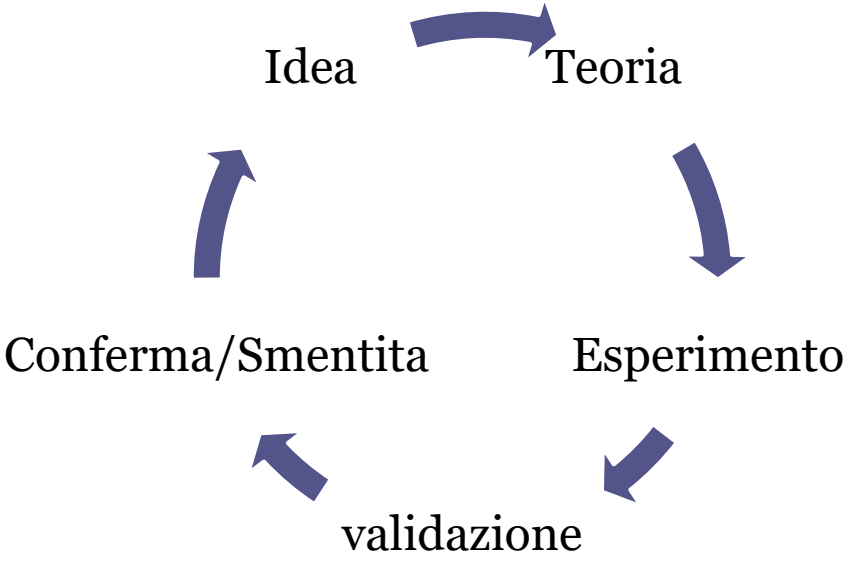
- la molteplicità degli aspetti rende praticamente impossibile un accordo generale sulla definizione
- anche dall'etimologia (la parola greca ***λόγος)*** emerge una diversità di interpretazione secondo che ci si riferisca al significato:

discorso o

ragionamento

La LOGICA: scienza del ragionamento

- Argomento :Si occupa del modo in cui l'uomo ragiona (*ragionamento*)
- Metodo : scientifico



LE TRE VIE PER ARRIVARE ALLO STUDIO DELLA LOGICA

- **DIALETTICA**
- **PARADOSSI**
- **DIMOSTRAZIONI**

1. Dialettica

- Iniziata dai **sofisti(seconda metà V sec. a.C)**: **Protagora e Gorgia** (protagonisti di dialoghi platonici)
- *sofista* nel linguaggio comune è sinonimo di persona che fa discorsi capziosi
- I sofisti in realtà si occupavano **dell'arte del discorso e quindi ne studiavano le regole**
- per questo motivo si possono intravvedere nel loro pensiero le origini della logica

2. Paradossi

ragionamenti apparentemente corretti,
ma che portano a conclusioni
che contraddicono l'opinione corrente: *παρά-δοξα*
*(*che porta ad una contraddizione)
La logica studierà che cosa sta dietro questo tipo di ragionamento, come analizzarlo , come riformularlo.

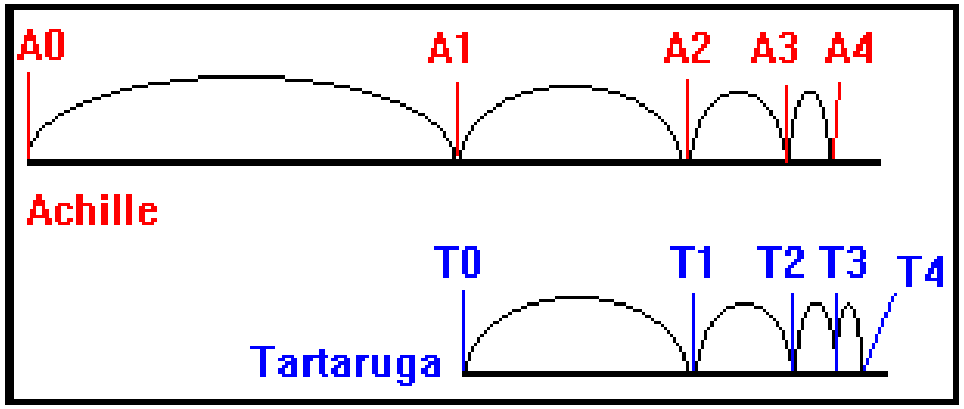
I più famosi paradossi della antichità:

a. **Mentitore (** cretese **Epimenide –VI sec.a.C)**:
Tutti i cretesi sono bugiardi

PARADOSSI

.B) **Achille e la tartaruga (Zenone 490-430 a.C)**:

Achille sembra che non possa raggiungere mai la tartaruga.



Il paradosso del mentitore

- Epimenide diceva: Tutti i Cretesi sono mentitori"
- Epimenide, che era Cretese, diceva la verità?

Il paradosso di Achille e della Tartaruga

- Achille fa una gara di corsa con una tartaruga. Egli corre dieci volte più veloce della tartaruga, perciò decide di darle 10 m di vantaggio.
- In questo modo Achille non riuscirà mai a raggiungere la tartaruga, infatti:

1. per raggiungere la tartaruga Achille dovrà percorrere 10 m, ma nel frattempo la tartaruga avrà percorso 1 m e quindi sarà ancora in vantaggio…
2. per raggiungere la tartaruga Achille dovrà percorrere 1 m, ma nel frattempo la tartaruga avrà percorso 10 cm e quindi sarà ancora in vantaggio…
3. per raggiungere la tartaruga Achille dovrà percorrere 10 cm, ma nel frattempo la tartaruga avrà percorso 1 cm e quindi sarà ancora in vantaggio…
4. per raggiungere la tartaruga Achille dovrà percorrere 1 cm, ma nel frattempo la tartaruga avrà percorso 1 mm e quindi sarà ancora in vantaggio…

- Poiché questa situazione si ripete all'infinito, Achille, il corridore più veloce della Grecia, non raggiungerà mai la tartaruga……
- Il ragionamento è corretto?

Il paradosso della previsione

- Un condannato a morte riceve un messaggio di questo tipo, da parte del boia.
- L'esecuzione avverrà la settimana prossima in un giorno a sorpresa che tu non potrai in alcun modo prevedere.
- Il condannato ragiona così:

1. non può essere sabato, perché giunto a venerdì senza essere stato ucciso, io potrei prevederlo;
2. non può essere neppure venerdì perché giunto a giovedì ancora vivo potrei prevederlo;
3. non può essere giovedì perché...

In conclusione, se il boia mantiene quanto ha detto, non può eseguire la sentenza!

3. Dimostrazioni

- La matematica nasce **senza dimostrazioni** (v. papiro di Rhind)

Poiché i risultati venivano riportati senza giustificazione non si poteva avere la sicurezza della correttezza della intuizione

- Verso il 600 a.C. i greci,

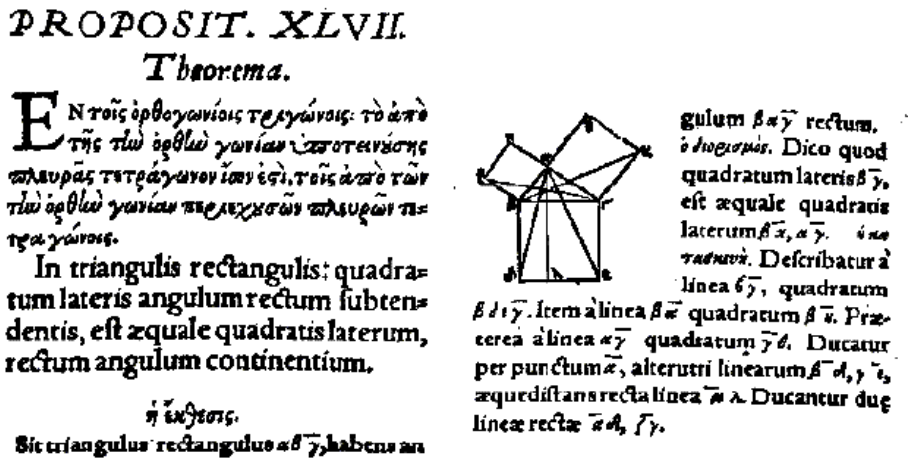
inventarono " un nuovo modo di fare matematica" : la dimostrazione.

- **La logica si occupa di che cosa rende corretta una dimostrazione**

- furono stimolati nello studio delle dimostrazioni da due famosi risultati :

Dimostrazioni

Il teorema di Pitagora intuito dalle civiltà precedenti , come riportato da Euclide prevede una dimostrazione molto complessa



PROPOSIT. XLVII.

Theorema.

ΕΝ τοῖς ὀρθογωνίοις τριγώνοις· τὸ ἀπὸ τῆς τὴν ὀρθὴν γωνίαν ὑποτεινούσης πλευρᾶς τετράγωνον ἴσον ἐστὶ, τοῖς ἀπὸ τῶν τὴν ὀρθὴν γωνίαν περιεχουσῶν πλευρῶν τετραγώνοις.

In triangulis rectangulis: quadratum lateris angulum rectum subtendentis, eſt æquale quadratis laterum, rectum angulum continentium.

ἡ ἔκθεσις.

Sit triangulus rectangulus $\alpha\beta\gamma$, habens an gulum $\beta\alpha\gamma$ rectum. ὁ διορισμὸς. Dico quod quadratum lateris $\beta\gamma$, eſt æquale quadratis laterum $\beta\alpha, \alpha\gamma$. ἀπόδειξις. Deſcribatur à linea $\beta\gamma$, quadratum $\beta\delta\varepsilon\gamma$. Item à linea $\beta\alpha$ quadratum $\beta\zeta$. Præterea à linea $\alpha\gamma$ quadratum $\gamma\theta$. Ducatur per punctum α, alterutri linearum $\beta\delta, \gamma\varepsilon$, æquedistans recta linea $\alpha\lambda$. Ducantur duæ lineæ rectæ $\zeta\alpha, \beta\gamma$.

Dimostrazioni

- Irrazionalità della radice di 2

Scoperta dai pitagorici rompe il connubio tra costruzione e misura: la realtà si mostra più estesa di ciò che è razionale .

- Rappresenta il primo esempio di dimostrazione per assurdo.

Cos'è la logica?

Nel linguaggio comune:

- modo di ragionare coerente, convincente

ma anche

- modo di legare parole o azioni che non trova necessariamente la sua spiegazione nella ragione, ma piuttosto nei sentimenti, nell'età, nel proprio campo di attività: *la logica del cuore, la logica infantile, …*

PREMESSE ⟶ **CONCLUSIONI**

deduzione

Cos'è la logica?

In senso matematico:

- si occupa dello studio delle regole che permettono di sviluppare un ragionamento in modo che, da affermazioni vere - dette premesse - , si giunga a conclusioni altrettanto vere attraverso un processo che prende il nome di *deduzione*

PREMESSE ⟶ **CONCLUSIONI**

deduzione

Distinguiamo allora…

Ragionamenti **logicamente validi**:

sono quelli per cui, se le premesse sono vere, le conclusioni sono anch'esse vere

Ragionamenti **corretti**:

sono quelli che, essendo logicamente validi, partono da premesse vere

TUTTI I RAGIONAMENTI CORRETTI SONO *LOGICAMENTE VALIDI*, MA NON TUTTI I RAGIONAMENTI LOGICAMENTE VALIDI SONO *CORRETTI*.

Proposizioni (logiche)

Sono quegli enunciati di cui si possa dire se sono **veri** (T = *true*) o **falsi** (F = *false*).

Ogni proposizione logica (detta anche *proposizione in senso matematico*) ha quindi un ***valore di verità***: T oppure F

In generale NON sono proposizioni (logiche):
- domande, esclamazioni, comandi
- frasi con riferimenti al futuro
- frasi del tipo: ‘mi piace’, ‘è bello’, …

Qualche esempio

1) Hai fame? **Non è una proposizione**
2) Roma è la capitale d'Italia. **È una proposizione vera**
3) Il cane è un bipede. **È una proposizione falsa**
4) 2 + 2 = 5 **È una proposizione falsa**
5) 56 > 25 **È una proposizione vera**
6) Devi studiare matematica! **Non è una proposizione**
7) Tom Cruise è un bravo attore. **Non è una proposizione**

Varranno i seguenti principi:

1 – principio del terzo escluso: una proposizione non può assumere valori diversi da T (vero) o F (falso)

2 – principio di non contraddizione: una stessa proposizione non può essere sia T (vera) sia F (falsa)

3 – principio di identità: assicura la costanza del valore di verità (T o F) di una proposizione nell'ambito di un certo discorso

Questi principi sono validi per la logica a due valori.

Proposizione semplici e composte

A: "Il numero 5 è primo"

B: "Il numero 7 è pari"

Sono ***proposizioni semplici*** (o elementari) in quanto contengono un solo predicato, riferito ad un solo soggetto.

Le lettere maiuscole utilizzate per "rappresentarle" prendono il nome di ***variabili logiche***.

Proposizione semplici e composte

A: “Il numero 5 è primo”

B: “Il numero 7 è pari”

1 - Il numero 5 non è primo.

2 - Il numero 5 è primo e il numero 7 è pari.

3 - Il numero 5 è primo o il numero 7 è pari.

4 - Se il numero 7 è pari, allora 5 è primo.

Le proposizioni dalla 1 alla 4 sono tutte composte, formate da più proposizioni semplici legate tra di loro da connettivi (detti *connettivi logici*).

I connettivi logici

A: “Il numero 5 è primo” B: “Il numero 7 è pari”

La negazione: **NON**

$\overline{A}$: “Il numero 5 non è primo”

La congiunzione: **E**

$A \wedge B$: “Il numero 5 è primo e il numero 7 è pari”

La disgiunzione (non esclusiva): **O**

$A \vee B$: “Il numero 5 è primo o il numero 7 è pari”

L’implicazione (materiale): **SE… ALLORA**

$B \rightarrow A$: “Se il numero 7 è pari allora il numero 5 è primo”

La negazione

La negazione di una proposizione A è la proposizione non A, che risulta:

vera se A è falsa

falsa se A è vera

La negazione di una proposizione A si indica con il simbolo di sopralineatura.

A	$\overline{A}$
T	F
F	T

tavola di verità del connettivo **non**

A: “Milano è una metropoli”

$\overline{A}$: “Milano non è una metropoli”

“Non è vero che Milano è una metropoli”

Non è corretto: “*Milano è un villaggio*”

La congiunzione

La congiunzione di due proposizioni A e B è la proposizione A **e** B; essa risulta essere vera solo se lo sono entrambe le proposizioni che la compongono.

La congiunzione di due proposizioni si indica con il simbolo $\wedge$ (et).

tavola di verità del connettivo **et**

A	B	A $\wedge$ B
T	T	T
T	F	F
F	T	F
F	F	F

La congiunzione

Osservazione 1: Non sempre in una proposizione composta mediante il connettivo **et** compaiono due distinti predicati.

A	B	A ∧ B
T	T	T
T	F	F
F	T	F
F	F	F

p.e. "L'autostrada A1 passa da Bologna e da Firenze"

comprende le due seguenti proposizioni:

A: "L'autostrada A1 passa da Bologna"

B: "L'autostrada A1 passa da Firenze"

La congiunzione

Osservazione 2: Non sempre la presenza della congiunzione **e** in una affermazione corrisponde all'uso del connettivo **et**.

A	B	A ∧ B
T	T	T
T	F	F
F	T	F
F	F	F

p.e. "Il mantello della zebra è bianco e nero"

esprime un'unica proprietà del soggetto
e non la presenza contemporanea di due proprietà distinte

(come sarebbe invece nella proposizione
"Maria ha i capelli lunghi e lisci")

La congiunzione

A	B	A ∧ B
T	T	T
T	F	F
F	T	F
F	F	F

Osservazione 3: Nel linguaggio naturale ci sono altre congiunzioni (diverse da e) che dal punto di vista logico risultano equivalenti al connettivo **et**.

p.e. La prof.ssa Boselli non insegna inglese, ma insegna tedesco. (FALSA: perché?)

L'ornitorinco è un oviparo, che allatta i suoi piccoli.

La luna non è un pianeta, bensì un satellite.

Infatti le proposizioni proposte come esempio possono essere vere solo se entrambe le proposizioni che le compongono sono vere!!

La congiunzione

A	B	A ∧ B
T	T	T
T	F	F
F	T	F
F	F	F

<u>Osservazione 4</u>: Si può essere in presenza di una proposizione ottenuta come congiunzione logica, anche se sono assenti congiunzioni in senso grammaticale.

p.e. Roma è la capitale d'Italia; Parigi è la capitale della Francia.

Mario è partito mercoledì per Roma.

(Mario è partito mercoledì ed è partito per Roma)

5 è un numero naturale pari.

(5 è un numero naturale ed è un numero pari)

La disgiunzione non esclusiva

La disgiunzione non esclusiva (o disgiunzione inclusiva) di due proposizioni A e B è la proposizione A **o** B; essa risulta essere falsa solo se sono false entrambe le proposizioni che la compongono; è vera in tutti gli altri casi.

La disgiunzione non esclusiva di due proposizioni si indica con il simbolo $\vee$ (vel).

tavola di verità del connettivo **vel**

A	B	A $\vee$ B
T	T	T
T	F	T
F	T	T
F	F	F

La disgiunzione non esclusiva

Si parla di disgiunzione non esclusiva in quanto non si richiede che il verificarsi di una delle due proposizioni che la compongono escluda il verificarsi dell'altra.

A	B	A ∨ B
T	T	T
T	F	T
F	T	T
F	F	F

p.e. "Milano è capoluogo di provincia o di regione" è vera

Il connettivo **vel** corrisponde quindi a ciò che nel linguaggio commerciale viene indicato con **e/o**; nella lingua italiana non esiste un vocabolo che possieda in modo altrettanto univoco questo significato; noi lo attribuiremo ad **o**, **oppure**.

La disgiunzione esclusiva

La disgiunzione esclusiva di due proposizioni A e B è la proposizione **o** A **o** B; essa risulta essere vera solo nel caso in cui una delle due proposizioni che la compongono è vera mentre l'altra è falsa.

La disgiunzione esclusiva di due proposizioni si indica con il simbolo ⩒ (**aut**).

tavola di verità del connettivo **aut**

A	B	A ⩒ B
T	T	F
T	F	T
F	T	T
F	F	F

La disgiunzione esclusiva

Osservazione: La tavola di verità del connettivo aut si presenta come quella della espressione logica

$$(A \vee B) \wedge \overline{(A \wedge B)}$$

A	B	A ∨ B
T	T	F
T	F	T
F	T	T
F	F	F

A	B	A ∨ B	A ∧ B	$\overline{A \wedge B}$	(A ∨ B) ∧ $\overline{(A \wedge B)}$
T	T	T	T	F	F
T	F	T	F	T	T
F	T	T	F	T	T
F	F	F	F	T	F

Espressioni logiche

Componendo più variabili logiche mediante i connettivi logici si ottengono delle *espressioni logiche*.

Valgono le seguenti definizioni:

1. Espressioni logiche equivalenti: due espressioni logiche nelle stesse variabili si dicono equivalenti se, in corrispondenza agli stessi valori di verità attribuiti a tali variabili, si ottengono uguali valori di verità per entrambe le espressioni (la loro tavola di verità coincide)
2. Tautologie: sono quelle espressioni logiche per le quali, in corrispondenza a qualsiasi scelta dei valori di verità attribuiti alle variabili che le compongono, assumono comunque valore di verità VERO (la tavola verità assume sempre valore VERO)
3. Contraddizioni: sono quelle espressioni logiche per le quali, in corrispondenza a qualsiasi scelta dei valori di verità attribuiti alle variabili che le compongono, assumono comunque valore di verità FALSO (la tavola verità assume sempre valore FALSO)

L'implicazione materiale

L'implicazione materiale tra due proposizioni A e B è la proposizione A **implica** B; essa risulta essere falsa solo nel caso in cui A sia vera e B sia falsa.

L'implicazione materiale di due proposizioni si indica con il simbolo $\rightarrow$ (**freccia**).

tavola di verità
del connettivo **freccia**

A	B	A $\rightarrow$ B
T	T	T
T	F	F
F	T	T
F	F	T

L'implicazione materiale

A	B	$A \rightarrow B$
T	T	T
T	F	F
F	T	T
F	F	T

Osservazione 1: Nel linguaggio naturale ci sono molti modi diversi di esprimere il connettivo freccia.

$$A \rightarrow B$$

si può leggere:

Se A, allora B

B, se A

Solo se B, allora A

Condizione sufficiente per B è A

Condizione necessaria per A è B

Un esempio....

A : “Mario è milanese” B : “(Mario) è lombardo”

$$A \rightarrow B$$

Se A, allora B : Se Mario è milanese, allora è lombardo

B, se A : Mario è lombardo, se è milanese

Solo se B, allora A : Solo se Mario è lombardo, allora è milanese

Condizione sufficiente per B è A :
Condizione sufficiente perché Mario sia lombardo è che sia milanese

Condizione necessaria per A è B :
Condizione necessaria perché Mario sia milanese è che sia lombardo

L'implicazione materiale

Osservazione 2: Il connettivo freccia ha la tavola di verità equivalente a quella della espressione logica

$$A \wedge \overline{B}$$

A	B	$A \rightarrow B$
T	T	T
T	F	F
F	T	T
F	F	T

A	B	$\overline{B}$	$A \wedge \overline{B}$	$\overline{A \wedge \overline{B}}$
T	T	F	F	T
T	F	T	T	F
F	T	F	F	T
F	F	T	F	T

L'implicazione materiale

1) $\mathbf{A \wedge \overline{B}}$ corrisponde a:

Non può essere vero sia A sia la negazione di B

(e noi sappiamo che in un ragionamento logicamente valido da premesse vere devono derivare conseguenze vere)

A	B	$A \rightarrow B$	$\overline{A \wedge \overline{B}}$
T	T	T	T
T	F	F	F
F	T	T	T
F	F	T	T

2) Il connettivo freccia può essere espresso mediante i connettivi non ed et; in generale, possono ritenersi fondamentali i tre connettivi non, et e vel, che permettono di costruire espressioni logiche con tavole di verità equivalenti a quelle di tutti gli altri connettivi.

connettivo AUT

L'implicazione materiale

A	B	$A \rightarrow B$
T	T	T
T	F	F
F	T	T
F	F	T

Osservazione 3: Nel comporre una proposizione mediante il connettivo freccia non si richiede che ci sia un nesso tra le due proposizioni elementari considerate (questo del resto vale anche per gli altri connettivi)

p.e. "Se Marte è un satellite, allora 6 è un numero pari"
(che è un'affermazione vera)

Ecco perché parliamo di "implicazione materiale" e non di "implicazione logica" (per la quale si addotta un simbolo diverso: $\Rightarrow$).

La doppia implicazione

La doppia implicazione tra due proposizioni A e B è la proposizione A **equivale a** B; essa risulta essere vera nei casi in cui A e B siano entrambe vere o entrambe false.

La doppia implicazione fra due proposizioni si indica con il simbolo $\leftrightarrow$ (doppia freccia).

tavola di verità del connettivo **doppia freccia**

A	B	$A \leftrightarrow B$
T	T	T
T	F	F
F	T	F
F	F	T

La doppia implicazione

Osservazione: Si può verificare l'equivalenza tra la tavola di verità del connettivo doppia freccia e quella dell'espressione logica:

A	B	$A \leftrightarrow B$
T	T	T
T	F	F
F	T	F
F	F	T

$$(A \rightarrow B) \wedge (B \rightarrow A)$$

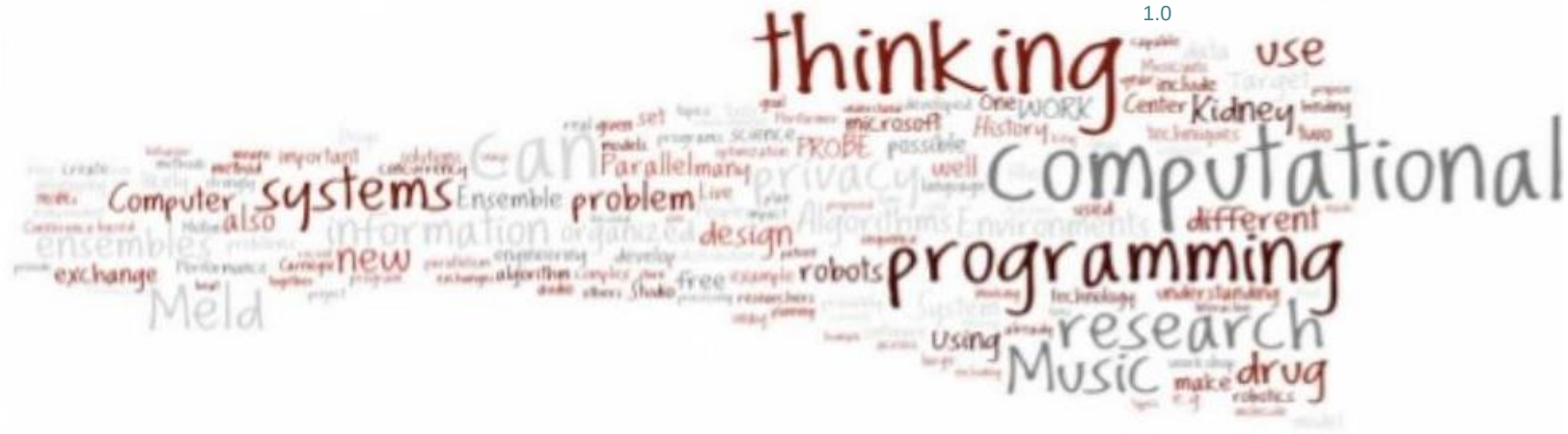
Questo significa che la proposizione $A \leftrightarrow B$ si potrà leggere:

A se, e solo se, B

Condizione necessaria e sufficiente per A è B

Attenzione! A volte, nel linguaggio naturale, interpretiamo le implicazione come se fossero doppie implicazioni.

IL pensiero computazionale



il pensiero computazionale

Il pensiero computazionale è un processo mentale per cui tramite competenze logiche si acquisisce la capacità di risolvere problemi in modo creativo ed efficiente.

Un esempio, tratto dal film Apollo 13

Che cos'è il pensiero computazionale?

- Jeannette Wing, direttrice del Dipartimento di Informati-ca della Carnegie Mellon University, secondo cui "è:
- **il processo mentale che sta alla base della formulazione dei problemi e delle loro soluzioni** così che le soluzioni siano rappresentate in una forma che può essere implementata in maniera efficace da un elaboratore di informazioni sia esso umano o artificiale".
- Ovvero è lo sforzo che un individuo deve mettere in atto per fornire a un altro individuo o macchina tutte e sole le "istruzioni" necessarie affinché questi eseguendole sia in grado di portare a termine il compito dato.

Il pensiero computazionale secondo il framework sviluppato dal Lifelong Kindergarten del MIT MediaLab.

- Concetti di pensiero computazionale:
- **Sequenza**: un'attività può essere espressa attraverso una serie consecutiva di singoli step o istruzioni.
- **Ciclo**: è un meccanismo per eseguire più volte la medesima sequenza in maniera iterativa.
- **Evento**: il verificarsi di un'azione causa lo scatenarsi di un'altra azione.
- **Parallelismo**: significa eseguire sequenze di istruzioni differenti allo stesso tempo.
- **Condizione**: è la possibilità di prendere decisioni sulla base del verificarsi di determinate situazioni.
- **Operatore**: fornisce supporto per la manipolazione di numeri e stringhe di caratteri.
- **Dati**: sono valori che possono essere salvati, recuperati e modificati durante l'esecuzione di un programma.

Pratiche di pensiero computazionale:

- **Essere incrementali e iterativi**: la progettazione è un processo adattativo dove la pianificazione può cambiare man mano che ci si avvicina alla soluzione del problema.
- **Testare e debuggare**: individuare problemi ed errori e correggerli.
- **Riusare** (*pattern recognition*): riconoscere come alcune parti di soluzione possono essere riusate nella stessa o riapplicate a problemi simili.
- **Remixare** (copiare per migliorare): grazie alla rete e all'ampia disponibilità di lavori di altri autori, è possibile prendere spunto da idee e codice per costruire cose più complesse di quelle che si sarebbero potute realizzare per conto proprio, dando un'ulteriore spinta alla propria creatività.
- •**Astrarre**: è il processo di riduzione della complessità, per far emergere l'idea principale mantenendo solo alcuni aspetti e tralasciandone altri.
- **Modularizzare** (scomporre): è il processo che consente di scomporre un problema complesso in problemi più semplici, per cui risolvendo i problemi più semplici si risolve anche il problema complesso.

Attitudini di pensiero computazionale:

- Esprimere se stessi: una persona dotata di pensiero computazionale vede nella tecnologia uno strumento per esprimere se stessi, la propria creatività e dire qualcosa di sé agli altri.
- Essere connessi: saper comunicare e lavorare con gli altri per raggiungere un obiettivo o una soluzione condivisa.
- Porre domande: saper sviluppare una mente vigile grazie alla quale è sempre viva la domanda di come un oggetto incontrato nel mondo reale possa funzionare.

Algoritmi e programmazione strutturata

Programmazione

Calcolatore Elettronico

È uno strumento in grado di eseguire insiemi di azioni elementari;

Le azioni vengono eseguite su oggetti **(*dati*)** per produrre altri oggetti **(*risultati*)**;

L'esecuzione di azioni viene richiesta all'elaboratore attraverso opportune direttive, dette ***istruzioni***.

Programmazione

È l'attività con cui si predispone l'elaboratore ad eseguire un particolare insieme di azioni su particolari dati, allo scopo di risolvere un certo problema.

Problemi da risolvere

I problemi che possono essere risolti con un calcolatore possono essere di varia natura per esempio:

- Sommare due numeri interi;
- Trovare il percorso ottimale per una consegna;
- Stabilire se una parola viene alfabeticamente prima o dopo di un'altra;
- Dati a e b, risolvere l'equazione ax +b = 0
- Dati due numeri trovare il maggiore;
- Gestire acquisti e prestiti dei libri di una biblioteca;
- Trovare gli zeri di una funzione f(x);

Risoluzione dei problemi

Affinché un problema sia risolvibile, in generale è necessario che la sua definizione sia chiara e completa. **Non tutti i problemi sono risolvibili attraverso l'uso del calcolatore**. In particolare esistono classi di problemi per le quali la soluzione automatica non è proponibile. Ad esempio:

- Se il problema presenta infinite soluzioni;
- Se per il problema non stato trovato un metodo risolutivo (Problema di Fermat: trovare tutti gli N per cui l'equazione $x^N+y^N=z^N$ sia soddisfatta);
- Non esiste un metodo risolutivo automatizzabile.

Analisi e programmazione

Tramite un elaboratore si possono risolvere problemi di varia natura. Il problema deve essere formulato in modo opportuno, perché sia possibile utilizzare un calcolatore per la sua soluzione

- Per **analisi e programmazione** si intende l'insieme delle attività preliminari atte a risolvere problemi utilizzando un elaboratore, dalla formulazione del problema fino alla predisposizione dell'elaboratore.
- **Scopo dell'analisi**: definire un algoritmo
- **Scopo della programmazione**: definire un programma

Risoluzione di problemi

- **Algoritmo**: elenco finito di istruzioni, che specificano le operazioni, eseguendo le quali si risolve una classe di problemi;
- **Programma**: ricetta che traduce l'algoritmo ed è direttamente comprensibile, pertanto eseguibile, da parte di un elaboratore;
- **Linguaggio di programmazione**: linguaggio rigoroso che permette la formalizzazione di un algoritmo in un programma (es: C, C++, Basic, Java ...);

Fasi di risoluzione del problema

Proprietà di un algoritmo

- Affinchè un elenco di istruzioni possa essere considerato un algoritmo devono essere soddisfatti tali requisiti:
- **Finitezza**: ogni algoritmo deve essere finito, cioè ogni singola istruzione deve poter essere eseguita in tempo finito e numero finito di volte;
- **Eseguibilità**: ogni istruzione dell'algoritmo deve essere eseguibile da parte dell'esecutore dell'algoritmo;
- **Non ambiguità**: devono essere definiti in modo univoco i passi successivi da seguire; devono essere evitati paradossi, contraddizioni ed ambiguità; il significato di ogni istruzione deve essere univoco per chiunque esegua l'algoritmo.

Proprietà di un algoritmo

Gli algoritmi devono essere formalizzati per mezzo di appositi linguaggi, dotati di strutture linguistiche che garantiscano precisione e sintesi;

I linguaggi naturali non soddisfano tali requisiti, infatti sono **ambigui**: la stessa parola può assumere significati diversi in contesti differenti (*pesca* è un frutto e uno sport) … sono **ridondanti**: lo stesso concetto può essere espresso in modi diversi, ad esempio

"somma 2 a 3", "calcola 2+3", "esegui l'addizione tra 2 e 3"

Proprietà di un algoritmo

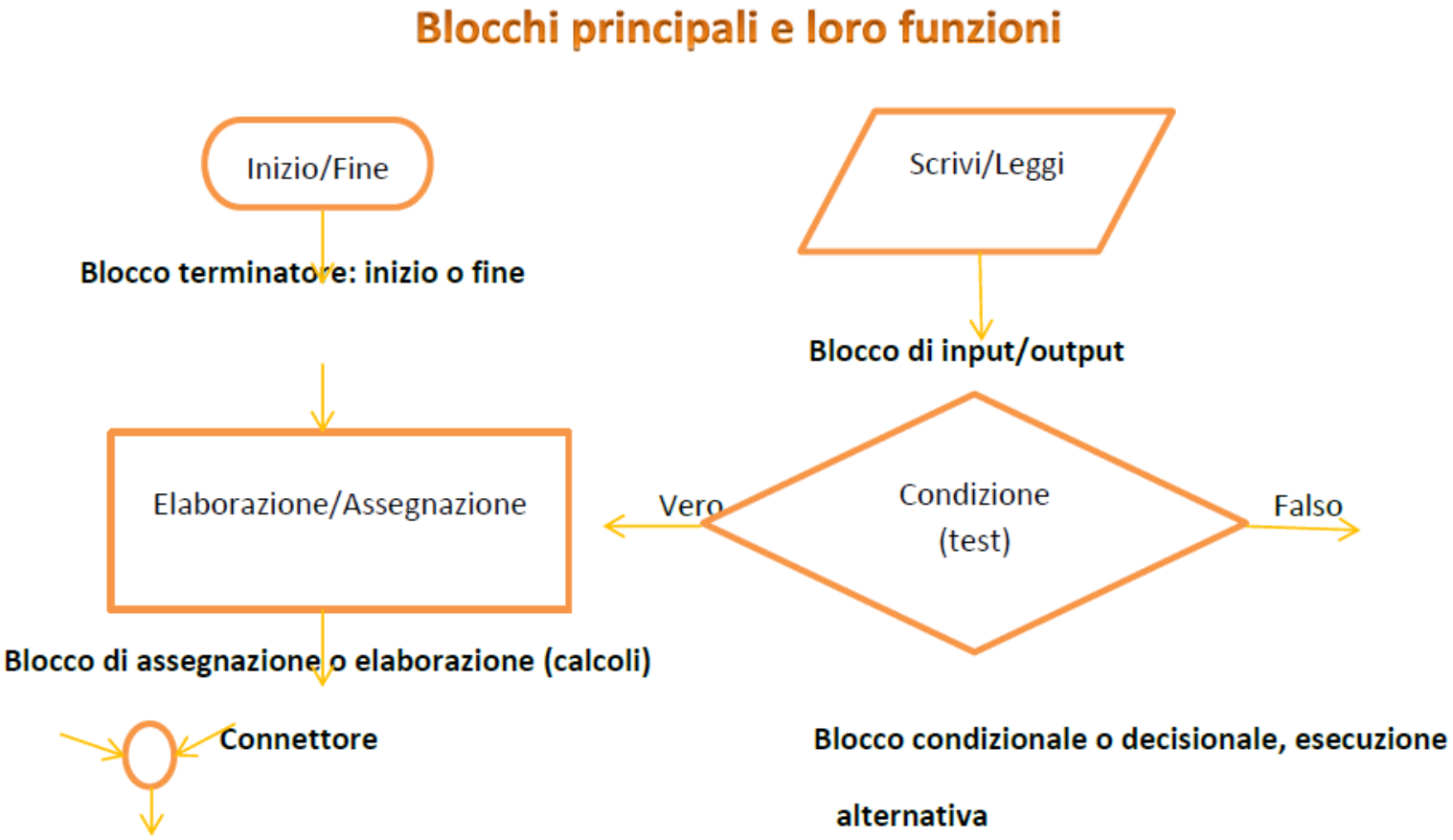
Un algoritmo può essere visto come un procedimento di calcolo (non necessariamente calcolo aritmetico), dal punto di vista sistemistico l'algoritmo a partire da un input fornisce un output.

Rappresentazione di algoritmi: diagrammi di flusso

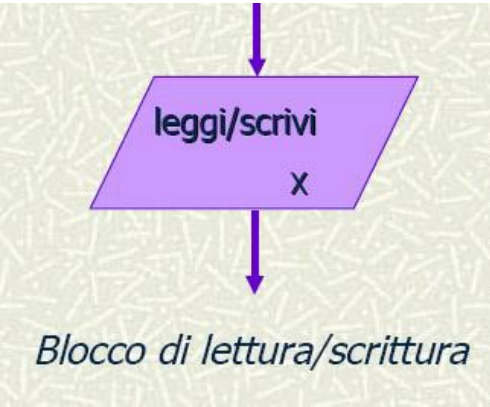
E' un formalismo che consente di rappresentare graficamente gli algoritmi.

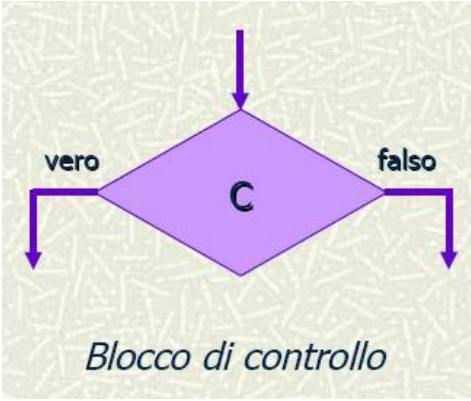
- un **diagramma a blocchi** descrive le azioni da eseguire e il loro ordine di esecuzione.
- Il diagramma a blocchi o flowchart è una rappresentazione grafica che descrive il flusso delle operazioni da eseguire per realizzare la trasformazione, definita nell'algoritmo, dai dati iniziali ai risultati.
- Ogni istruzione dell'algoritmo viene rappresentata all'interno di un blocco elementare, la cui forma grafica è determinata dal tipo di istruzione.
- I blocchi sono collegati tra loro da linee di flusso, munite di frecce, che indicano il susseguirsi di azioni elementari.

Rappresentazione di algoritmi: diagrammi di flusso



Rappresentazione di algoritmi: diagrammi di flusso



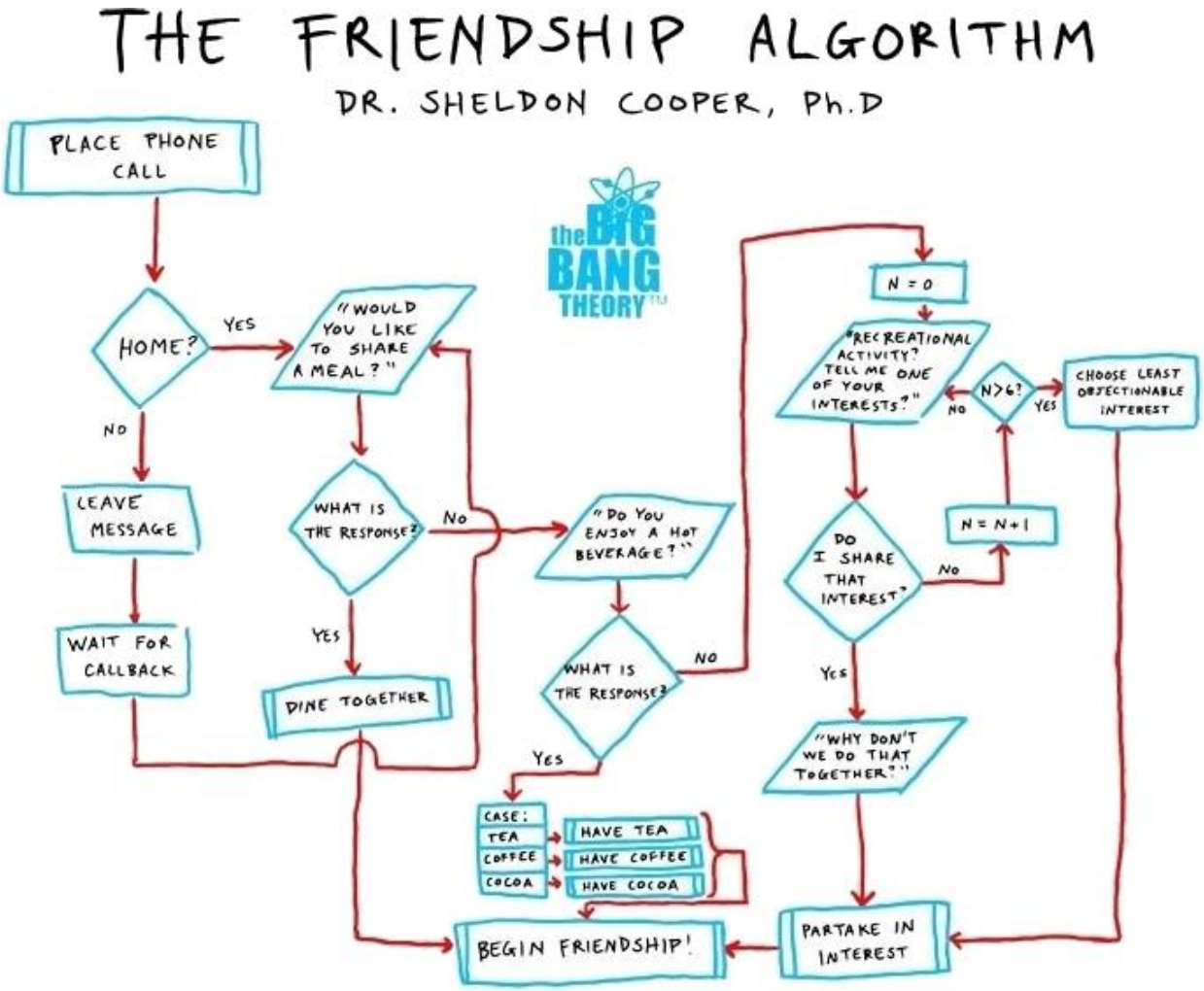


- ciascun blocco di azione o di lettura/scrittura ha una sola freccia entrante ed una sola freccia uscente.
- ciascun blocco di controllo ha una sola freccia entrante e due frecce uscenti.
- ciascuna freccia entra in un blocco oppure si innesta in un'altra freccia.
- ciascun blocco è *raggiungibile* dal blocco iniziale.
- il blocco finale è *raggiungibile* da qualsiasi altro blocco.

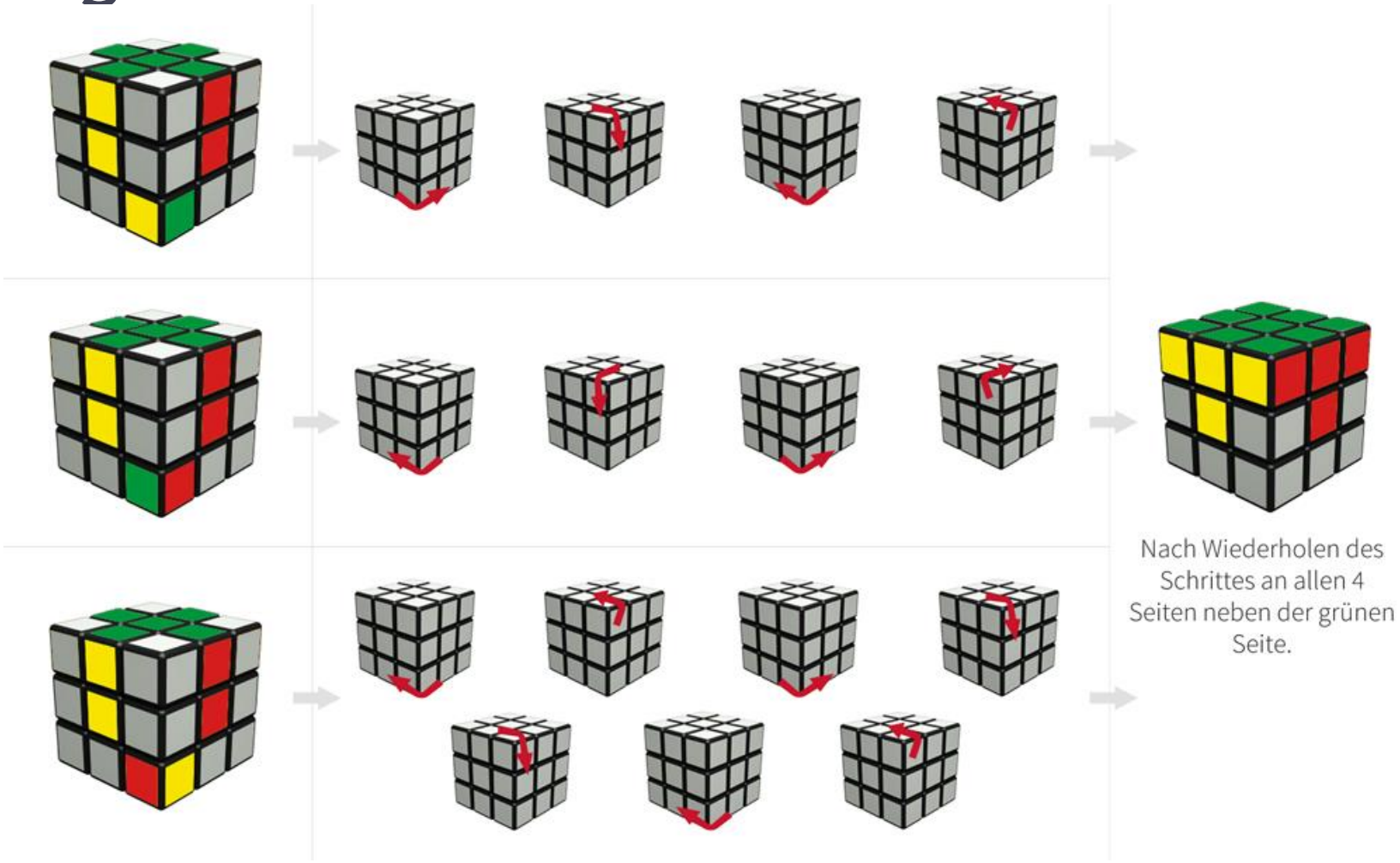
Algoritmi: un esempio non numerico

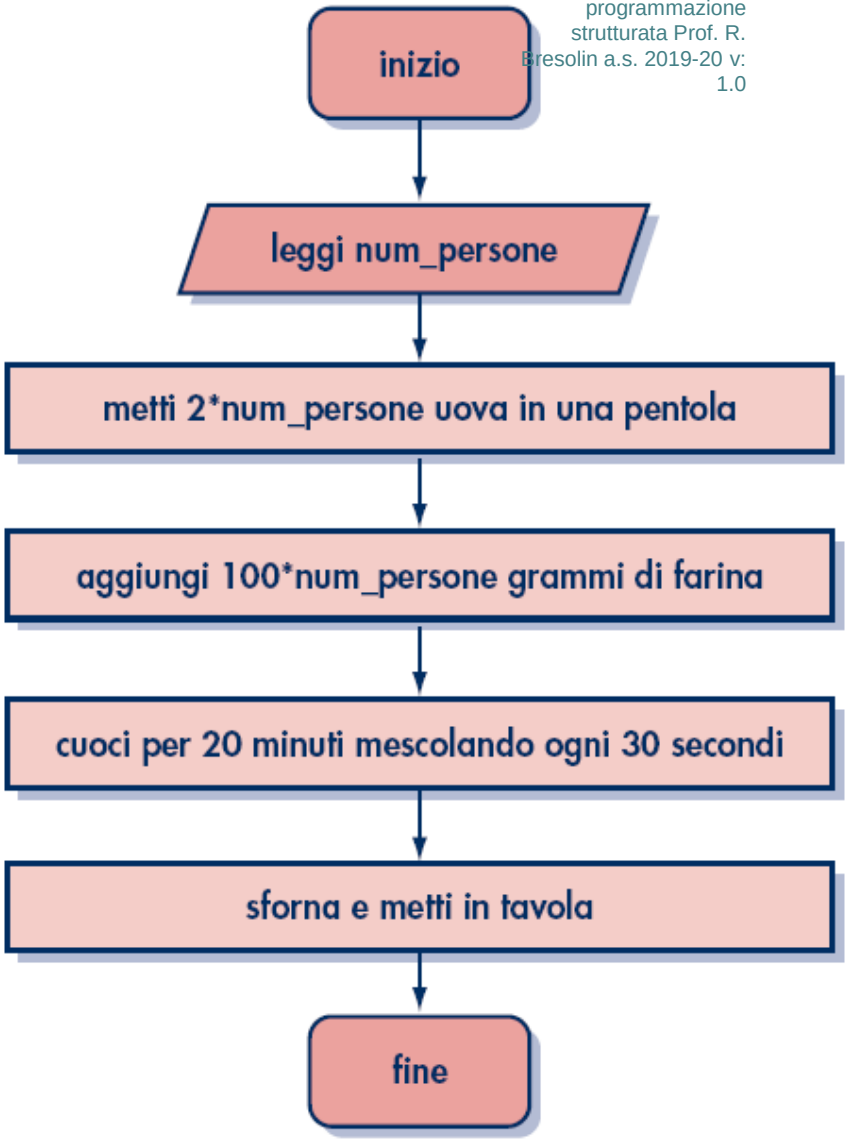
- Problema: preparare una torta (?!?)
- Dato iniziale: numero di persone
- Dato finale: la torta pronta
- Esecutore: in grado di comprendere i diagrammi a blocchi e di effettuare le operazioni elementari di cucina (mescolare, cuocere …)

Algoritmi non numerici



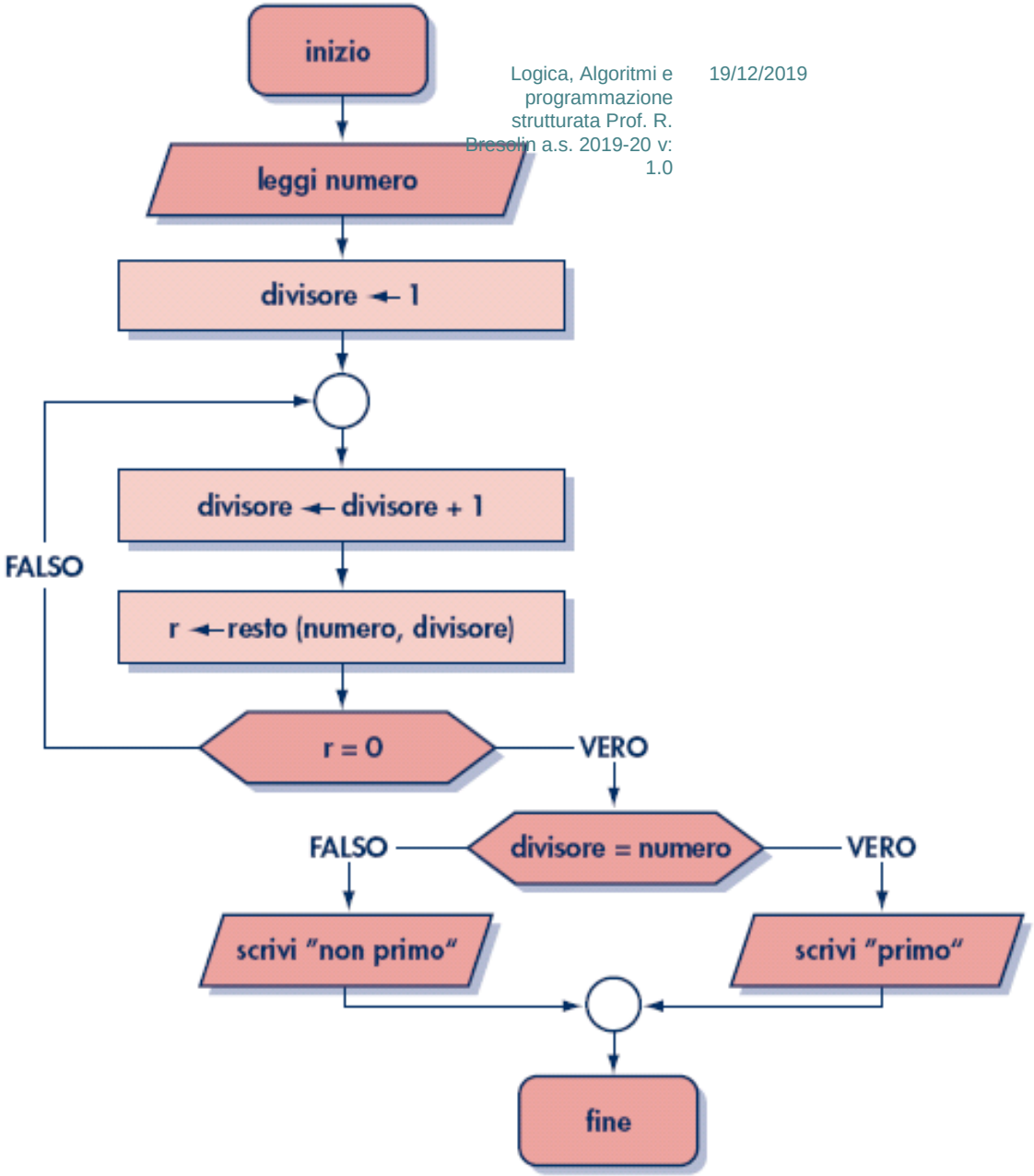
Algoritmi non numerici




inizio
leggi num_persone
metti 2*num_persone uova in una pentola
aggiungi 100*num_persone grammi di farina
cuoci per 20 minuti mescolando ogni 30 secondi
sforna e metti in tavola
fine

Algoritmi: un esempio numerico

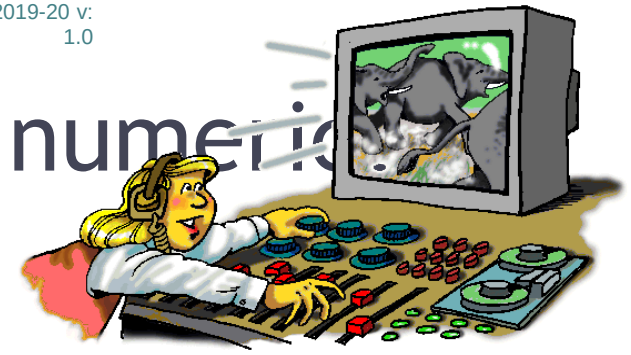
- Problema: verificare se un numero è primo
- Dato iniziale: il numero intero positivo
- Dato finale: “primo” o “non primo”
- Esecutore: in grado di comprendere i diagrammi a blocchi e di effettuare le operazioni matematica compresa il resto della divisione intera resto(n,d)


inizio
leggi numero
divisore ← 1
divisore ← divisore + 1
FALSO
r ← resto (numero, divisore)
r = 0
VERO
FALSO
divisore = numero
VERO
scrivi "non primo"
scrivi "primo"
fine

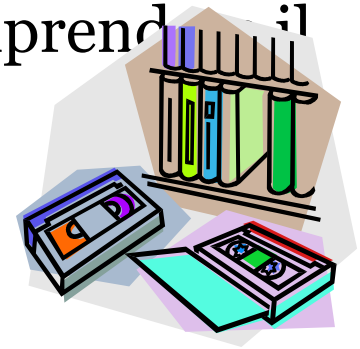
Verifica: esempio

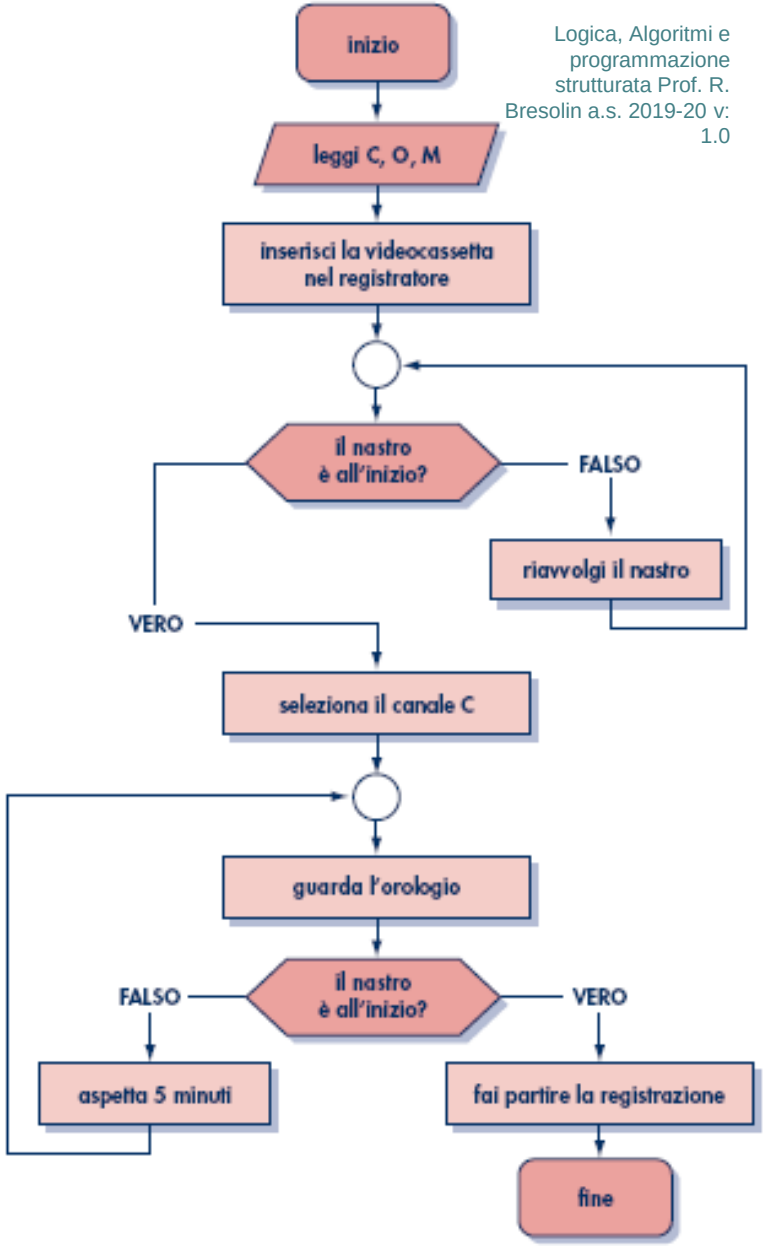
	numero	divisore	r	input	output
passo1	15	–	–	15	–
passo2	–	1	–	–	–
passo3	–	2	–	–	–
passo4	–	–	1	–	–
passo5	–	3	–	–	–
passo6	–	–	0	–	–
passo7	–	–	–	–	non primo

Algoritmi: un esempio non numerico



- Problema: insegnare ad un bambino ad utilizzare un videoregistratore per registrare un film.
- Dati iniziali: Canale, Ora e Minuto di inizio film
- Dato finale: Film registrato su videocassetta
- Esecutore (bambino) è in grado di comprendere i diagrammi a blocchi e di eseguire alcune operazioni elementari (prendere una videocassetta, comprendere il significato dei tasti del videoregistratore …)

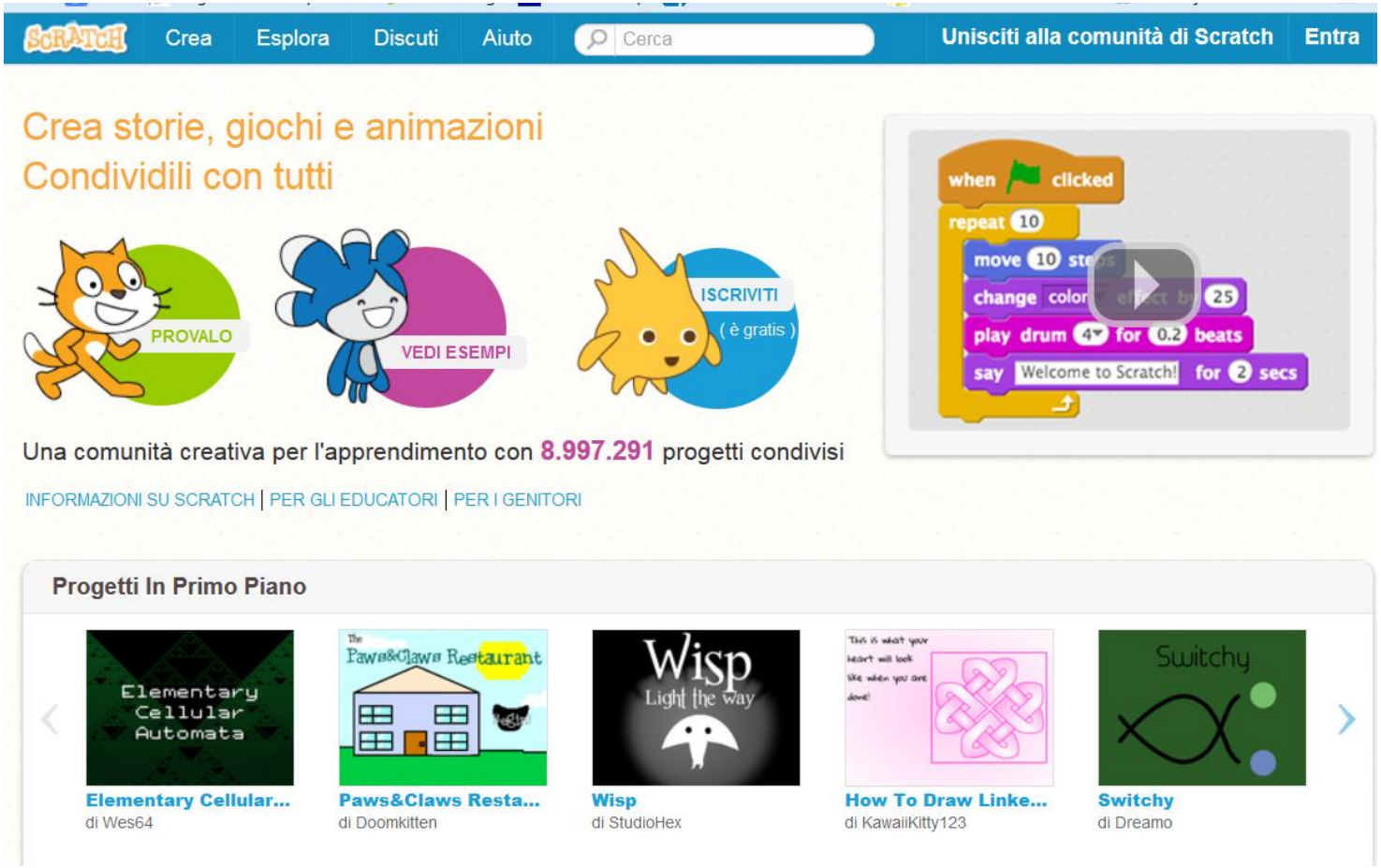



inizio
leggi C, O, M
inserisci la videocassetta nel registratore
il nastro è all'inizio?
FALSO
riavvolgi il nastro
VERO
seleziona il canale C
guarda l'orologio
il nastro è all'inizio?
FALSO
VERO
aspetta 5 minuti
fai partire la registrazione
fine

Diagrammi a blocchi: pregi e difetti

- Pregi:
  - Semplicità
  - Possibilità di seguire facilmente il flusso di esecuzione
- Difetti:
  - Per algoritmi complessi si ottiene una struttura molto complessa e risulta difficile decifrare il procedimento seguito nella risoluzione

BYOB ambiente integrato per creare programmi in modo divertente



AlgoBuild Flow Chart & Programs Designer

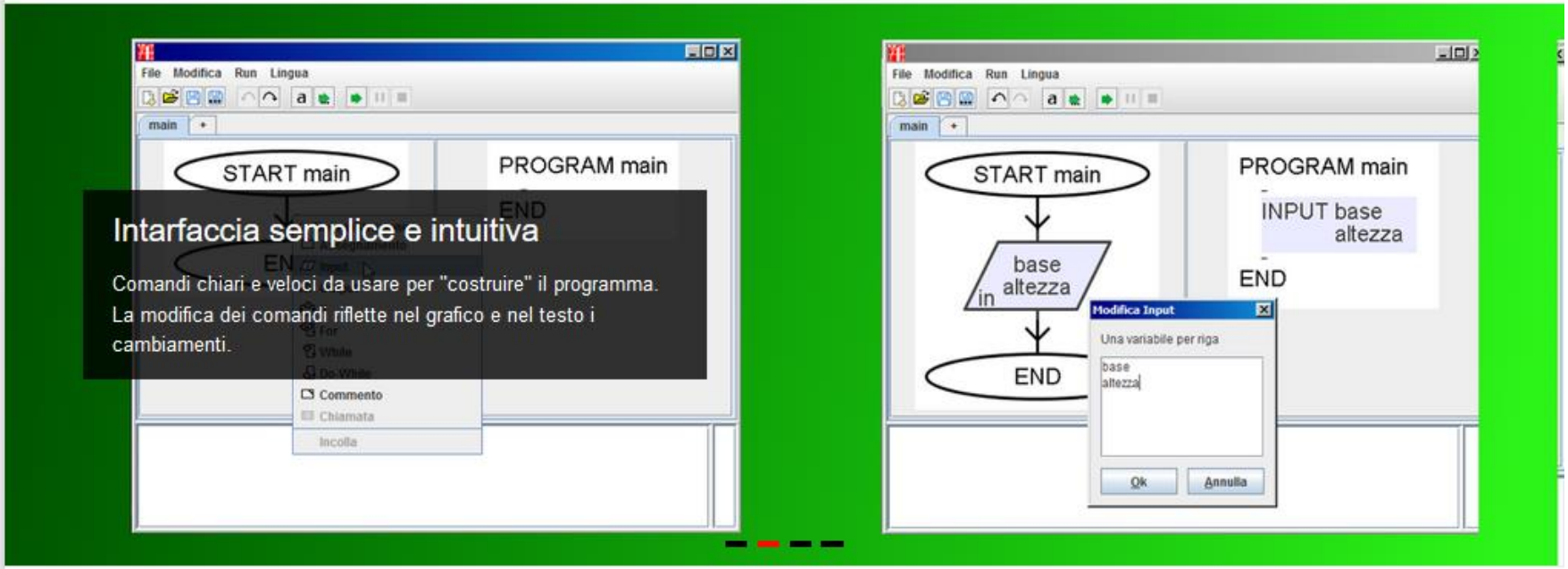
- Disegna diagrammi di flusso (Flow Chart) con AlgoBuild, il programma didattico per lo studio della programmazione e degli algoritmi.
- Semplice e potente ambiente per la programmazione strutturata con diagrammi di flusso (flowchart) e pseudo codice (pseudocode).
- E' Divertente e facile da usare ma basato su una sintassi formale grafica strutturata.

AlgoBuild Flow Chart & Programs Designer

- UN programma gratuito
- https://algobuild.com/it/download.html
- Con il suo manuale ed esempi
- https://algobuild.com/docsonline/ab_075/manuale-it/manuale-it.html
- Per funzionare ha bisogno di Java
- https://www.java.com/it/download/

AlgoBuild

Algoritmi & FlowChart



Crea Diagrammi di Flusso, Pseudo Codice, Programmi

Partenza... VIA!

Disegna i **programmi** con **AlgoBuild**, l'ambiente didattico per lo studio della programmazione e degli algoritmi.
Semplice e veloce permette di apprendere le nozioni base della **programmazione strutturata** per mezzo di **diagrammi di flusso** (flowchart) e **pseudo codice** (pseudocode).

Crea Diagrammi di Flusso, Pseudo Codice, Programmi

Partenza... VIA!

Disegna i **programmi** con **AlgoBuild**, l'ambiente didattico per lo studio della programmazione e degli algoritmi.
Semplice e veloce permette di apprendere le nozioni base della **programmazione strutturata** per mezzo di **diagrammi di flusso** (flowchart) e **pseudo codice** (pseudocode).
E' Divertente e facile da usare ma basato su una sintassi formale grafica strutturata.
Ottimizza il lavoro in laboratorio, in classe con la Lavagna Interattiva Multimediale e a casa. Guarda l'esempio Ciao mondo! con diagramma di flusso e pseudocodifica (flowchart and pseudocode).

Download

ULTIMO AGGIORNMENTO 11 gennaio 2016
(Build 00080_20160111_2300)

Qui puoi trovare il programma AlgoBuild, gli esempi e i manuali da scaricare.

Vai alla sezione Download

Tutorial

Se vuoi avere le informazioni di base per poter cominciare a lavorare con AlgoBuild, qui puoi trovarle.

Vai alla sezione Tutorial

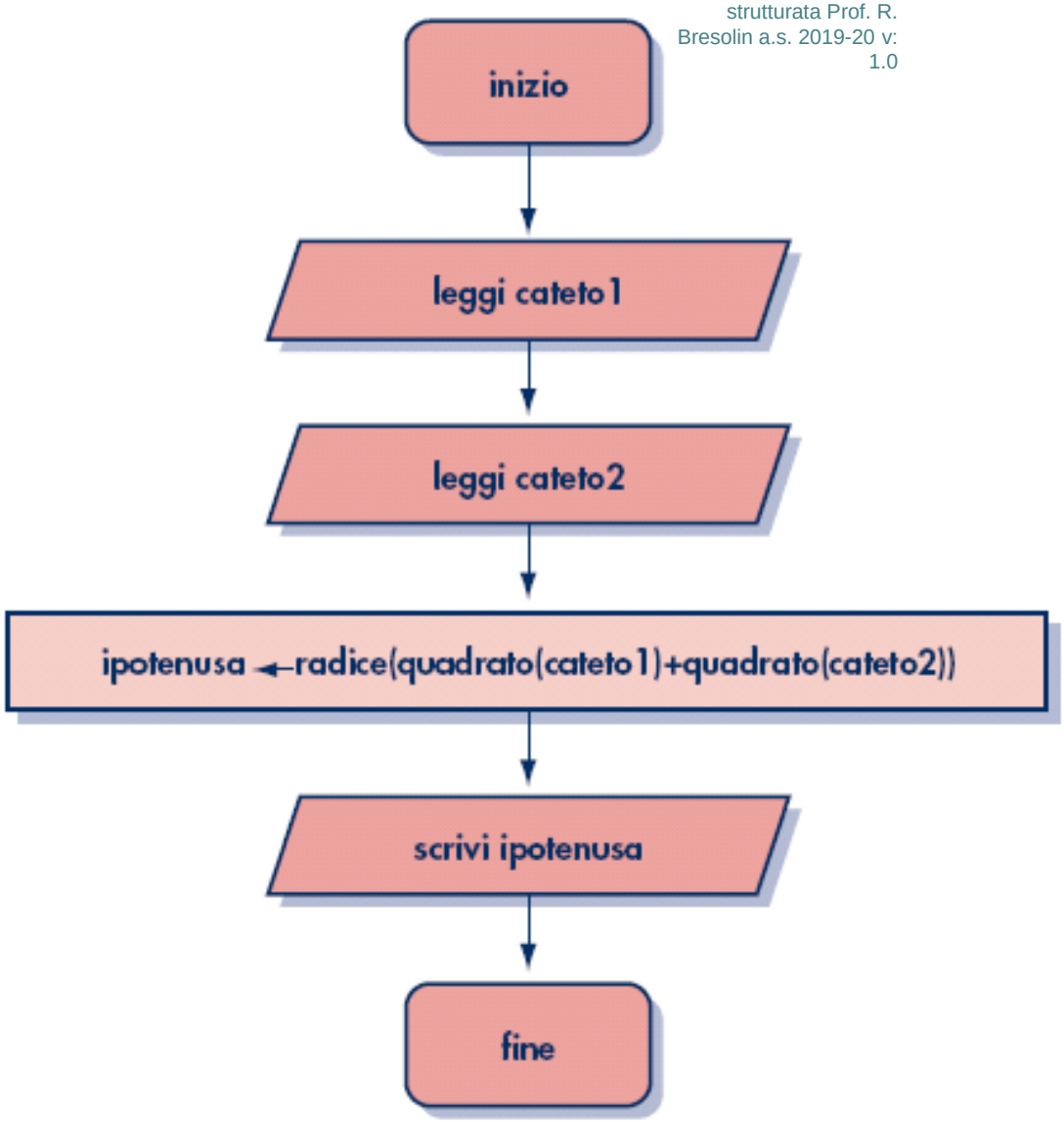
Manuale on line

Il manuale presenta in modo sintetico le istruzioni per l'installazione, l'elenco degli operatori e delle funzioni incorporate nel programma.

Vai alla sezione Manuale on line

Algoritmi: un esempio numerico

- Problema: date le misure dei cateti trovare la misura dell'ipotenusa in un triangolo rettangolo
- Dati iniziali: misura del cateto1 e del cateto2
- Dato finale: misura dell'ipotenusa
- Esecutore: in grado di comprendere i diagrammi a blocchi e di effettuare le operazioni sui numeri reali compreso elevamento al quadrato – quadrato(x) – calcolo della radice quadrata – radice(x) -


inizio
leggi cateto1
leggi cateto2
ipotenusa ←radice(quadrato(cateto1)+quadrato(cateto2))
scrivi ipotenusa
fine

Programmazione strutturata

La *programmazione strutturata* favorisce la descrizione di algoritmi facilmente documentabili e comprensibili.

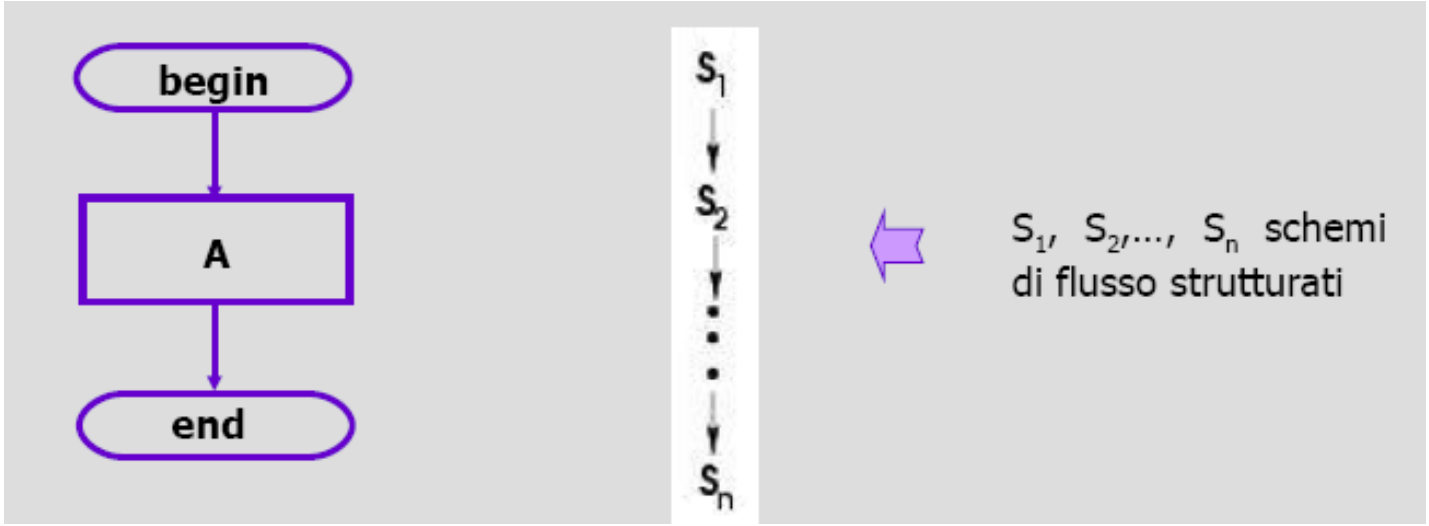
I blocchi di un diagramma a blocchi strutturato sono collegati secondo i seguenti schemi di flusso:

- ***Schema di sequenza*** – più schemi di flusso sono eseguiti in sequenza.
- ***Schema di selezione*** – un blocco di controllo subordina l'esecuzione di due possibili schemi di flusso al verificarsi di una condizione.
- ***Schema di iterazione*** – si itera l'esecuzione di un dato schema di flusso.

Programmazione strutturata

Quindi un diagramma a blocchi strutturato è un diagramma a blocchi nel quale gli schemi di flusso sono strutturati.

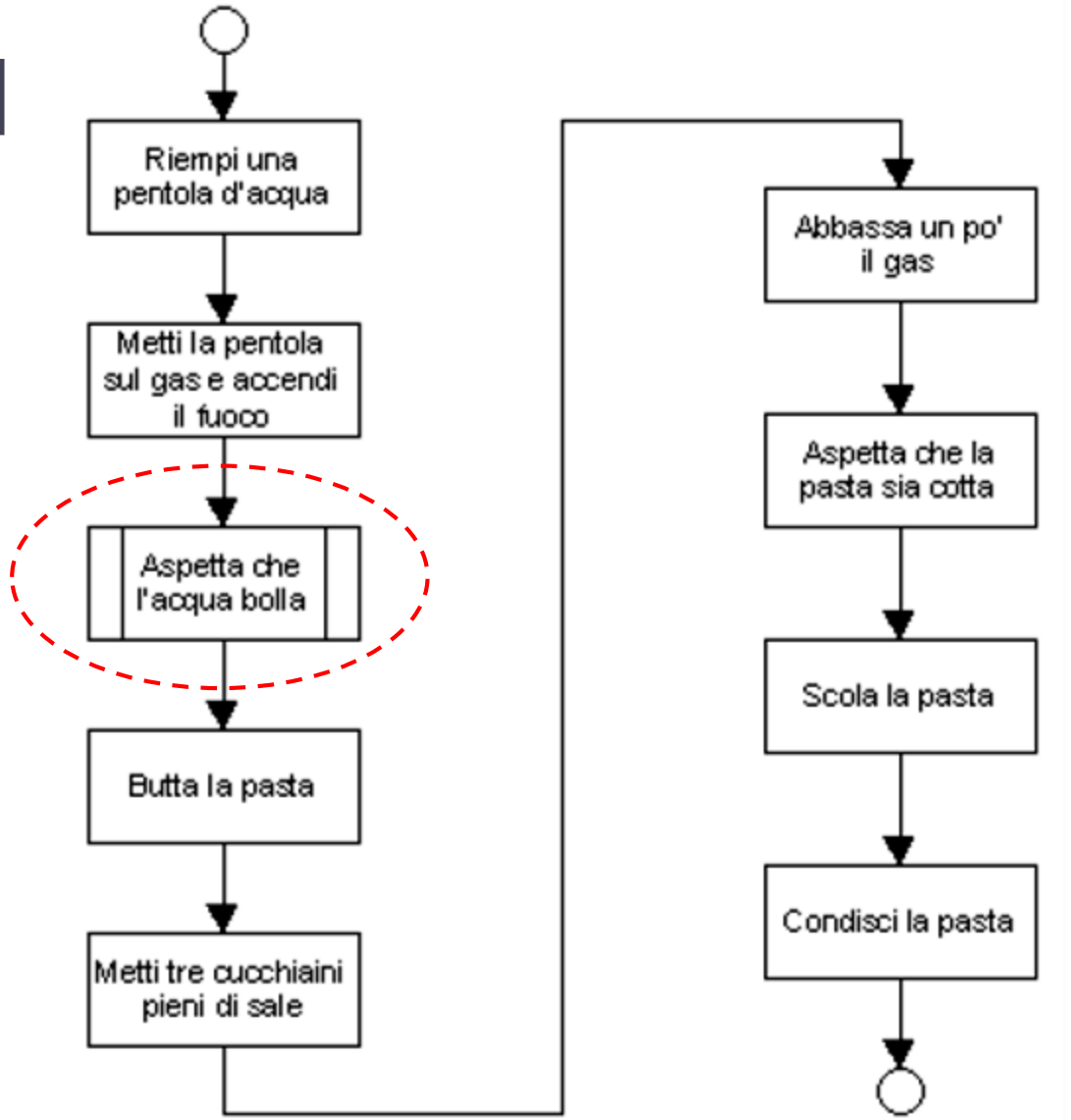
- *Lo schema di sequenza* è una sequenza di passi eseguiti uno alla volta, nessun passo è ripetuto e l'ordine di esecuzione dei passi è lo stesso in cui sono scritti.



Esempio seq

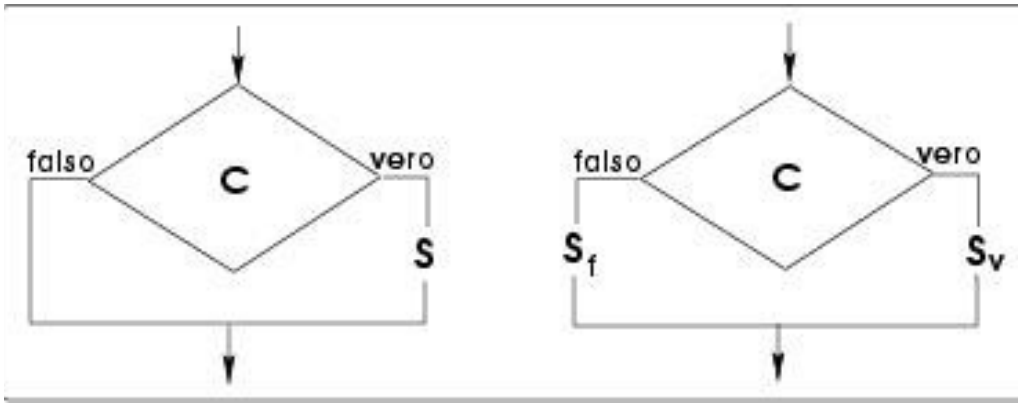
Esercizio 1

Si disegni un flow chart che rappresenta il flusso di un programma per un robot che deve far cuocere la pasta asciutta, procedendo per macroistruzioni.



Programmazione strutturata

Lo ***schema di selezione*** è usato quando si deve effettuare una scelta tra due passi alternativi.

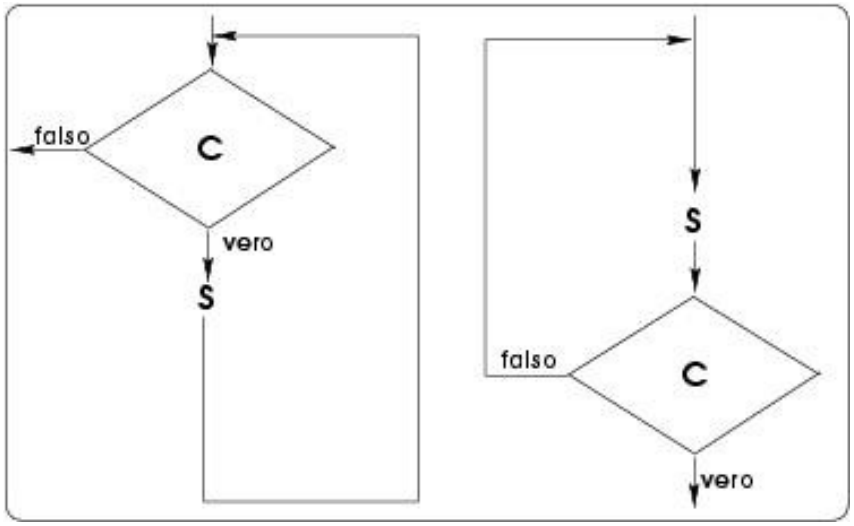


Nel primo caso, lo schema S viene eseguito solo se la condizione C è vera; se C è falsa non viene eseguita alcuna azione.

Nel secondo caso, viene eseguito solo uno dei due schemi S_v o S_f in dipendenza del valore di verità della condizione C.

Programmazione strutturata

Lo ***schema di iterazione*** permette la ripetizione di certi passi un numero arbitrario o fisso di volte.

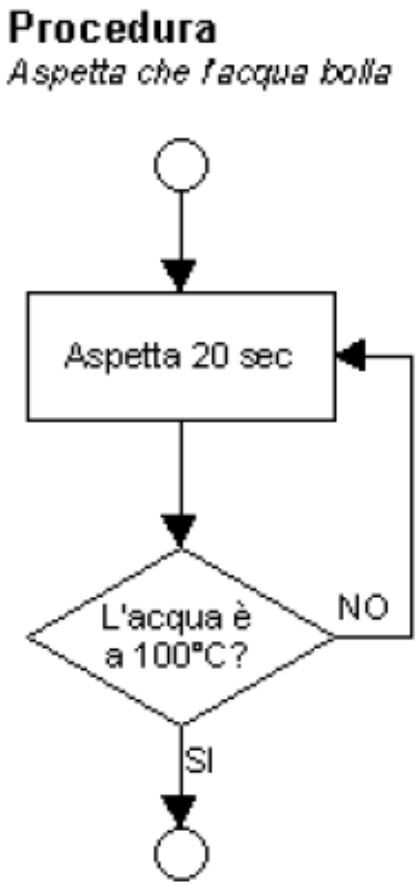


Nel primo caso, S può non venire mai eseguito se la condizione C è subito falsa;
nel secondo caso S viene eseguito almeno una volta.

Esempio di ciclo

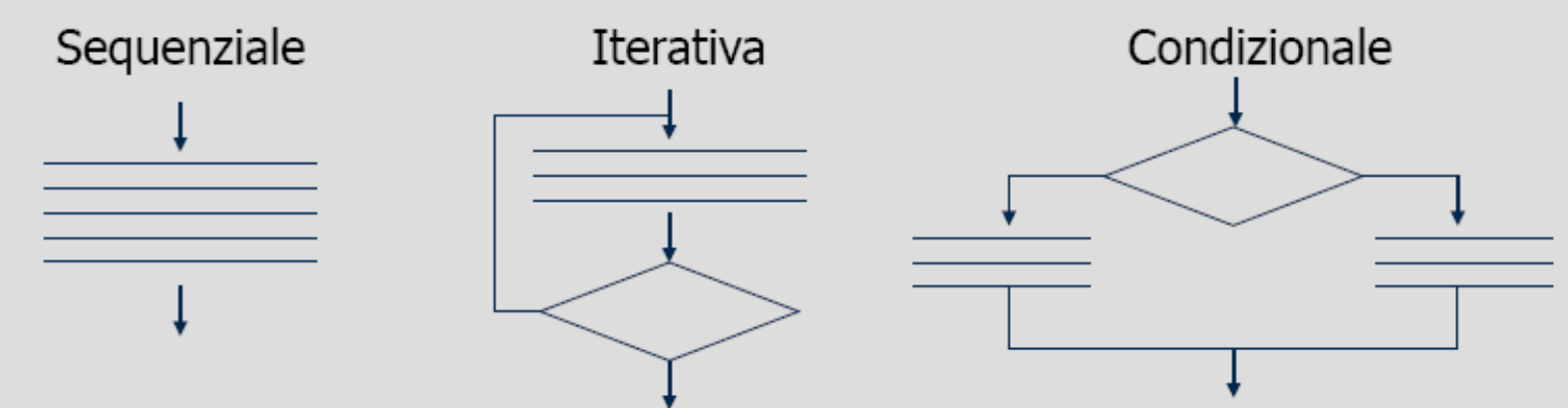
Tornando all'esempio 1 del robot che deve essere programmato a cuocere la pasta abbiamo indicato nella sequenza delle istruzioni genericamente la procedura "aspetta che l'acqua bolla".

Volendo dettagliare tale procedura dovremmo usare un ciclo in cui l'istruzione " aspetta 20 s" viene eseguita almeno una volta e poi si esegue il controllo per finire il ciclo.



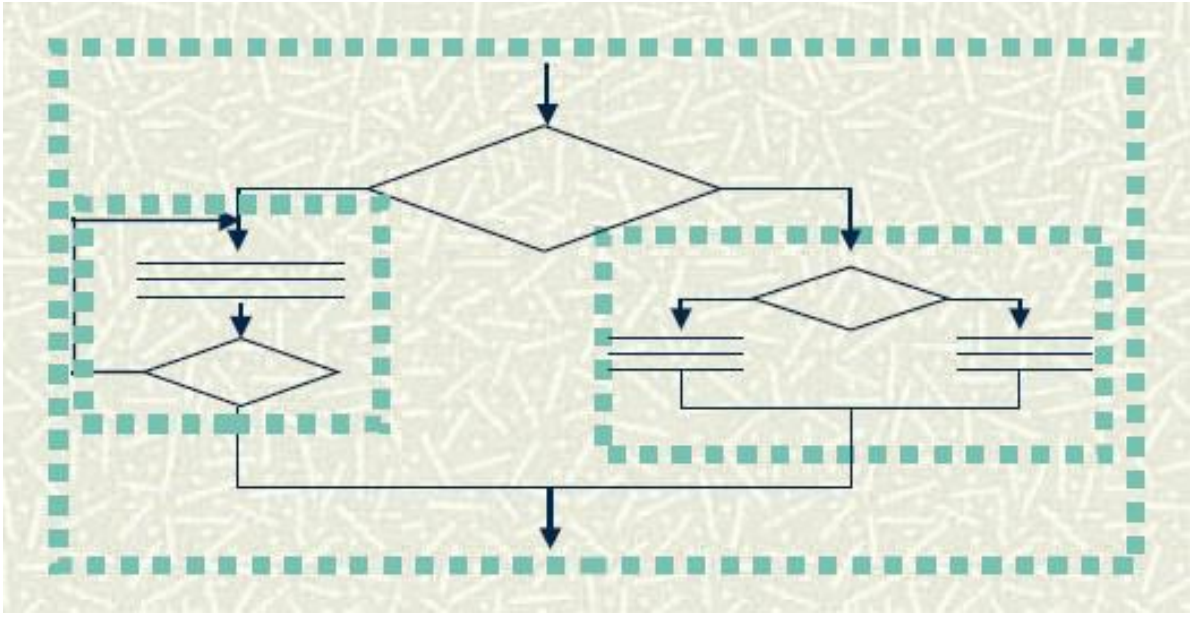
Rappresentazione di algoritmi: diagrammi di flusso

- **È stato dimostrato** (teorema fondamentale della programmazione di Bohm-Jacopini, 1966) **che ogni programma può essere codificato riferendosi esclusivamente ad un algoritmo strutturato** e quindi attenendosi alle tre strutture fondamentali:



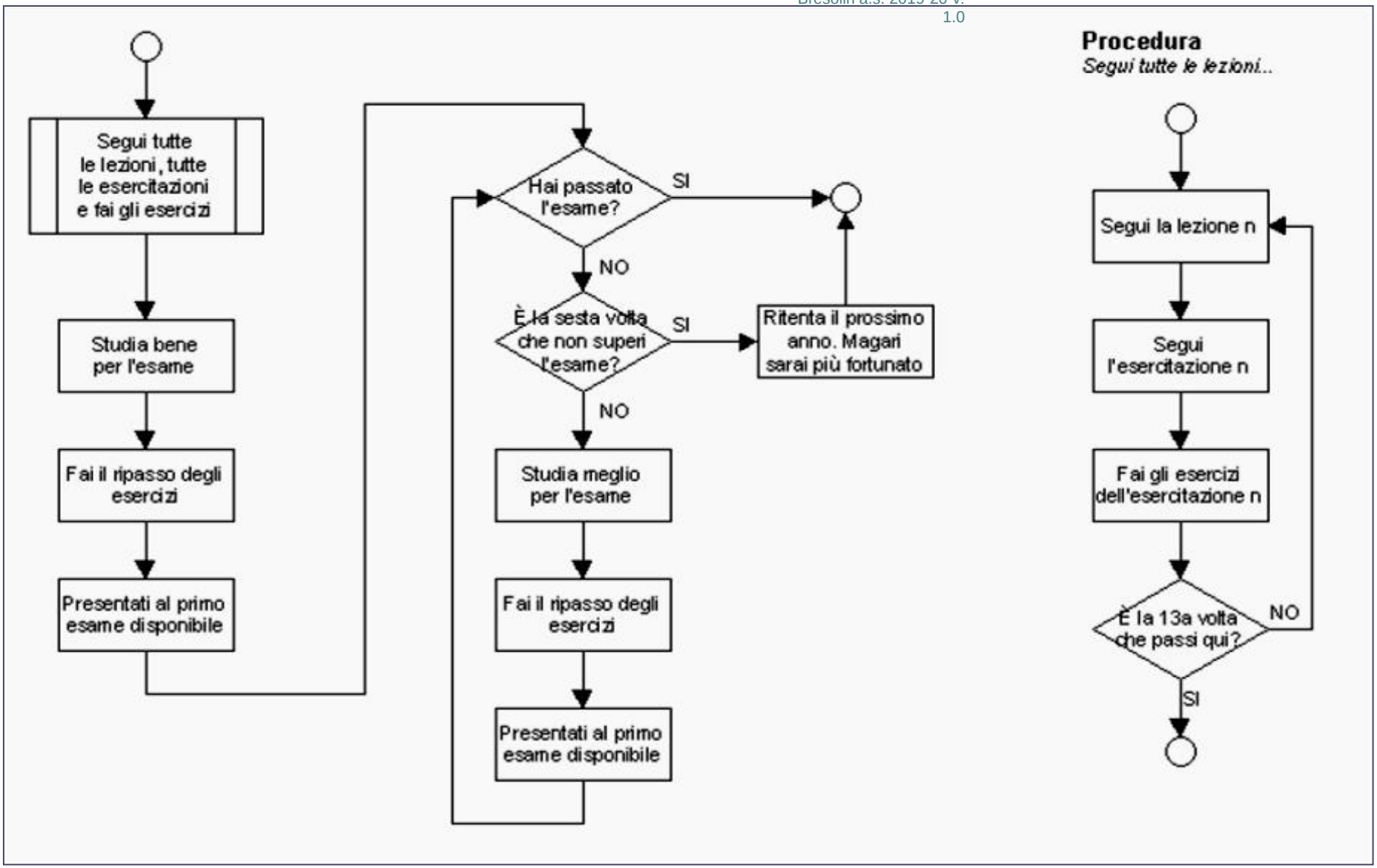
Programmazione strutturata

In un diagramma strutturato non apparirà mai una istruzione di salto incondizionato; I tre schemi fondamentali possono essere ***concatenati***, uno di seguito all'altro, o ***nidificati***, uno dentro l'altro; non possono in nessun caso essere "intrecciati" o "accavallati".



Esempio 2

Disegnare il Flow Chart che simula le operazioni da compiere, per uno studente modello, per superare un esame universitario, coprendo l'arco temporale che va dall'inizio del corso al superamento dell'esame.


Segui tutte le lezioni, tutte le esercitazioni e fai gli esercizi
Studia bene per l'esame
Fai il ripasso degli esercizi
Presentati al primo esame disponibile
Hai passato l'esame?
SI
NO
È la sesta volta che non superi l'esame?
SI
Ritenta il prossimo anno. Magari sarai più fortunato
NO
Studia meglio per l'esame
Fai il ripasso degli esercizi
Presentati al primo esame disponibile
Procedura
Segui tutte le lezioni..
Segui la lezione n
Segui l'esercitazione n
Fai gli esercizi dell'esercitazione n
È la 13a volta che passi qui?
NO
SI

Altri esercizi

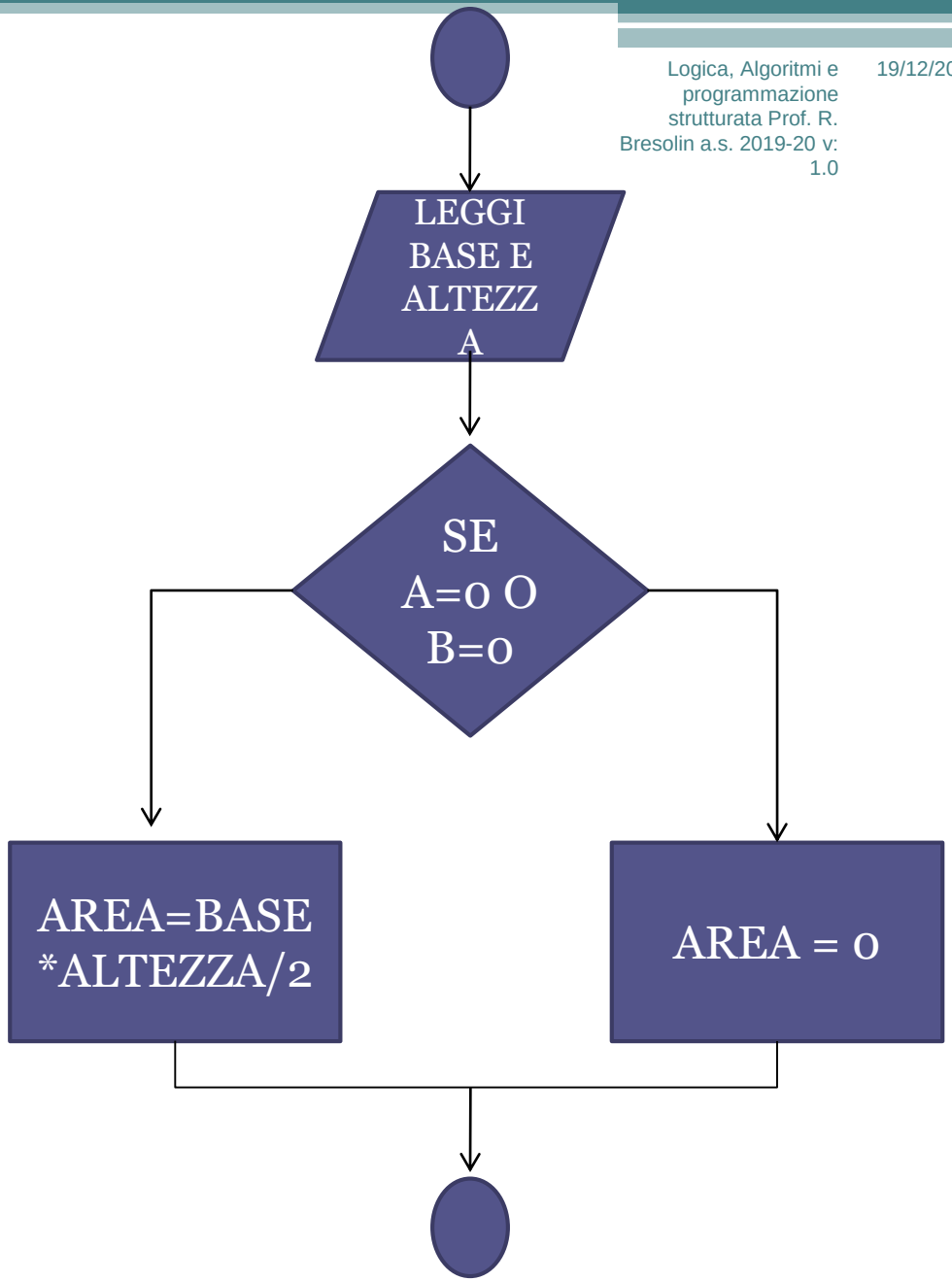
- **Esercizio 3**

Fare il diagramma di flusso con le operazioni da compiere per calcolare area e perimetro di un triangolo isoscele, supponendo che gli input possano essere:

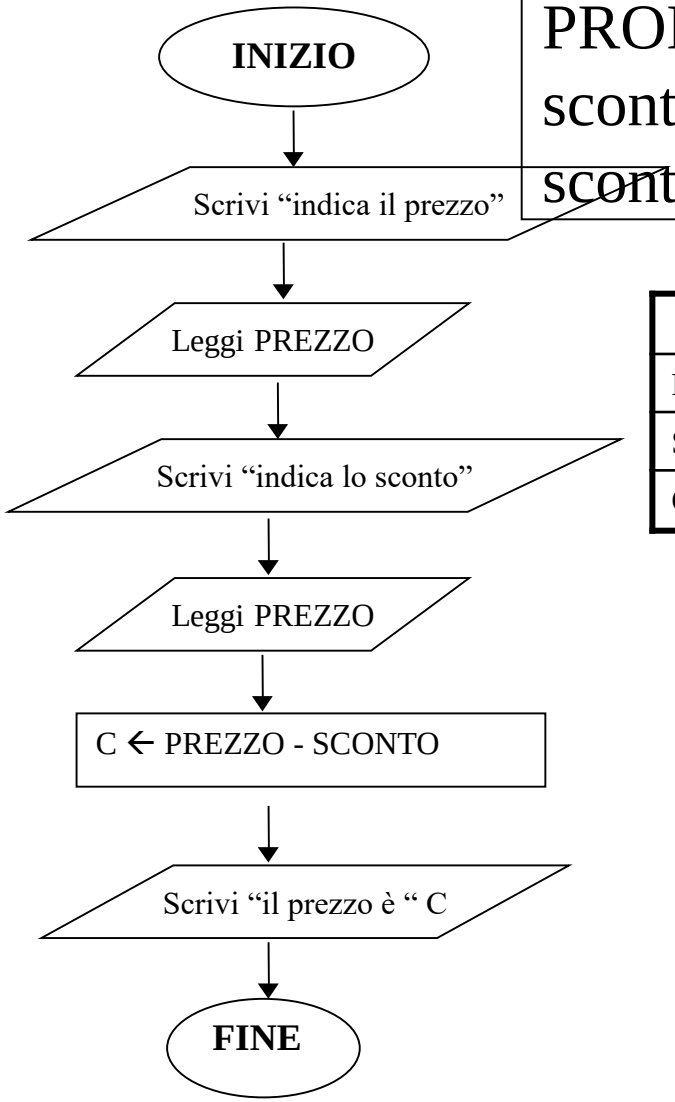
  - lunghezza della base e dell'altezza

- **Esercizio 4**

Realizzare un diagramma di flusso che, date le dimensioni del pavimento in una stanza rettangolare, e dato il lato di una piastrella quadrata, trovi quante sono le piastrelle da utilizzare per la pavimentazione.

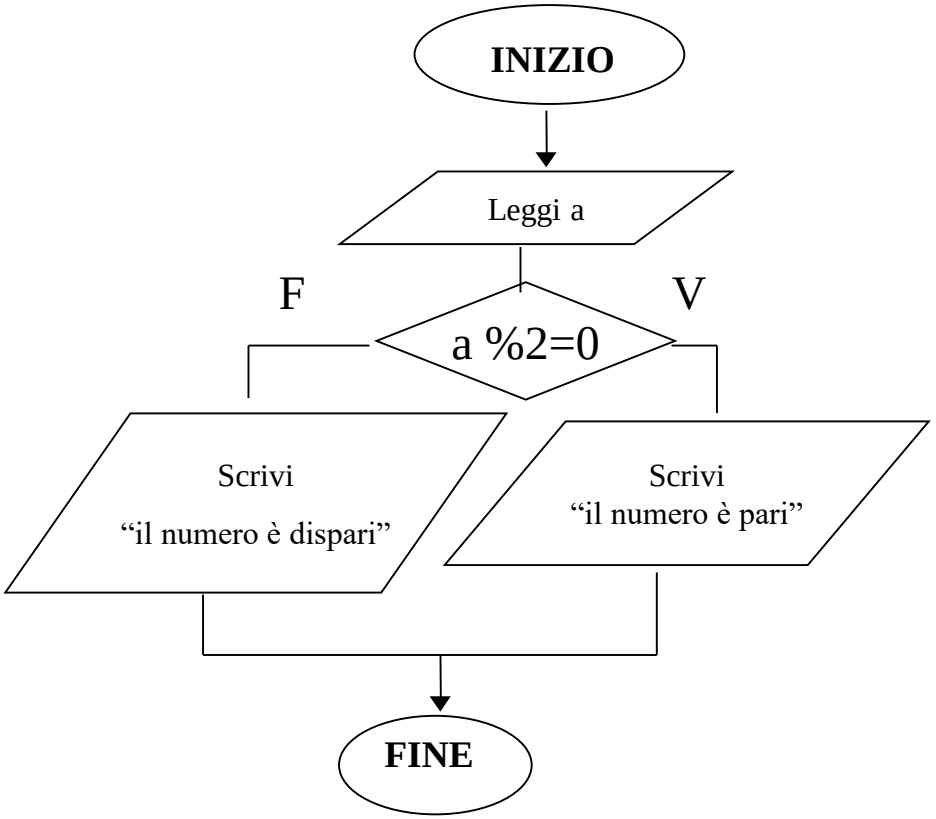

LEGGI BASE E ALTEZZA
SE A=0 O B=0
AREA=BASE *ALTEZZA/2
AREA = 0

PROBLEMA: dato il prezzo di un prodotto e lo sconto effettuato, calcolare e comunicare il prezzo scontato



nome	I/O	V/C	N/A	descrizione
PREZZO	I	V	N decimale	Prezzo del prodotto
SCONTO	I	V	N decimale	Sconto effettuato
C	O	V	N decimale	Prezzo scontato

PROBLEMA: dato un numero si comunichi se il numero è pari o dispari



Bibliografia-Sitografia

- Algoritmi + Strutture Dati = Programmi. Niklaus Wirth Ed: tecniche nuove
- *Introduzione agli algotmi. T.H. Cormen-C.E. Leirson-R.L. Rivest Ed Jackson Libri*
- *Tecnologie informatiche Office 2010, Camagni Nicolassi , Hoeply*
- *Algoritmi. Apogeo Libri*