UNIVERSITÀ
degli STUDI
di CATANIA

Problemi, algoritmi, diagrammi di flusso

Corso di programmazione I (A-E / O-Z) AA 2025/26

Corso di Laurea Triennale in Informatica

Fabrizio Messina
fabrizio.messina@unict.it

Dipartimento di Matematica e Informatica

Overview

Algoritmi

Algoritmo

Dato un problema, un **algoritmo** è una sequenza di passi concepita per essere eseguita automaticamente da una macchina in modo da risolvere il problema dato.

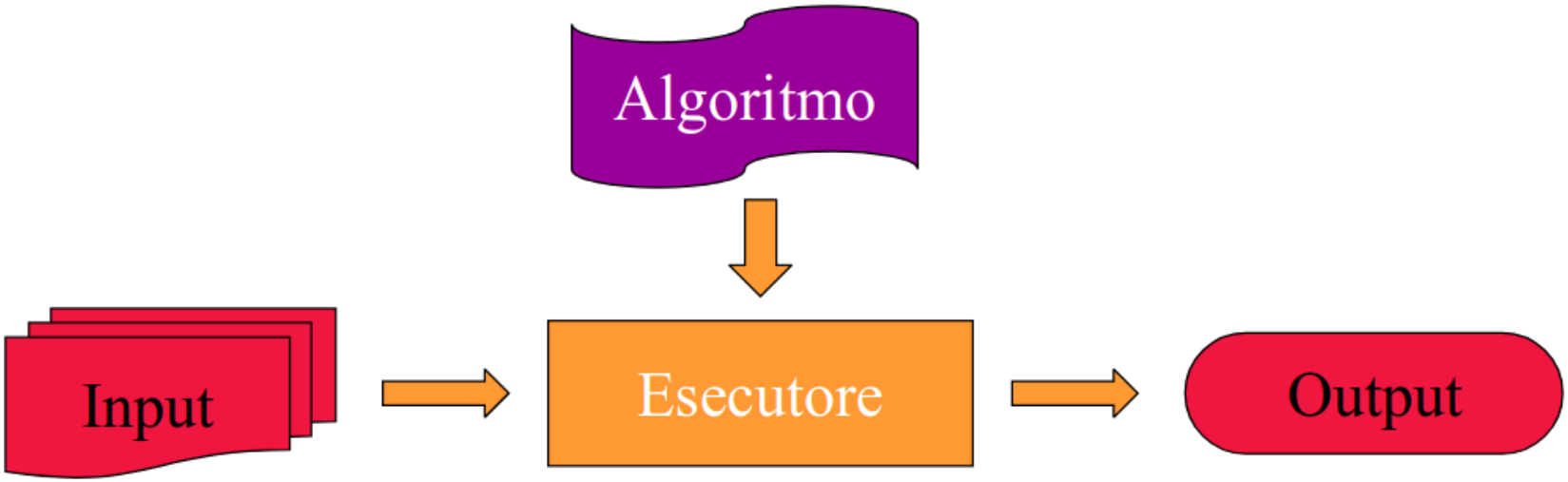
Un problema risolvibile mediante un algoritmo si dice **computabile**.

Esempio

Preparazione del risotto ai funghi:

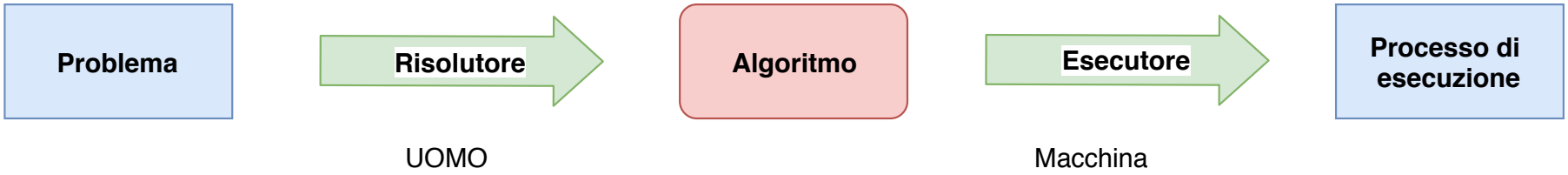
- Dati gli Ingredienti (riso, funghi, cipolla, pepe, ...)
- Seguire la ricetta (preparare i funghi, tostare il riso, procedere con la cottura, etc)
- Servire il piatto a tavola

Risoluzione di un problema



1. Prendere i dati iniziali (**input**)
2. Concepire l'algoritmo e codificarlo affinche' sia **interpretabile** da uno opportuno risolutore.
3. Avviare un **esecutore**.
4. Attendere la **fine del lavoro dell'esecutore** ed infine prelevare lo **output**.

Risoluzione di un problema: uomo vs macchina



Uomo è **reale risolutore** (concepisce algoritmo)

Macchina è solo **esecutore**.

- genera un **processo di esecuzione** e..
- alla fine di esso un output

Definizione meno informale di algoritmo

Algoritmo

Sequenza **ordinata** e **finita** di passi (azioni o istruzioni) che producono un ben determinato **risultato in un tempo finito**.

Caratteristiche di un algoritmo

1. Azioni **eseguibili** e **non ambigue**
   - "abbastanza", "a volontà" non sono espressioni ammissibili
2. **Determinismo**.
   - Fatto un passo, il **successivo** è uno ed uno solo, ben **determinato**.
   - **Alternative** sono ammesse, ma la scelta deve essere univoca.
3. **Numero finito di passi**.
4. **Terminazione**
   - NB: Numero finito di passi **non implica** terminazione.

Caratteristiche di un algoritmo

Ogni passo deve:

- **terminare** entro un intervallo **finito di tempo**;
- produrre un effetto **osservabile**;
- produrre lo **stesso effetto** ogni volta che viene eseguito a partire dalle **stesse condizioni iniziali** (input, valori iniziali delle varibili, etc)

Esempi

1. Algoritmo che non termina

1. Si consideri un numero N
2. Scrivere N.
3. Scrivere il numero successivo.
4. Ripetere il passo precedente.

Quale algoritmo? Dipende dal tipo di input..

2. Ricerca di un nome in elenco

- Elenco non ordinato (lista di firme)
  - Ricerca sequenziale..
- Elenco ordinato (elenco telefonico)
  - Ricerca dicotomica..

Codifica degli algoritmi: linguaggi e programmi

Codifica di un algoritmo

Osservazione

La macchina deve essere in grado di **comprendere** i passi dell'algoritmo.

Comprendere l'algoritmo: attribuire la corretta **semantica** a tutti i passi o istruzioni in modo che essi siano **eseguiti in modo corretto**.

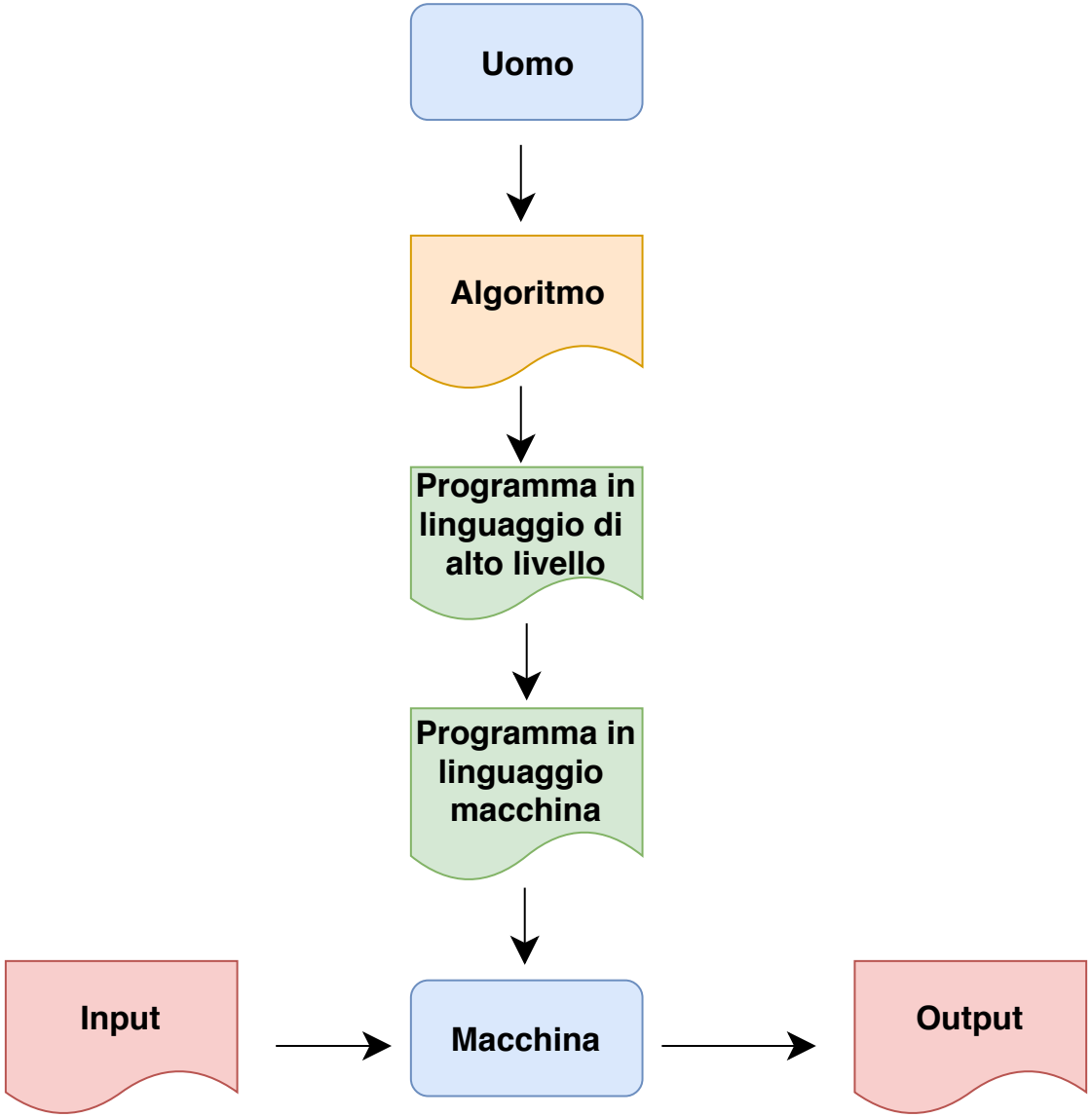
Codifica di un algoritmo

Programma

Algoritmo **codificato** in un uno opportuno **linguaggio** di “alto livello”.

Il risultato di tale codifica viene chiamato **programma**

Codifica di un algoritmo



Progettazione di algoritmi

Progettare un algoritmo

Risotto ai funghi

1. Preparare il brodo vegetale (*)
2. Far bollire i funghi **per 3 minuti circa**
3. Trifolare i funghi (*)
4. Tostare il riso **fino a quando sarà ben caldo** (*).
5. Sfumare con vino bianco (*)
6. Aggiungere brodo vegetale
7. Cuocere **per 12 minuti** circa aggiungendo brodo vegetale **quanto basta**
8. Aggiungere i funghi trifolati e cuocere **per altri 5 minuti**
9. Lasciare riposare **per 5 minuti**

NB: I passi con asterisco rappresentano sottoproblemi.

Progettare un algoritmo

Un sottoproblema: Preparare il brodo vegetale

- Ingredienti: 2 gambi di Sedano, 4 carote medie, 1 ciuffo di prezzemolo, 1 cipolla grande.
- Riempire una casseruola di acqua e portare velocemente a ebollizione.
- Versare gli ingredienti nella casseruola e fare sobbollire per circa un'ora e trenta.
- A cottura completata filtrare il brodo con un canovaccio da cucina.

Progettare un algoritmo

Un altro sottoproblema: sfumare con il vino bianco

- versare mezzo bicchiere di vino bianco
- far sobbollire a fuoco lento **fino a quando tutto l'alcool sarà evaporato**

La ricetta precedente è un esempio di cosa significa **scomporre un problema in sottoproblemi**.

Ogni sottoproblema può essere scomposto in **problemi via via più elementari**.

Approccio Top-down

1

Si costruisce una **visione generale del problema**, senza scendere nel dettaglio delle sue parti

- ES: preparare il brodo vegetale.

2

Ogni parte del sistema è *successivamente rifinita* per **decomposizione** aggiungendo dettagli.

- ES: Lista di ingredienti per il brodo, come prepararli, tempo di cottura e filtraggio

Approccio Top-down

3

Si opera, se necessario, mediante **successive decomposizioni**, che permetteranno di specificare ulteriori dettagli.

4

Il processo di decomposizione potrà **concludersi** quando la specifica avrà fornito **sufficienti dettagli** da poter validare il modello.

Approccio Bottom Up

1

Parti individuali del sistema sono *specificate* in dettaglio.

2

La parti vengono **connesse** tra loro per formare **componenti** più grandi.

Approccio Bottom Up

3

Successive connessioni/composizioni permetteranno di realizzare un sistema più completo.

Bottom up (puro) si usa spesso quando

- si hanno a disposizione **svariate componenti pronte** per essere utilizzate. Queste possono essere collegate insieme a formare componenti più grandi.
- si dispone di una certa **esperienza** nella realizzazione di un sistema che risolve lo **stesso problema o problemi simili**.

Top Down vs Bottom Up

Spesso si adotta un **approccio ibrido**

Esempio

Stampare tutti i nomi di persona presenti in un testo.

1. (TD) Leggere il testo, riga per riga, separando le singole parole.
   - (BU) Usare la funzione **getLine()**.
   - (BU) Usare la funzione **getWords()** sull'intera riga.
2. (TD) Memorizzare ogni nome di parola quando esso viene letto.
3. (TD) Stampare tutti i nomi di parola memorizzati.
   - (BU) Usare la funzione *print()* su tutte le parole.

Diagrammi di flusso

Descrizione di un algoritmo

La descrizione di un algoritmo è **propedeutica** alla sua successiva **codifica**.

Ma va usato un linguaggio generale, **indipendente** dalla codifica stessa:

1. DIAGRAMMI DI FLUSSO
2. PSEUDO-CODICE

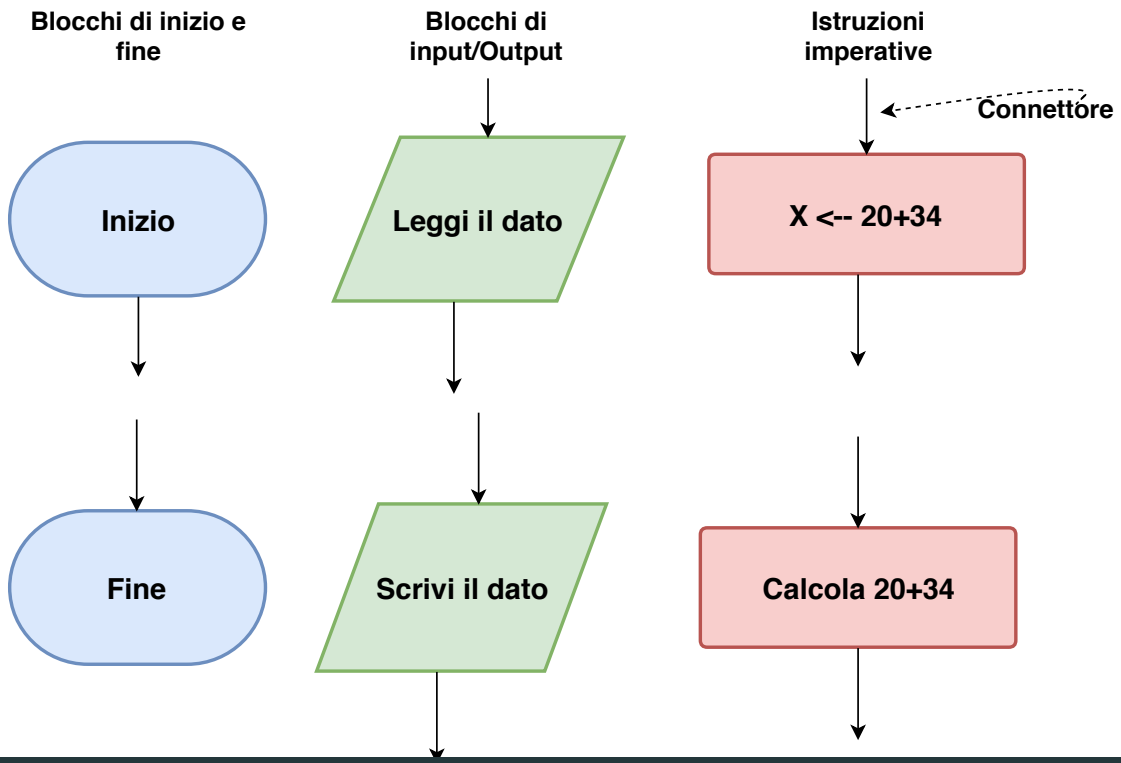
Diagrammi di flusso

Un diagramma di flusso permette di descrivere in **modo grafico** le **azioni** che costituiscono un algoritmo nonché il loro **flusso di esecuzione**.

- I **Blocchi** rappresentano le azioni
- I **connettori** permettono di specificare in quale **ordine** vanno eseguite le azioni

Blocchi e connettori

L'ordine di esecuzione delle istruzione avviene in base ai connettori.
La posizione dei connettori determina il **flusso di esecuzione**



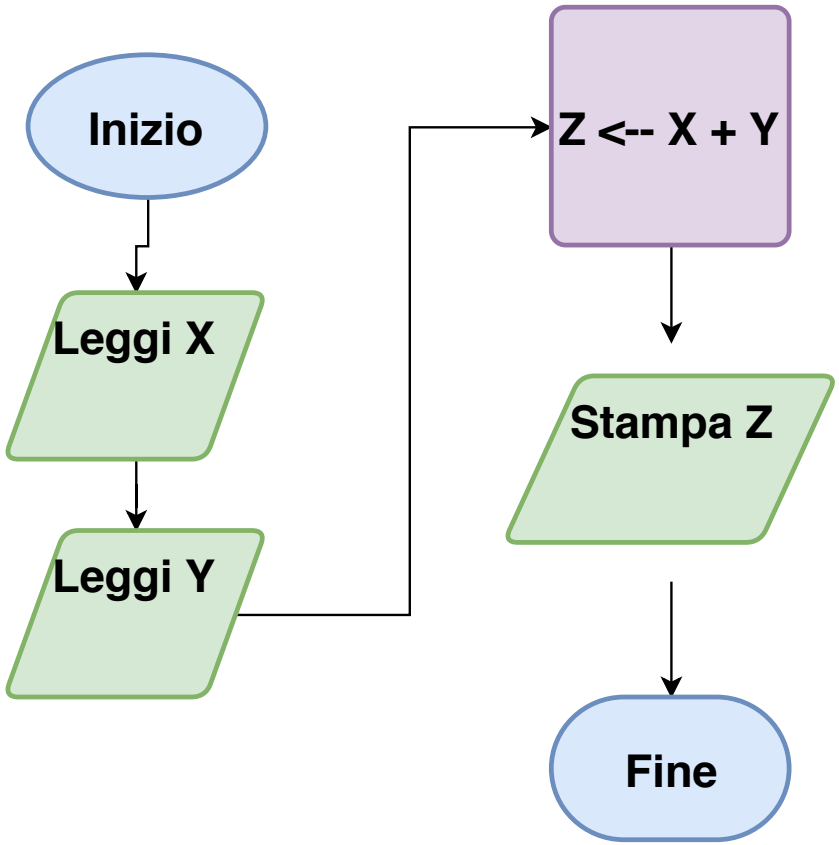
Istruzioni di assegnamento

Variabile $\leftarrow$ Espressione

- Variabile: Entità caratterizzata da
  - **nome**
  - **valore** il quale puó cambiare nel tempo
  - ES: X, Y, Z, Pippo, ...
- Espressione: **combinazione** di operatori aritmetici, costanti e variabili che da luogo ad un **risultato** numerico.
  - ES: $X + 2$, $Y/2$, $Y\%2$, ...
- ES: $X \leftarrow Y + 2$, $Y \leftarrow Y/2 + Z$

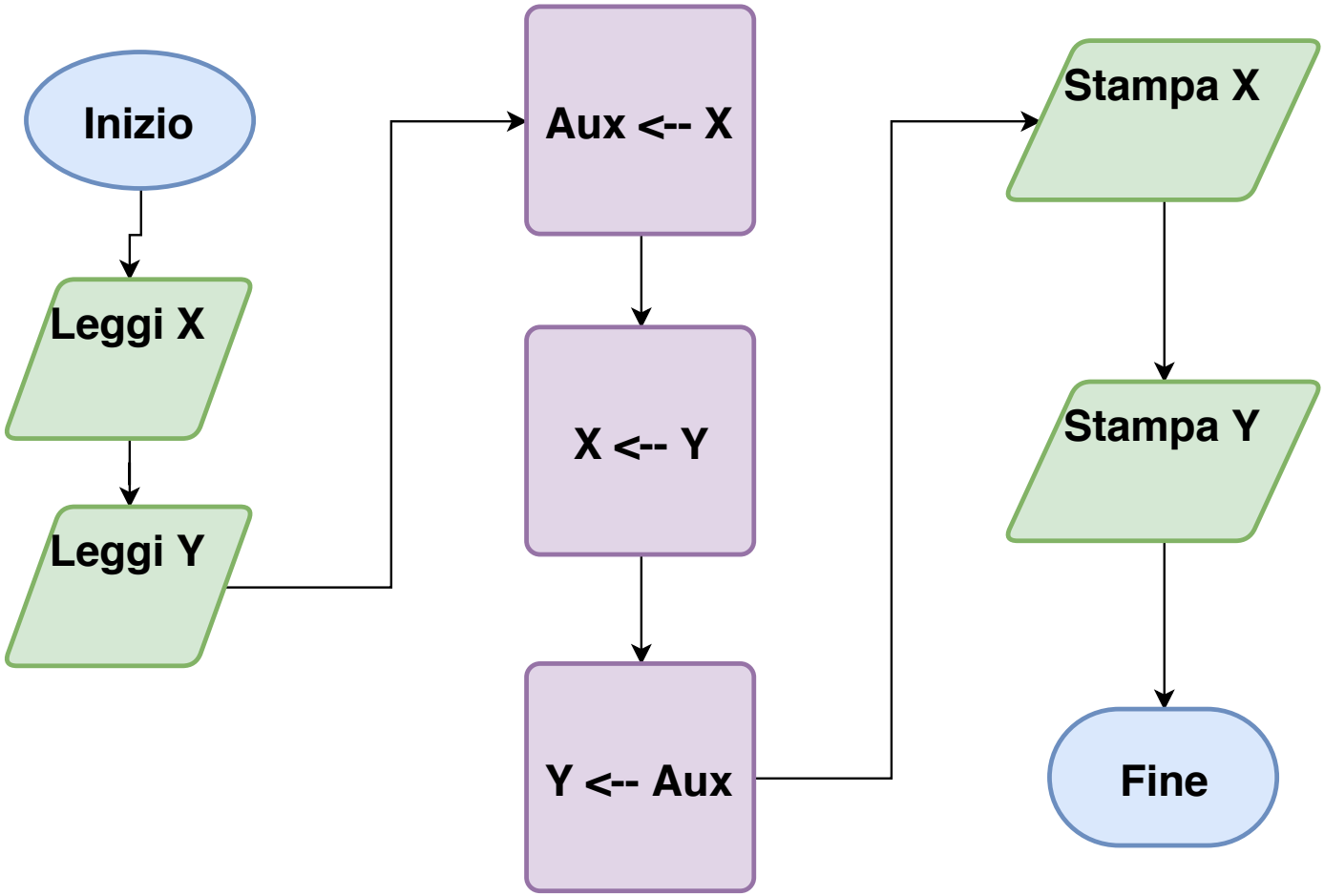
Diagrammi di flusso

Esempio: Somma di due numeri letti in input



Diagrammi di flusso

Esempio: Scambio di due numeri



Istruzioni condizionali

Ci sono dei momenti in cui il **flusso** di esecuzione può scegliere tra **diverse direzioni**.

In genere, questi **salti** sono subordinati al verificarsi di una **condizione** (che puó risultare vera o falsa);

$\Longrightarrow$ Istruzione condizionale

Diagrammi di flusso

ℹ Proposizione

Costrutto linguistico del quale si può affermare la **veridicità**.
ES: Il numero 2 é pari $\Longrightarrow$ VERO
Il valore di verità di una proposizione è il suo essere vera o falsa.

ℹ Predicato

Un predicato è una proposizione il cui valore di **verità dipende** dall'istanziazione di alcune **variabili**
ES: La variabile X è minore di 30. / La variabile Y è maggiore della variabile X

Valutazione di un predicato

Operazione che permette di **determinare** se il predicato é vero o falso, sostituendo alle variabili i loro valori attuali.
I valori VERO e FALSO sono **valori logici o booleani**

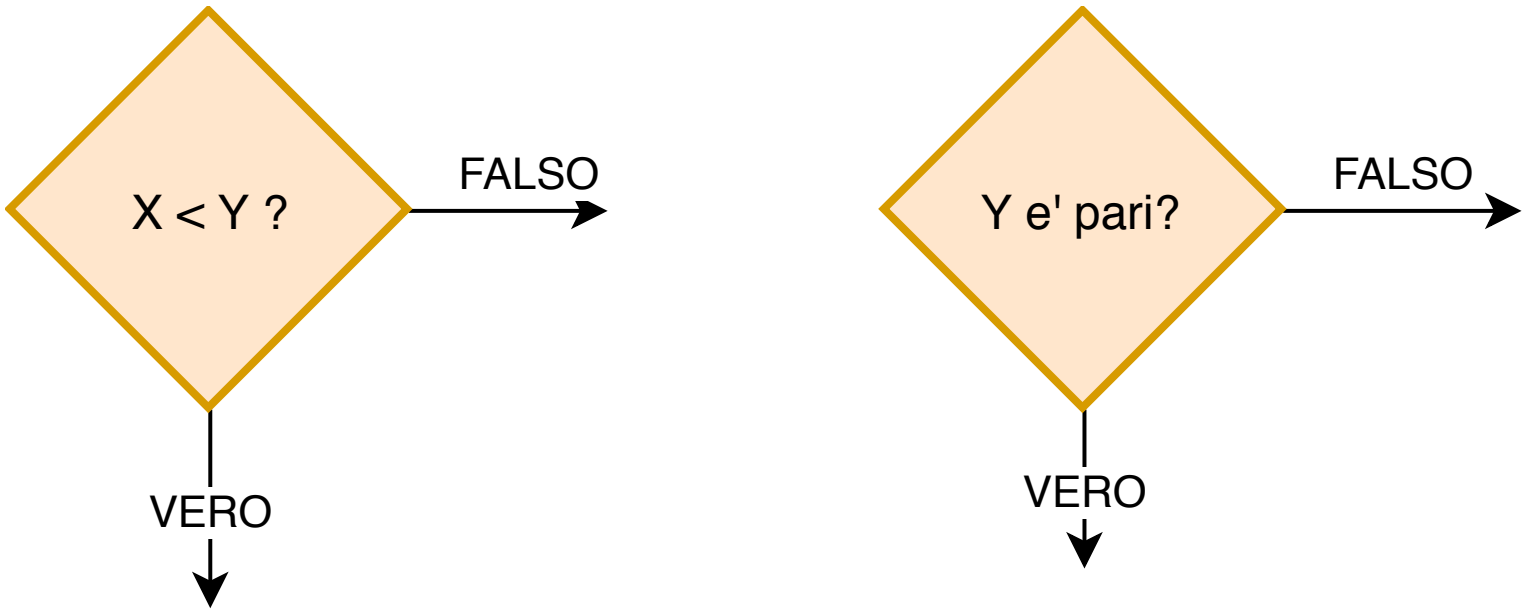
Operatori relazionali

Esprimere in **modo conciso** i predicati mediante **variabili e operatori relazionali**:

$=$ (uguale)	$\neq$ (diverso)
$\leq$ (minore o uguale)	$\geq$ (maggiore o uguale)
$<$ (minore)	$>$ (maggiore)

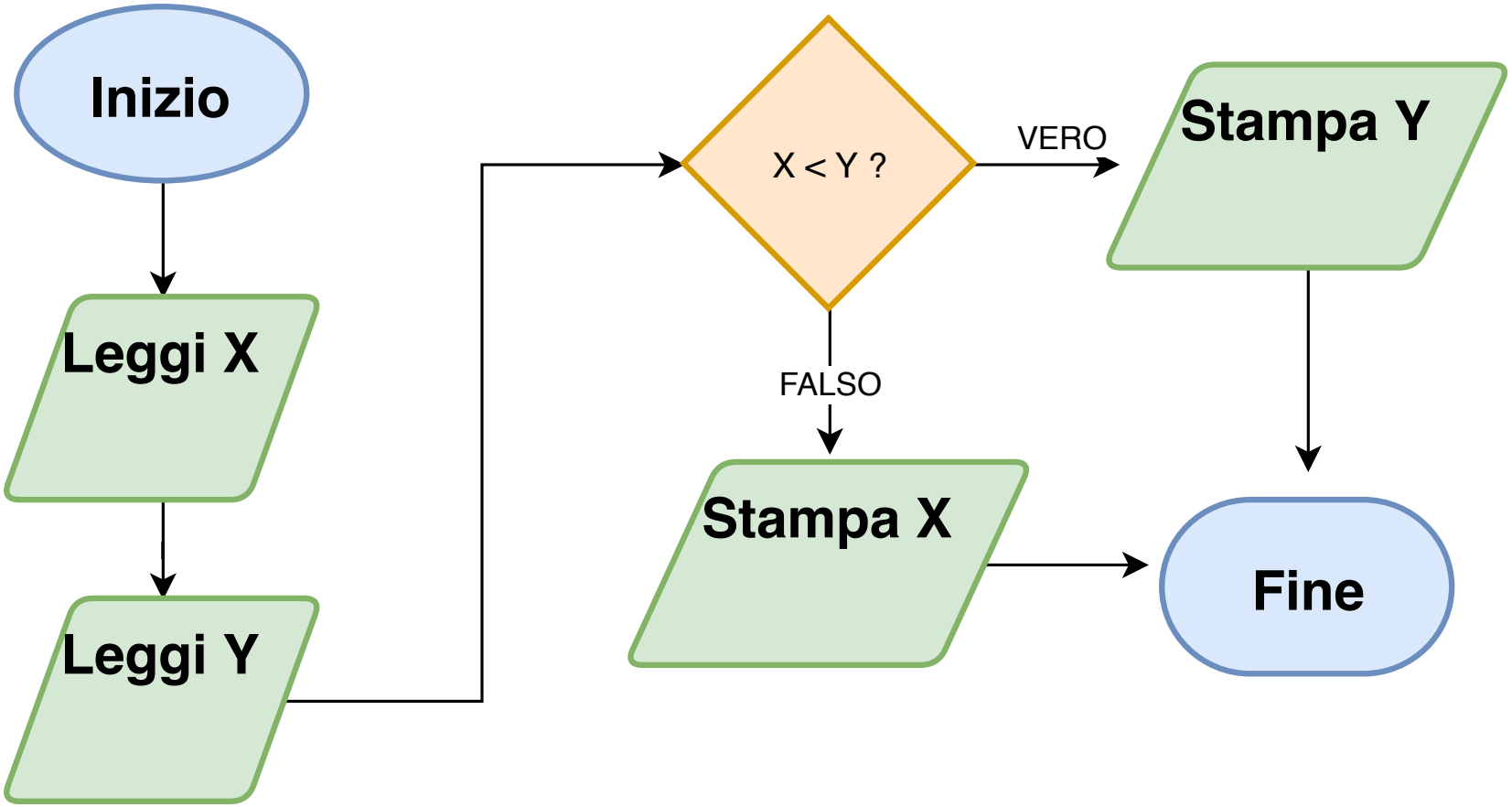
Blocco di istruzione condizionale

Il valore di verità del predicato "$X < Y$" (a sinistra) o del predicato "Y è *pari*" determina il prossimo passo nel flusso di esecuzione



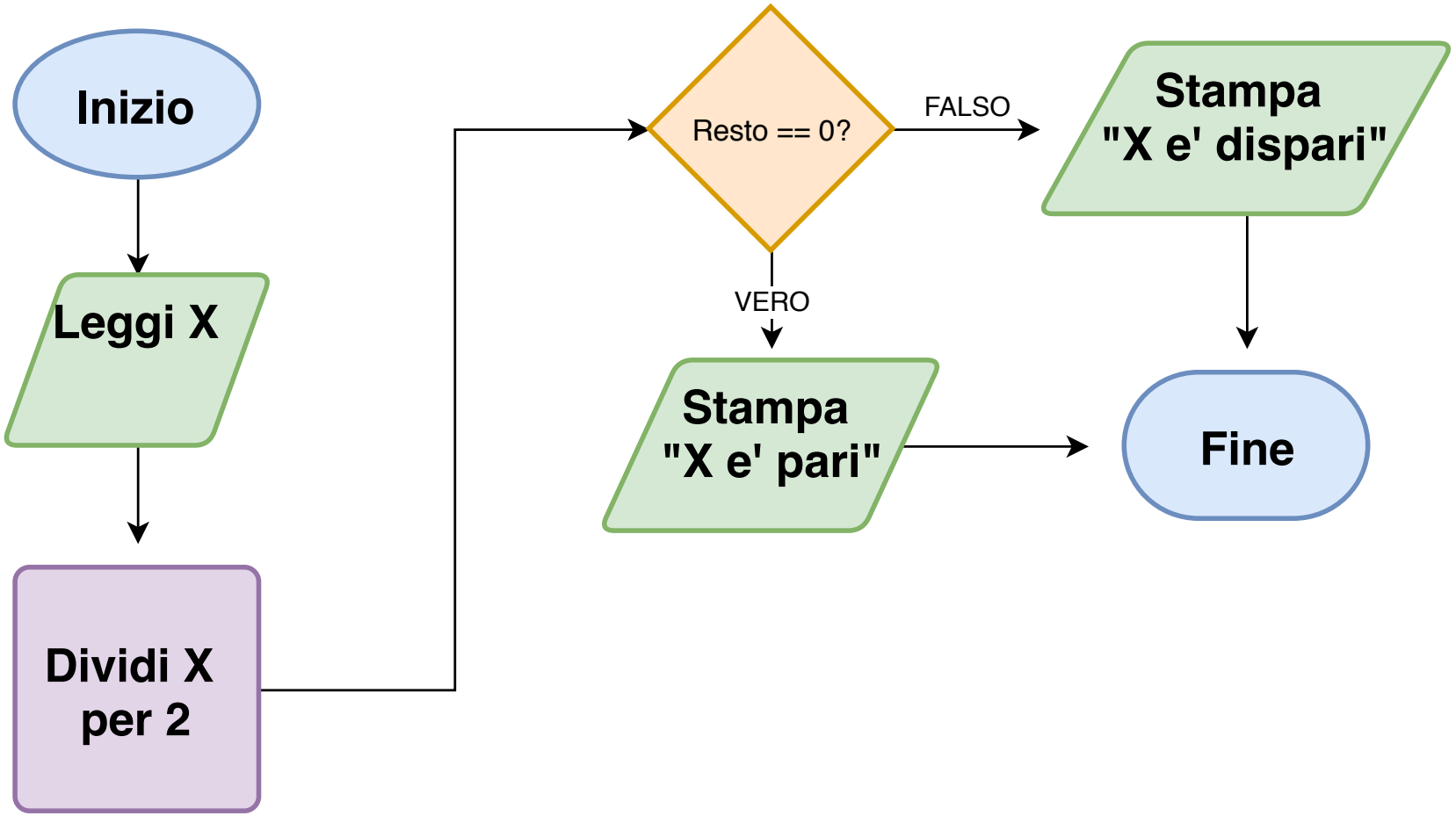
Diagrammi di flusso

Esempio: stampare il massimo tra due numeri



Diagrammi di flusso

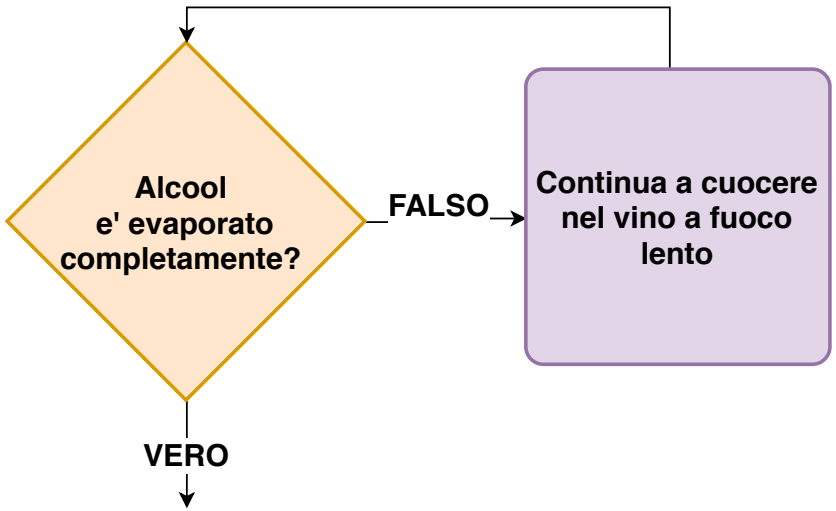
Esempio: Determinare se un numero è pari o dispari



Cicli (loop) o ripetizioni

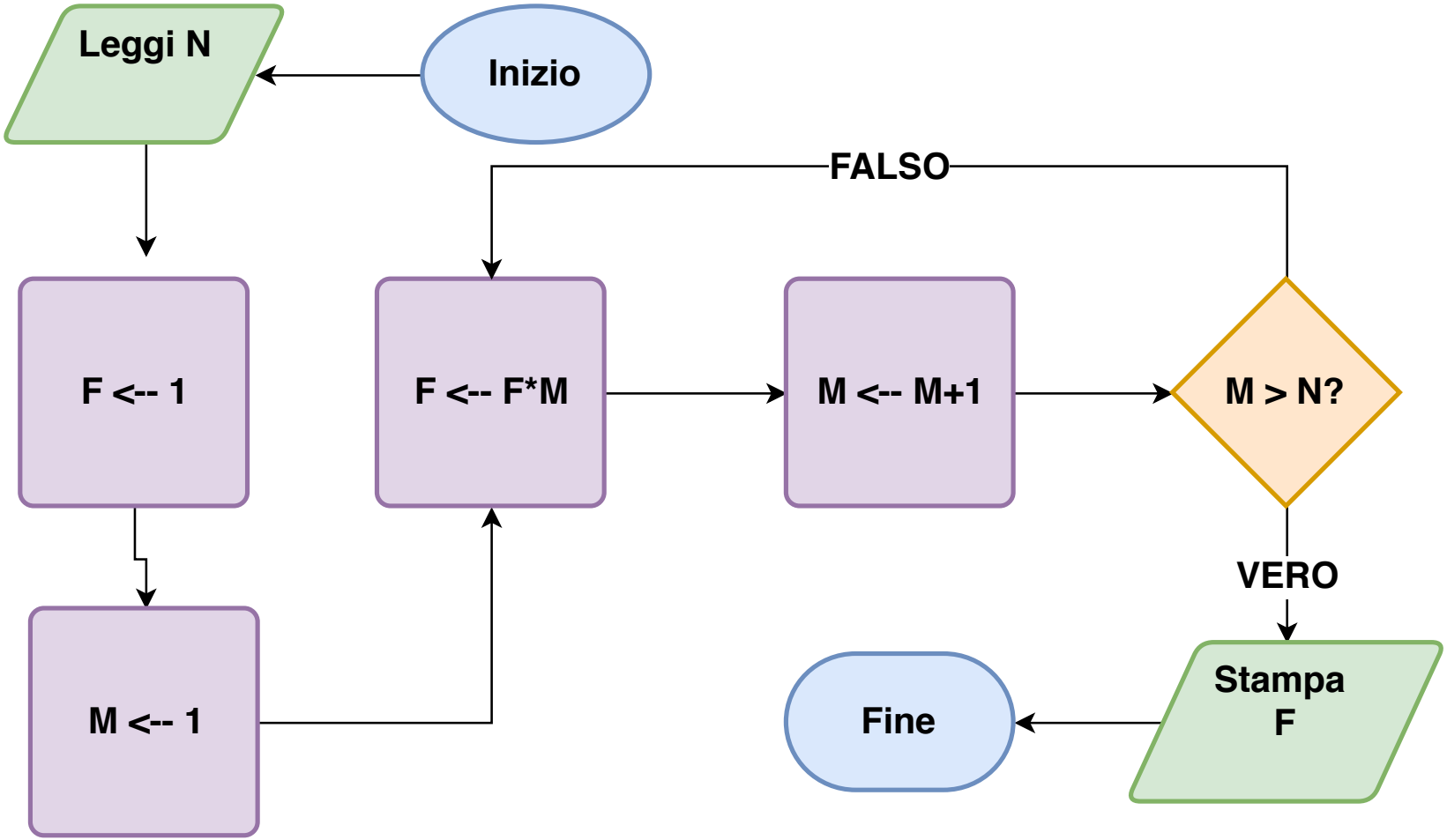
Loop o ciclo: sfumare con vino bianco

(dopo aver versato mezzo bicchiere di vino bianco . . .) far sobbollire a fuoco lento **fino a quando tutto l'alcool sarà evaporato**



Notazione Lineare Strutturata (NLS)

Esempio: calcolo del fattoriale di un numero



Considerazioni sui diagrammi di flusso

Se gli algoritmi da rappresentare sono **articolati e complessi**, i diagrammi di flusso a blocchi possono riverlarsi:

- poco pratici $\rightarrow$ soggetti ad errori
- poco leggibili

Alternativa: NLS (Notazione Lineare Strutturata)

Notazione lineare strutturata

Costrutti

Sequenza

Equivalente ad uno o più blocchi di operazioni che si susseguono.

Selezione

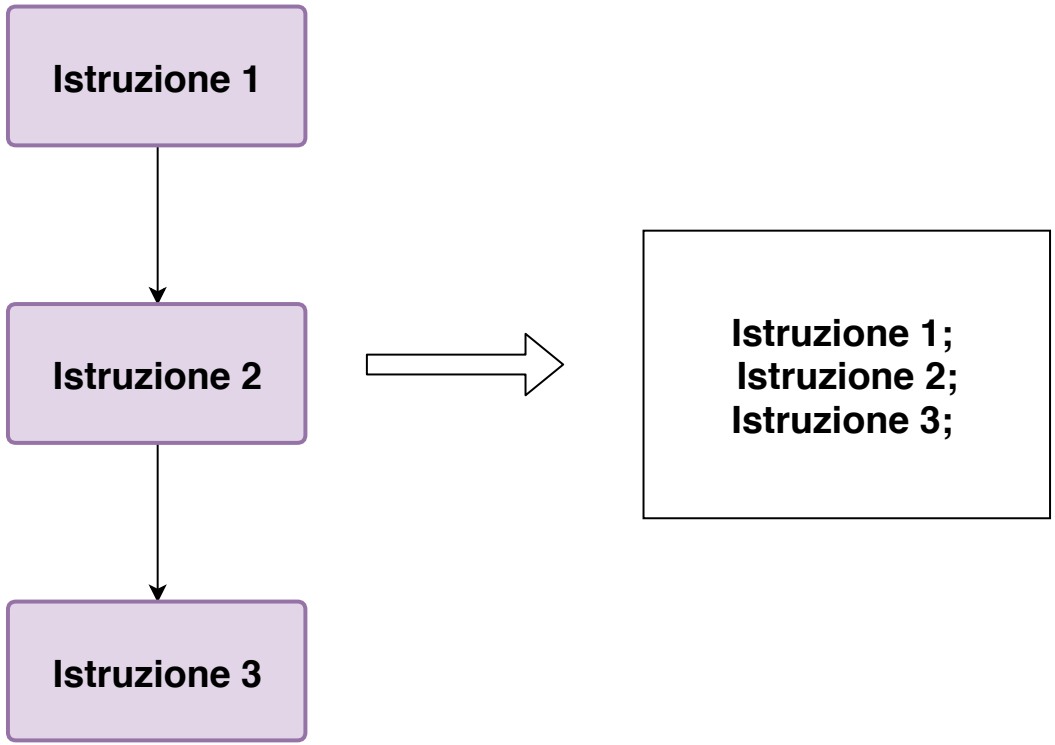
Equivalente al blocco condizionale

Iterazione

Equivalente al blocco condizionale piu uno o piu blocchi di operazioni disposti in modo da formare un ciclo

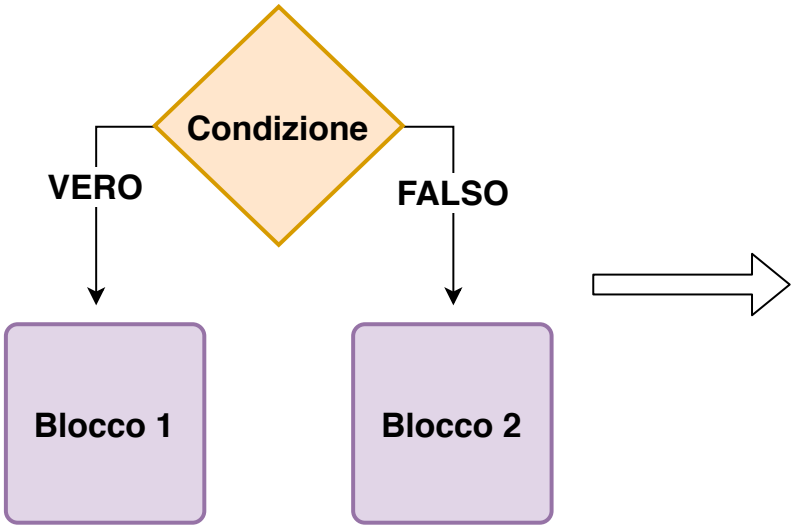
Notazione lineare strutturata

Sequenza



Notazione lineare strutturata

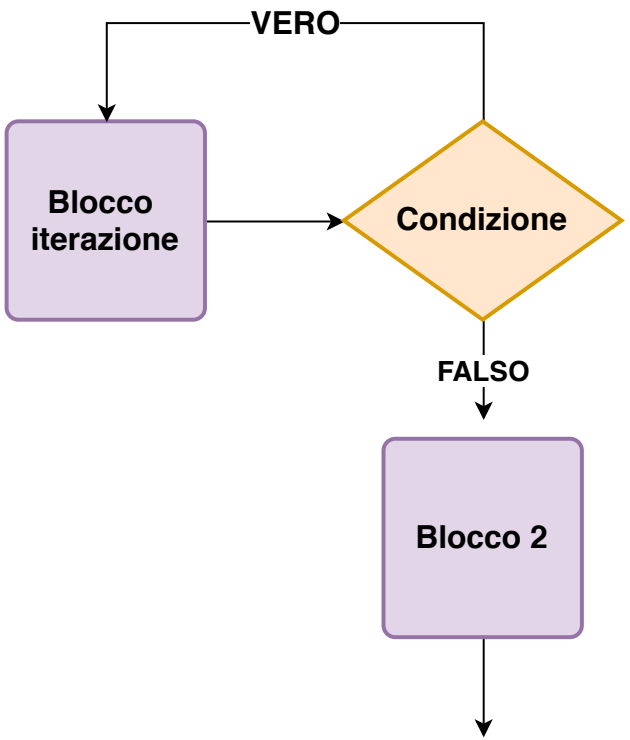
Selezione




```
If (condizione) Then
  blocco 1;
Else
  blocco 2;
End If
```

Notazione lineare strutturata

Ciclo o Iterazione



While(condizione)
    Blocco iterazione;
End While

Blocco 2;

Notazione lineare strutturata

Teorema di Böhm-Jacopini (1966)

Ogni algoritmo può essere costruito utilizzando unicamente tre strutture (o schemi di controllo):

1. la sequenza
2. la selezione
3. il ciclo o iterazione

↯

$\Rightarrow$ Ogni altro tipo di istruzione può essere **sostituito da una combinazione dei tre schemi precedenti**

Notazione lineare strutturata

In un qualsiasi linguaggio di programmazione sono **sufficienti espressioni che rappresentino le tre strutture NLS** per scrivere e implementare qualsiasi programma.

Ovviamente i **linguaggi di programmazione** mettono a disposizione anche **altri costrutti** (ad esempio il costrutto `for` del linguaggio C/C++).

NLS: esempi

Somma di due numeri

```
1 Inizio
2     Leggi X
3     Leggi Y
4     Z ← X + Y
5     Stampa Z
6 Fine
```

Massimo tra due numeri

```
1 Inizio
2     Leggi X
3     Leggi Y
4     If (X > Y) then
5         Stampa X
6     Else
7         Stampa Y
8     End If
9 Fine
```

Stampa i numeri da 1 a N

```
1 Inizio
2     Leggi N
3     M ← 0
4     While(M < N) Do
5         M ← M + 1
6         Stampa M
7     End While
8 Fine
```

Somma dei primi N numeri

```
1  Inizio
2      Leggi N
3      i ← 0
4      S ← 0
5      While (i < N) Do
6          i ← i + 1
7          S ← S + i
8      End While
9      Stampa S
10 Fine
```

Stampa le prime N+1 potenze del numero 2

```
1  Inizio
2     Leggi N
3     M ← 0
4     P ← 1
5     While(M <= N) do
6        Stampa P
7        P ← P · 2
8        M ← M + 1
9     End While
10 Fine
```

Algoritmo di euclide per il m.c.m

```
1  Inizio
2     Leggi A,B
3     MA ← A
4     MB ← B
5     While(MA <> MB) do
6        If(MA > MB) Then
7           MB ← MB + B
8        Else
9           MA ← MA + A
10       End If
11    End While
12    Stampa "mcm=" MA
13 Fine
```

Es: mcm(3,7) = 21

MA	MB
3	7
6	
9	
	14
12	
15	
	21
18	
21	

Algoritmo di Euclide per il M.C.D.

```
1  Inizio
2     Leggi A,B
3     If(A<B) Then
4        MB ← A
5        MA ← B
6     Else
7        MB ← B
8        MA ← A
9     End If
10    While(MB <> 0) do
11       r ← MA%MB
12       MA ← MB
13       MB ← r
14    End While
15    Stampa "MCD=" MA
16 Fine
```

Es: MCD(21,14) = 7

MA	MB	MA % MB
21	14	7
7	7	0
7	0	—