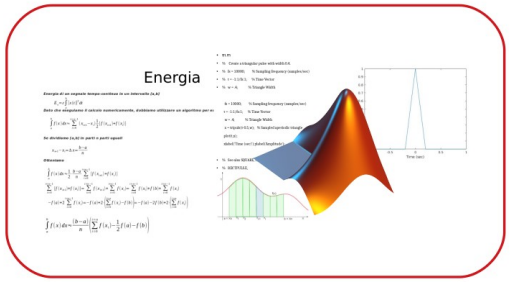
UNIVERSITÀ DEGLI STUDI DI TRIESTE

Programmazione Informatica

Ingegneria Civile e Ambientale
Ingegneria Navale

Programmazione - Matlab 2 – Input/Output (I/O), strutture di controllo



Antonio Fiumara
antonio.fiumara@dia.units.it

A. A. 2025-26

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Struttura del Corso

Gli argomenti trattati nel corso saranno:

- Lezione Teoria 1 – Introduzione, presentazione del corso, codifica dell'informazione
- Lezione Teoria 2 – Rappresentazione dell'informazione in un computer
- Lezione Teoria 3 – Algebra booleana, introduzione e applicazioni nella programmazione
- Lezione Teoria 4 – Architettura dei calcolatori. HW, SW, Sistemi Operativi
- Lezione Teoria 5 – Algoritmi e strutture dati
- Lezione Teoria 6 – Implementazione di algoritmi notevoli (ordinamento , ricerca, ecc)

- Lezione Programmazione - Matlab 1 – Introduzione, installazione, comandi di base, matrici
- **Lezione Programmazione - Matlab 2 – Algoritmi, Input/Output (I/O), strutture di controllo**
- Lezione Programmazione - Matlab 3 – Funzioni
- Lezione Programmazione - Matlab 4 – Sistemi Lineari, Algebra lineare e Analisi matematica
- Lezione Programmazione - Matlab 5 – Funzioni nel dominio del tempo e della frequenza
- Lezione Programmazione - Matlab 6 – Analisi ed elaborazione dei segnali
- Lezione Programmazione - Matlab 7 – Ripasso e approfondimenti
- Esonero esame Programmazione Matlab (3/12/25)

Introduzione

Algoritmi
Strutture di controllo
Funzioni di input

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Algoritmo

Definizione:

- Un algoritmo è un procedimento che risolve un determinato problema attraverso una sequenza finita e ordinata di passi elementari, chiari e non ambigui, in un tempo ragionevole.
- In altre parole, un algoritmo è un insieme di istruzioni che definiscono una sequenza di operazioni mediante le quali si risolvono tutti i problemi di una classe.

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Algoritmo - esempio

Algoritmo euclideo - calcolo del massimo comune divisore

Il Massimo Comune Divisore (MCD) è il più grande numero naturale per il quale due o più numeri possono essere divisi con resto nullo.

- L'algoritmo è costituito dai seguenti "passi"
- 1) Consideriamo i due numeri a, b
- 2) se b=0 il calcolo è terminato: MCD(a,b) = a; andare al passo 7.
- 3) se b>a scambiare a con b.
- 4) Calcolare il resto r della divisione di a per b
- 5) Se r=0 allora MCD(a,b) = b; andare al passo 7.
- 6) Sostituire a con b, b con r, andare al passo 2.
- 7) Fine

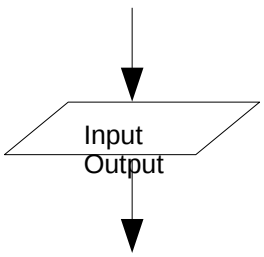
UNIVERSITÀ DEGLI STUDI DI TRIESTE

Algoritmo - proprietà

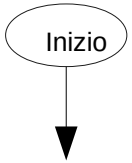
- Non si può risolvere un problema senza prima fissare un insieme finito di azioni elementari che l'esecutore sia in grado di comprendere ed eseguire
- **Non-ambiguità:** ogni azione deve essere univocamente interpretabile dall'esecutore dell'algoritmo
- **Eseguibilità:** ogni azione deve essere eseguibile in un tempo finito da parte dell'esecutore dell'algoritmo
- **Finitezza:** per ogni insieme di dati di ingresso, il numero totale di azioni da eseguire deve essere finito.
- Proprietà desiderabile:
- **Efficienza:** deve risolvere il problema utilizzando al meglio le risorse a disposizione

Diagrammi di flusso

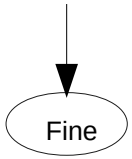
- Un algoritmo può essere descritto mediante un testo, oppure in forma grafica. Il diagramma di flusso rappresenta un metodo grafico di rappresentazione degli algoritmi ed è largamente utilizzato. Gli elementi di uso più frequente sono riportati qui di seguito



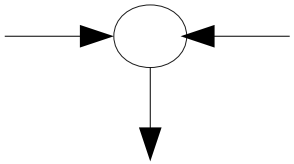
Input / Output



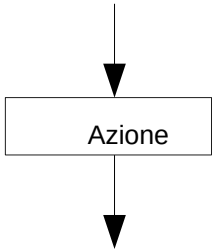
Punto d'inizio



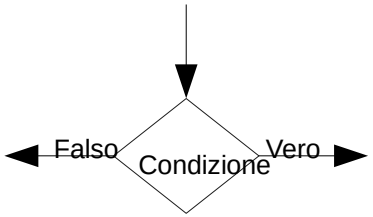
Punto di fine



Nodo



Azione o gruppo di istruzioni



Deciso re

Diagrammi di flusso

Calcolo MCD

1) Consideriamo i due numeri a, b che non siano contemporaneamente nulli.

Se a=0 e b=0 segnala errore e termina

2) se b=0, il calcolo è terminato: MCD(a,b) = a; andare al passo 7.

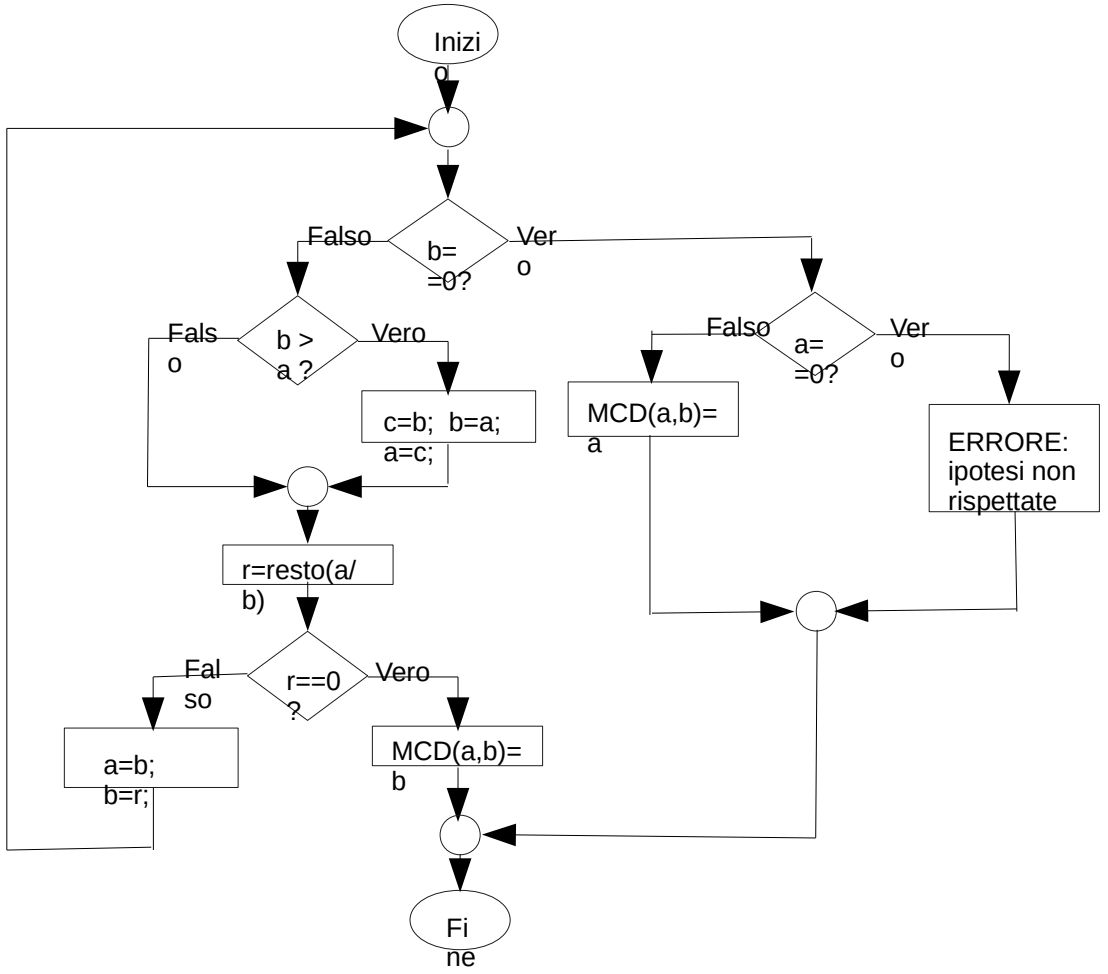
3) se b>a scambiare a con b.

4) Calcolare il resto r della divisione di a per b

5) Se r=0 allora MCD(a,b) = b; andare al passo 7.

6) Sostituire a con b, b con r, andare al passo 2.

7) Fine



UNIVERSITÀ DEGLI STUDI DI TRIESTE

Algoritmo proprietà

Requisiti fondamentali:

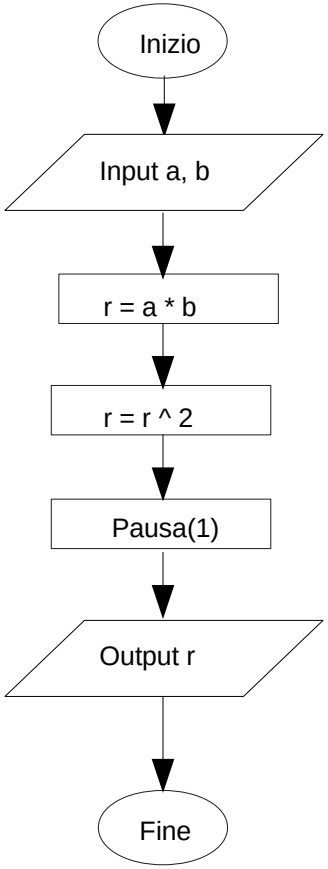
- **Completezza**. Un linguaggio strutturato deve fornire la **sequenza**, almeno una struttura di tipo alternativa (**if … then … else**), e almeno una struttura di tipo iterativa (**cicli**);
- **Singolo punto di ingresso e di uscita**.
- Ogni struttura di controllo, deve poter essere considerata come una **singola macro-istruzione** dal punto di vista del controllo complessivo, con un ben identificato punto di ingresso e un ben identificato punto di uscita.
- **Componibilità.** Ogni struttura di controllo può avere fra le sue istruzioni controllate, ricorsivamente, altre strutture di controllo (senza limiti).
- Gli ultimi due vincoli fanno sì che, per quanto concerne il flusso del controllo, ciascuna struttura di controllo definisca un ambito completamente isolato dalle altre, e incapace di interferire o subire interferenze. Questo rappresenta un requisito indispensabile per l'applicazione di metodologie di progetto e sviluppo tradizionalmente legate alla programmazione strutturata, come la progettazione **top-down** e lo **sviluppo per raffinamenti successivi**.

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Strutture di controllo - Sequenza

Struttura base, con un unico punto di inizio e un unico punto di fine.

- La sequenza è composta blocchi che rappresentano istruzioni, semplici o complesse, che vengono eseguite in ordine temporale
- Qualunque programma strutturato deve sempre poter essere rappresentato da una sequenza
- L'esecuzione di un blocco può iniziare solo dopo il completamento del blocco precedente (nel caso della programmazione parallela questo vincolo non è presente)
- Ogni blocco può contenere istruzioni semplici, altre strutture di controllo o interi sottoprogrammi o funzioni

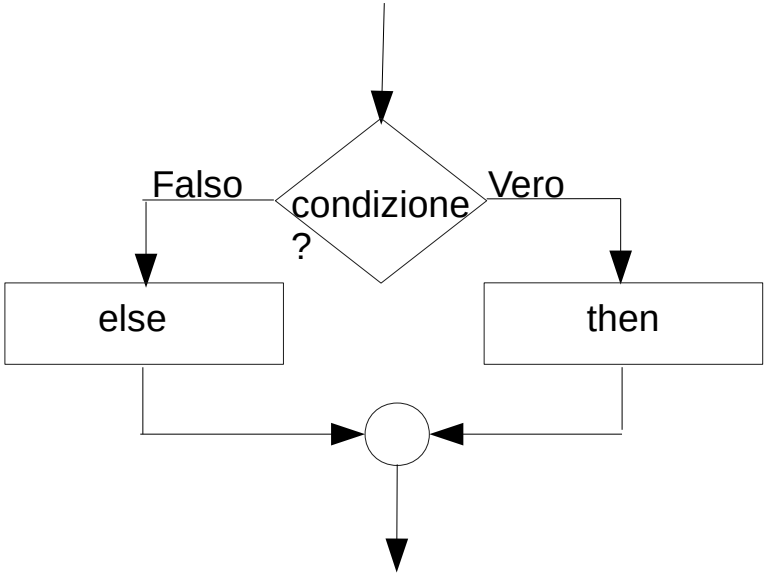


https://it.wikipedia.org/wiki/Programmazione_strutturata

UNIVERSITÀ DEGLI STUDI DI TRIESTE

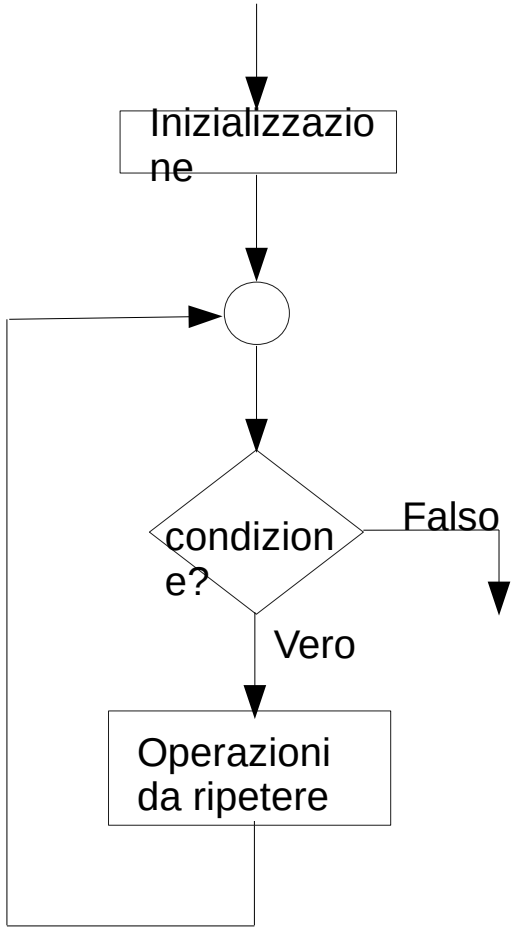
Strutture di controllo – Selezione (if … then … else)

- Struttura condizionale
- unico punto di inizio e un unico punto di fine.
- È costituita da un blocco condizionale e da due alternative, a seconda del risultato della condizione



Strutture di controllo – Ciclo while

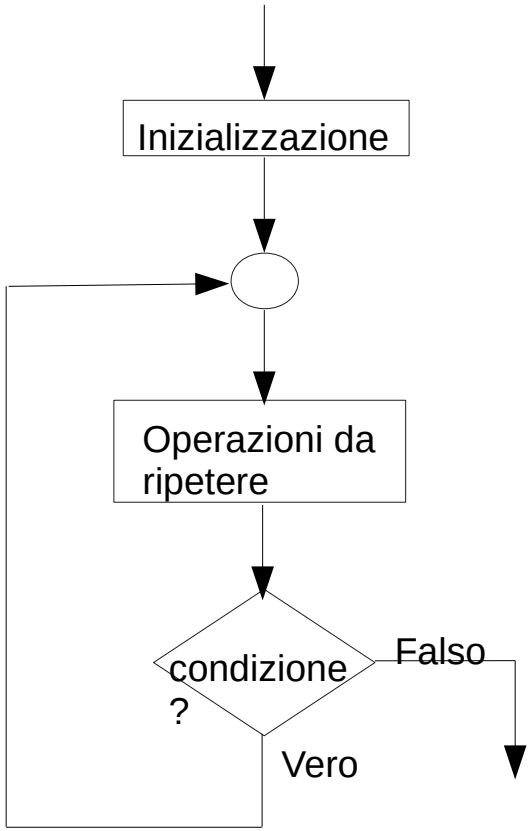
- unico punto di inizio e un unico punto di fine.
- È costituita da un blocco condizionale e da un blocco da ripetere finché il risultato della condizione rimanga vero
- N.B. La condizione viene verificata **prima** dell'esecuzione del blocco di istruzioni del ciclo. Se la condizione all'inizio del ciclo è falsa, non viene eseguito il blocco di istruzioni.



UNIVERSITÀ DEGLI STUDI DI TRIESTE

Strutture di controllo – Ciclo do

- ***N.B. NON IMPLEMENTATO IN MATLAB***
- unico punto di inizio e un unico punto di fine.
- È costituita da un blocco condizionale e da un blocco da ripetere finché il risultato della condizione rimanga vero
- N.B. La condizione viene verificata **dopo** dell'esecuzione del blocco di istruzioni del ciclo, quindi anche se la condizione all'inizio del ciclo è falsa, il blocco di istruzioni del ciclo viene eseguito almeno una volta.



Diagrammi di flusso – Esempio

Esempio: Algoritmo euclideo

1) Consideriamo i due numeri a, b che non siano contemporaneamente nulli.

Se a=0 e b=0 segnala errore e termina

2) se b=0, il calcolo è terminato: MCD(a,b) = a; andare al passo 7.

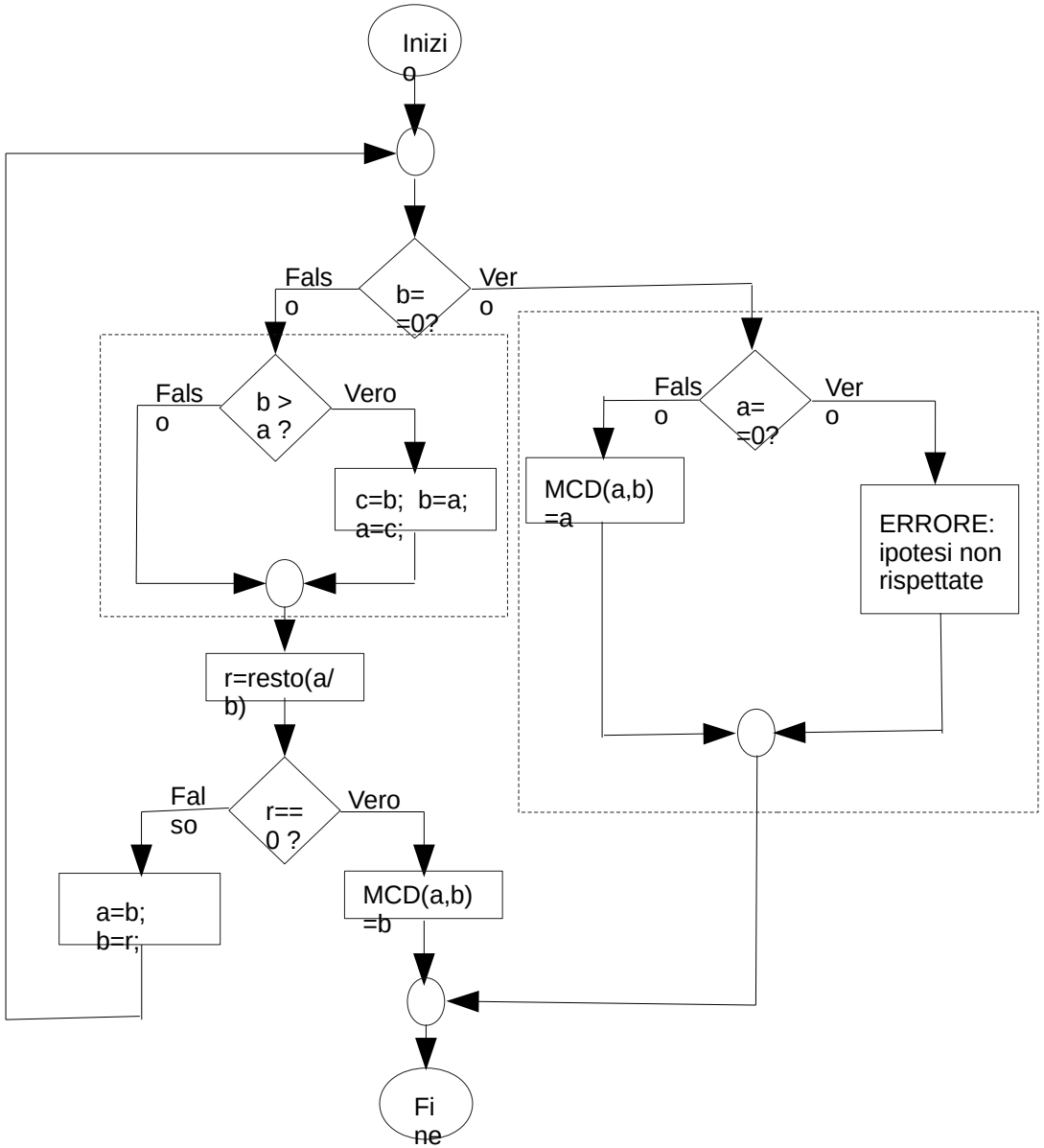
3) se b>a scambiare a con b.

4) Calcolare il resto r della divisione di a per b

5) Se r=0 allora MCD(a,b) = b; andare al passo 7.

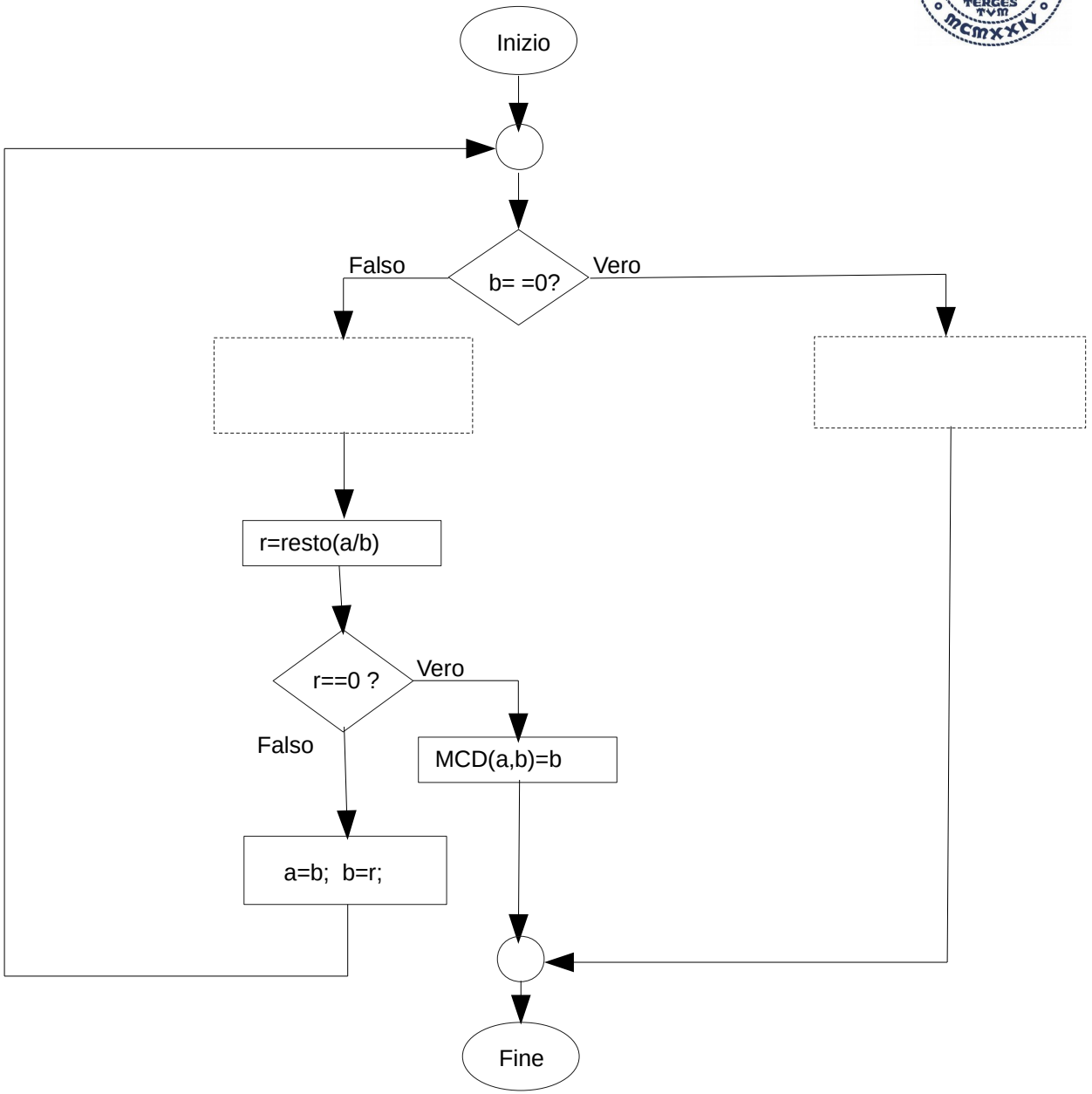
6) Sostituire a con b, b con r, andare al passo 2.

7) Fine

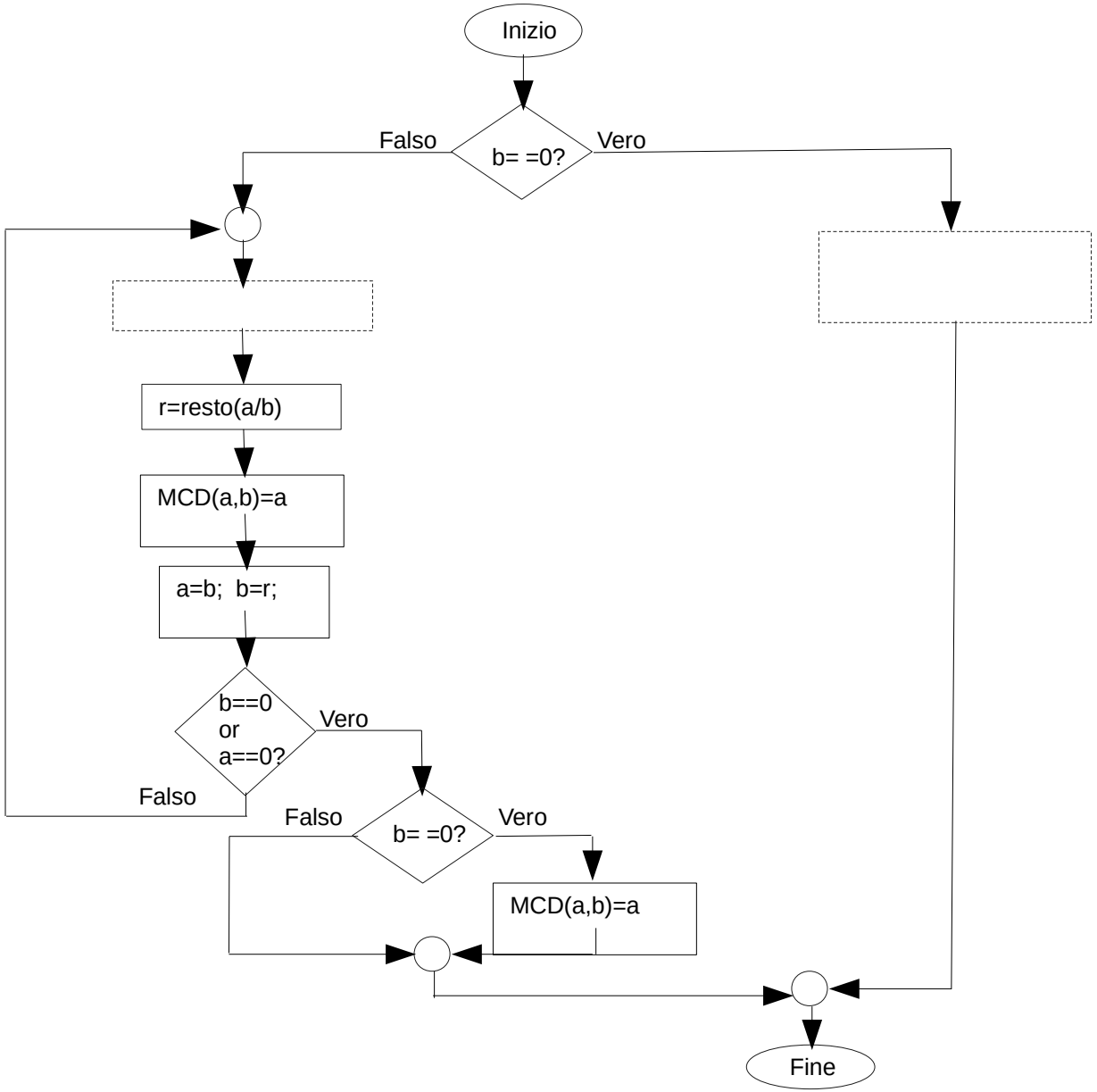


Diagrammi di flusso Algoritmo euclideo

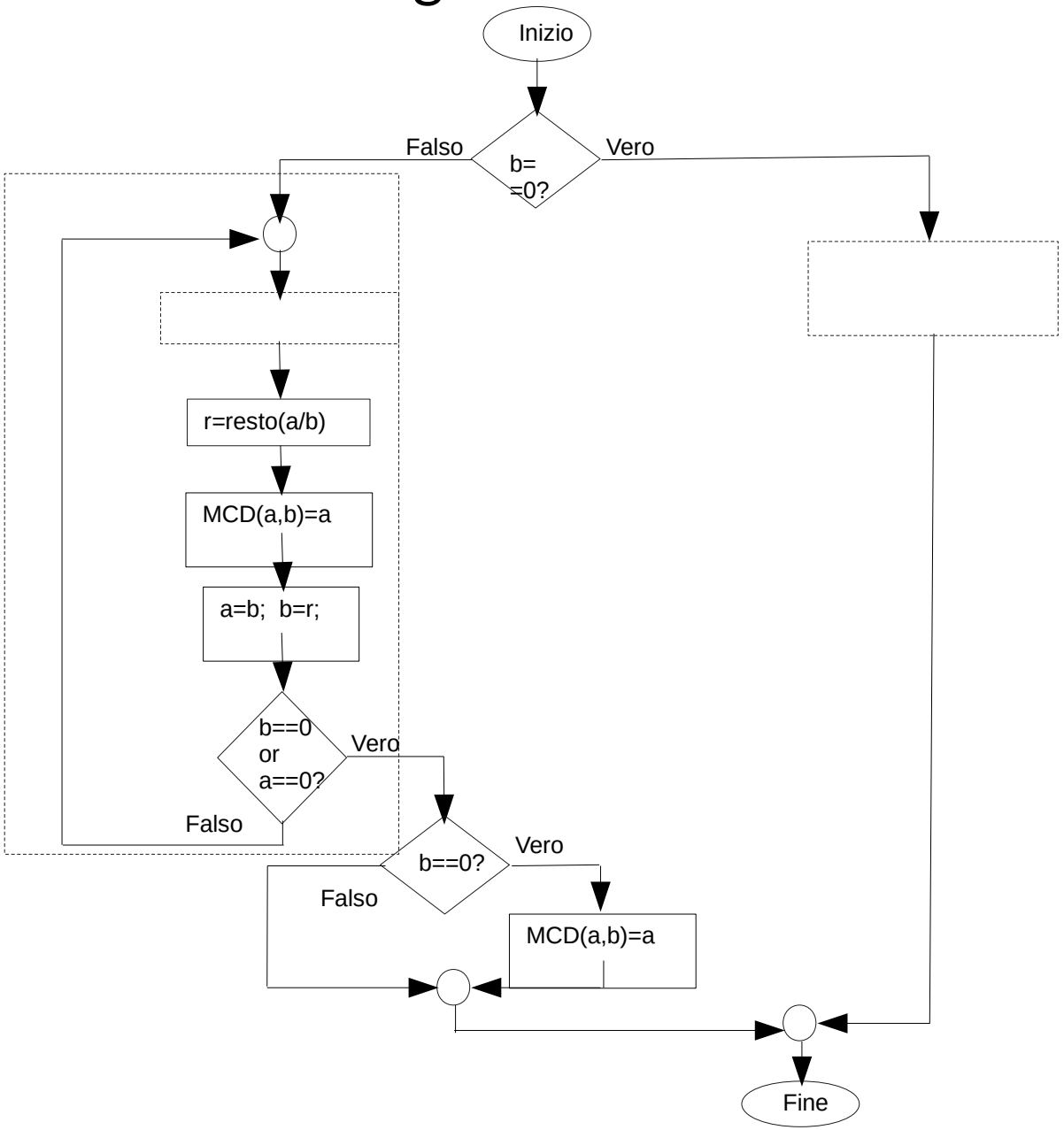
! Non è strutturato



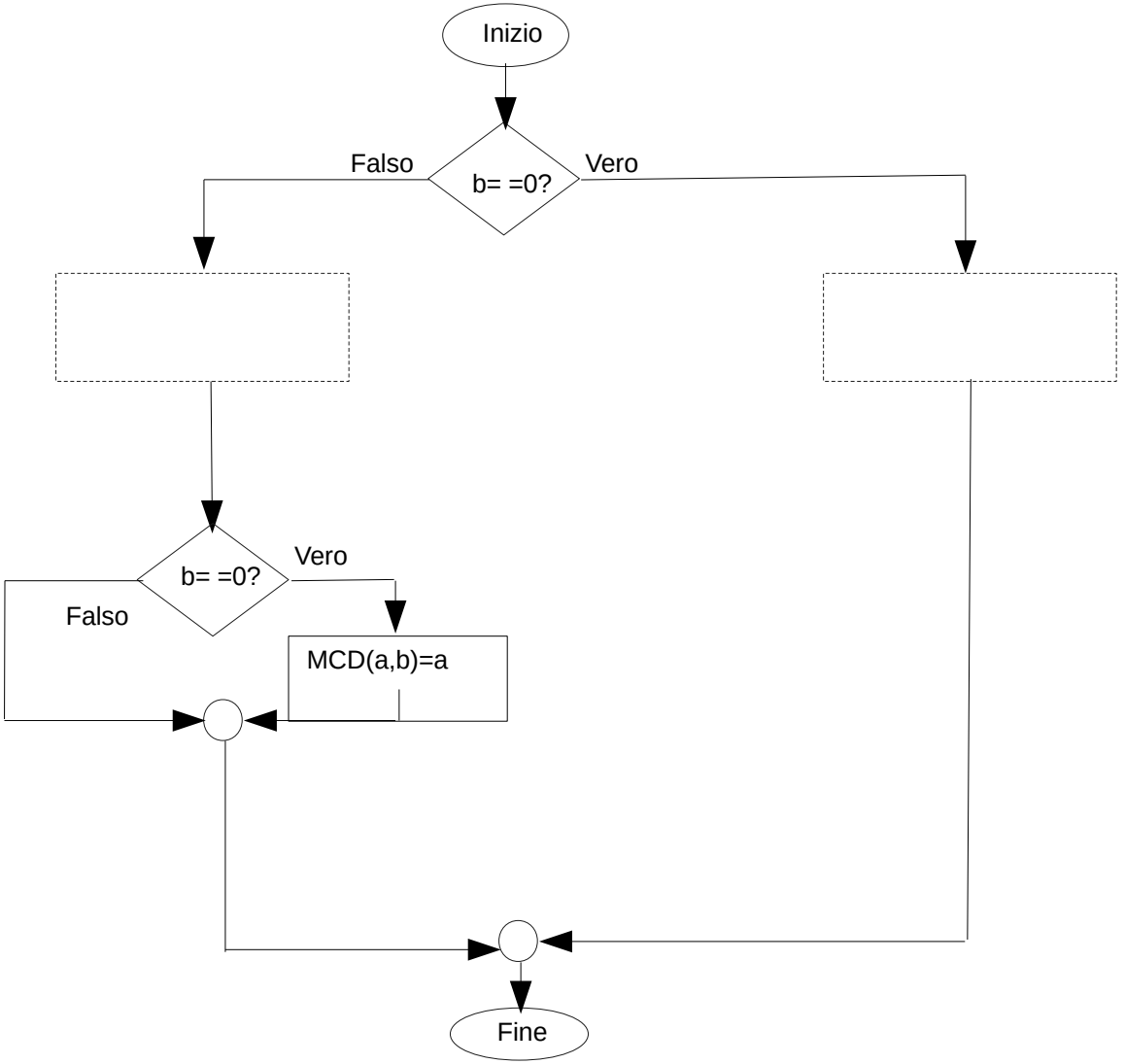
Trasformiamolo in algoritmo strutturato



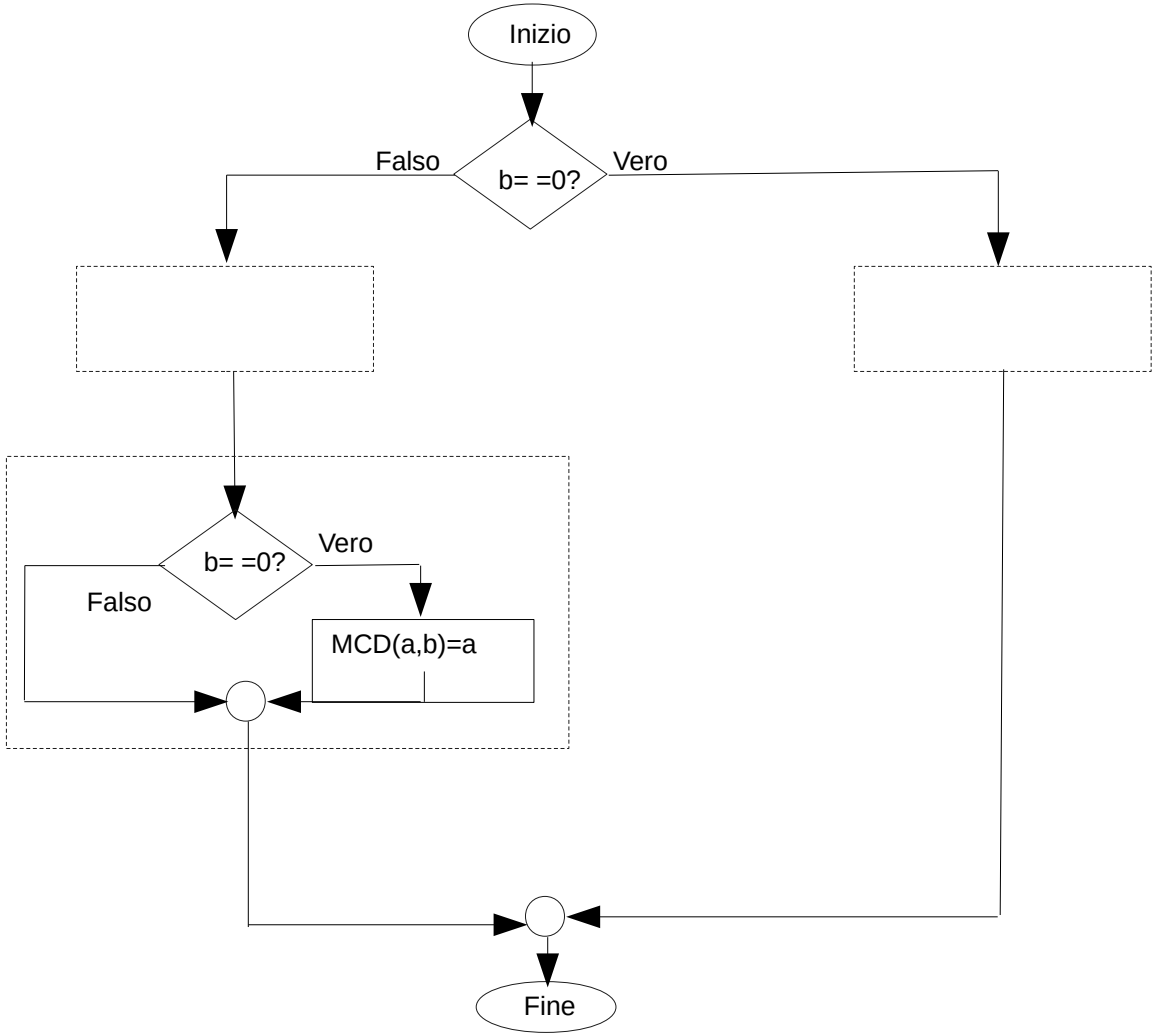
Trasformiamolo in algoritmo strutturato



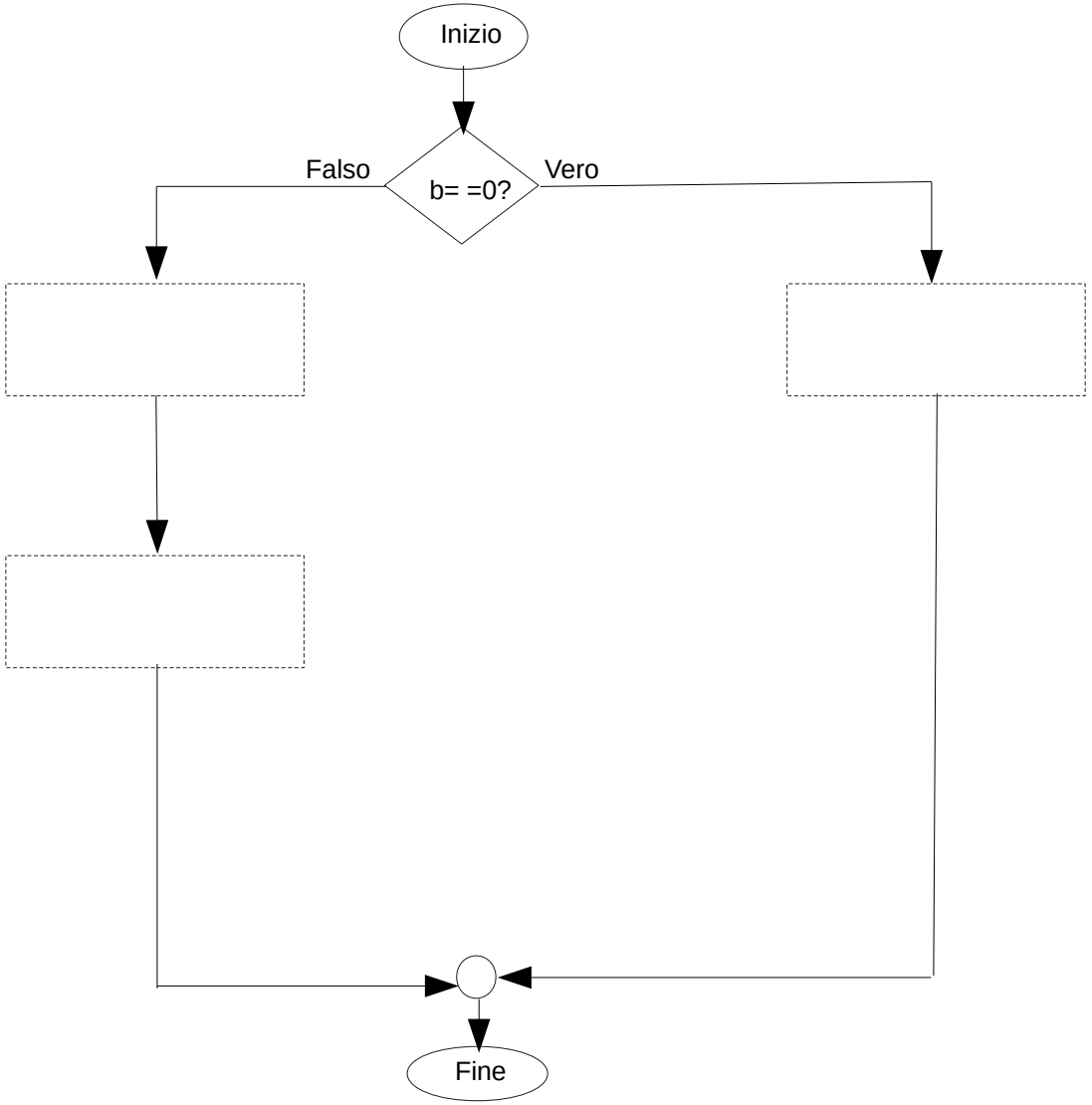
Trasformiamolo in algoritmo strutturato



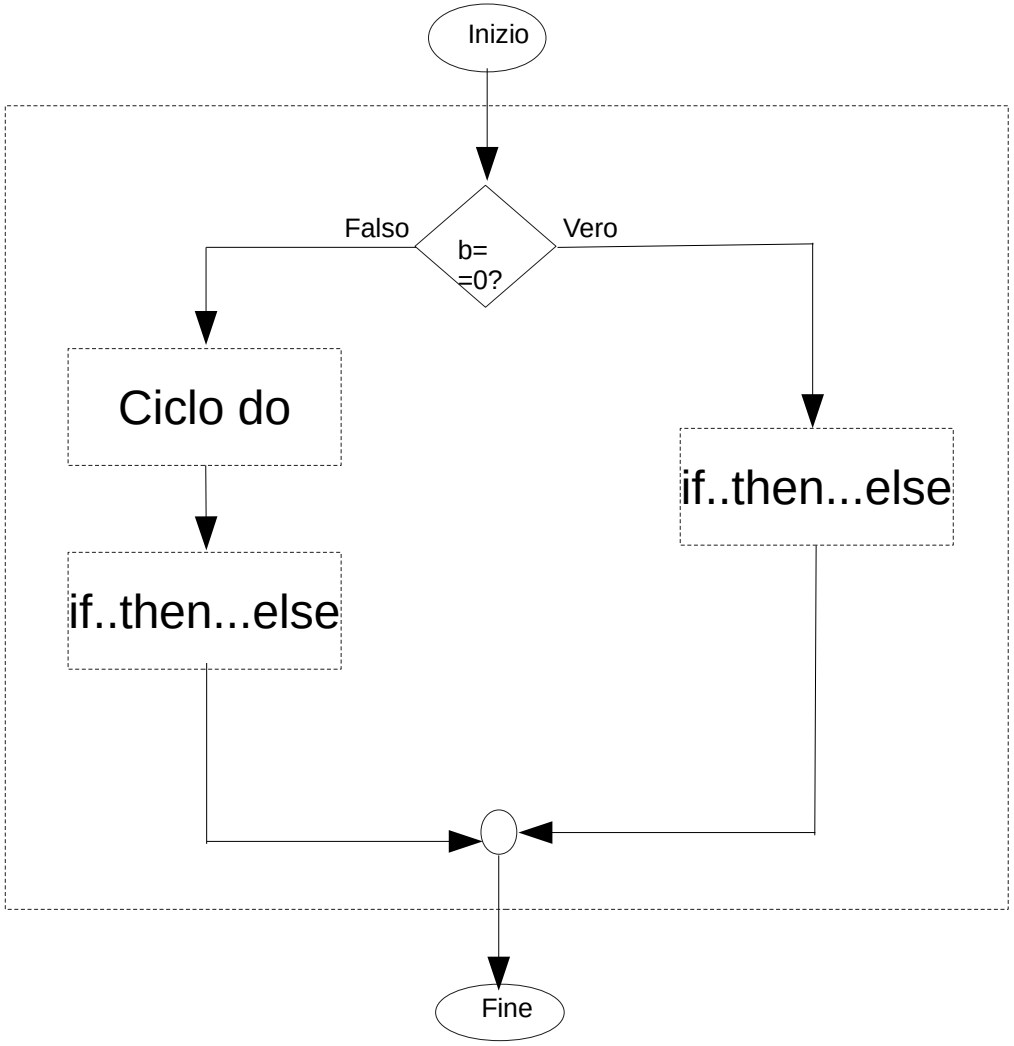
Trasformiamolo in algoritmo strutturato



Trasformiamolo in algoritmo strutturato



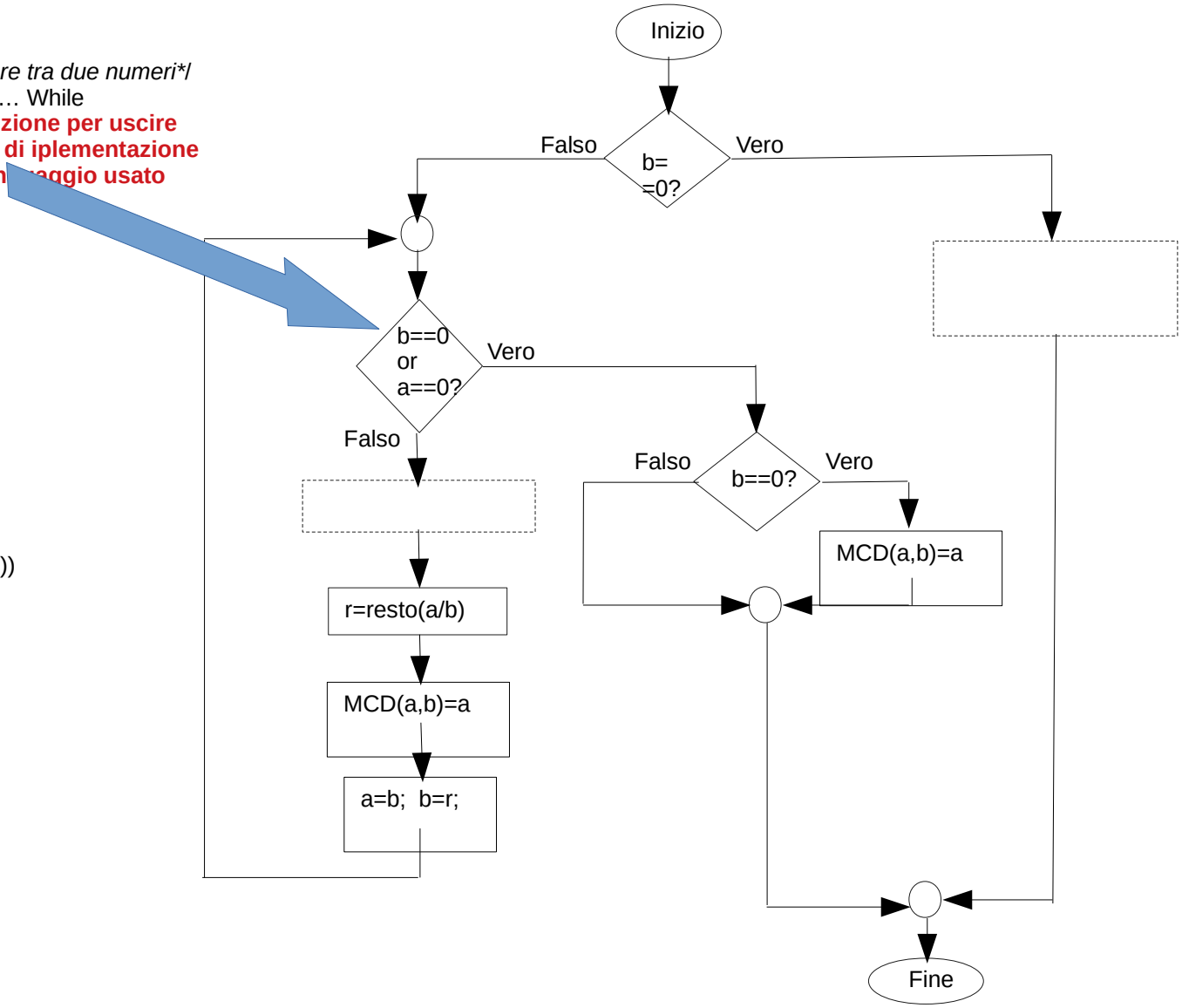
Trasformiamolo in algoritmo strutturato



Codice Matlab

% Calcolo del Massimo Comune Divisore tra due numeri/*
% N.B. in Matlab non esiste il ciclo DO … While
%N.B. In Matlab, come in C, la condizione per uscire dal ciclo è FALSE. Modificare in fase di iplementazione a seconda delle caratteristiche del linguaggio usato

```
clear all;
a=2184;
b=43;
r=0;
Mcd=0;
errore =0;
 if (b==0)
          if(a==0)
                    errore=1
          else
                    Mcd=a;
          end
          else
                    while((b~=0) && (a ~=0))
                              if (b>a)
                                        c=a;
                                        a=b;
                                        b=c;
                              end
                              r=mod(a,b);
                              Mcd=a;
                              a=b;
                              b=r;
           end
                    if (b==0)
                              Mcd=a;
                     end
          end
          sprintf("mcd=%12d\n", Mcd)
```



Codice Matlab

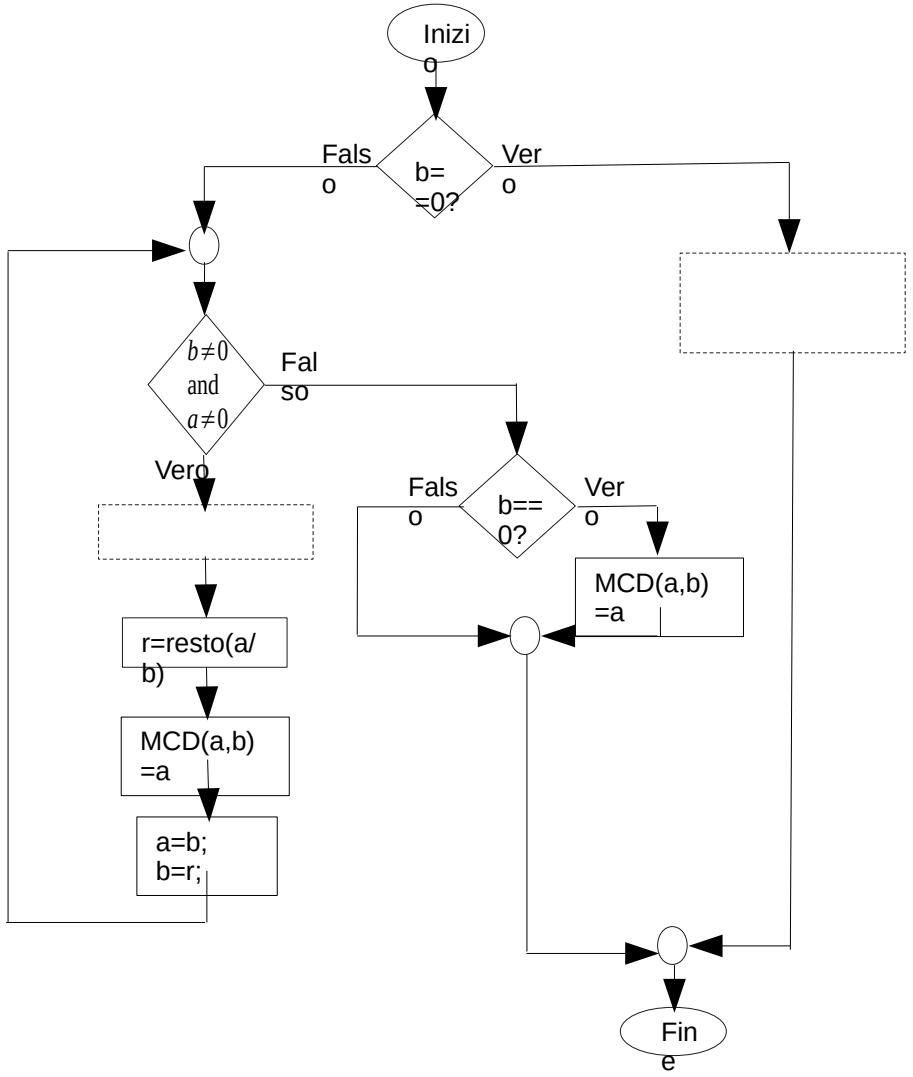
Se u e w sono variabili booleane si ha che (teorema di De Morgan):

$\overline{u \text{ or } w} = \bar{u} \text{ and } \bar{w}$

$\overline{u \text{ and } w} = \bar{u} \text{ or } \bar{w}$

Da cui si ottiene

$\overline{(b=0 \text{ or } a=0)} = (b \neq 0 \text{ and } a \neq 0)$



UNIVERSITÀ DEGLI STUDI DI TRIESTE

Strutture in Matlab

In MATLAB sono disponibili le 2 strutture di controllo di base: selezione (if) e iterazione (while), più altre 2 strutture supplementari, la selezione multipla (switch) e iterazione con conteggio (for)

```
if condizione (logico/Booleano)
  % Caso di condizione vera
  istruzioni
else
  % Caso di condizione falsa
  istruzioni
end

switch espressione
  case espr1
    % Caso espressione = espr1
    istruzioni
  case espr2
    % Caso espressione = espr2
    istruzioni
  ...
  otherwise
    % Caso complementare
    istruzioni
end
```

```
while condizione
  % Caso di condizione vera
  istruzioni
end

for indice = valori
  % indice = valore scalare
  istruzioni
end
```

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Strutture in Matlab

All’interno dei cicli while e for è possibile passare direttamente all’iterazione successiva, saltando le istruzioni rimanenti, o uscire del tutto dal ciclo attraverso i comandi

- **continue**
- **break**

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Esercizio

- Scrivere uno script che acquisisca da tastiera due interi $n \geq 0$ e $b > 1$ e determini le cifre di n in base b, formattando opportunamente il risultato.

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Esercizio

- Scrivere uno script che acquisisca da tastiera due interi n ≥ 0 e b > 1 e determini le cifre di n in base b, formattando opportunamente il risultato.
- funzioni **input** e **fprintf** per l'I/O dei dati
- algoritmo delle divisioni intere successive per eseguire il calcolo utilizzando il ciclo **while**:

```
% Input dei dati
n = input('Inserire il numero intero positivo n: ') ;
b = input('Inserire la base b>1: ') ;

% Algoritmo delle divisioni successive
q = n ; % Quoziente di lavoro
a = 0 ; % Vettore delle cifre calcolate
i = 0 ; % Indice di iterazione
while q > 0
  i    = i + 1 ;
  a(i) = mod( q , b ) ;      % resto della divisione
  q    = ( q - a(i) ) / b ; % quoziente
end

% Output formattato
fprintf( '(%d)_10 = (' , n ) ; ...
fprintf( '%d' , a(end:-1:1) ) ; ...
fprintf( ')_%d\n' , b ) ;
```

Esercizio

- Scrivere uno script che acquisisca da tastiera
  - un vettore di coefficienti $\mathbf{a} = (a_0, a_1, \ldots, a_n)$
  - un intervallo $[b, c]$
  - un intero N

  e calcoli il polinomio

$$\begin{aligned} p(x) &= a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n \\ &= a_0 + x(a_1 + x(\cdots a_{n-1} + x(a_n))), \end{aligned}$$

  sfruttando l'ultima forma, su N valori di x equidistanziati in $[b, c]$.
- Utilizzeremo un ciclo `for` per calcolare esplicitamente $p(x)$:

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Esercizio

```
% Input dei dati
a  = input('Inserire il vettore dei coefficienti: ') ;
bc = input('Inserire l''intervallo [b,c] in forma di vettore: ') ;
N  = input('Inserire il numero di valori di x: ') ;
n  = length(a) - 1 ; % grado del polinomio

% Vettore riga degli N valori di x da b=bc(1) a c=bc(2)
x = linspace( bc(1) , bc(2) , N ) ;

% Calcolo del polinomio
p = zeros( 1 , N ) ; % inizializzazione polinomio a 0
for i = n : -1 : 0
    p = a(i+1) + x .* p ; % gli indici dei vettori partono sempre da 1
end
```

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Energia

Energia di un segnale tempo-continuo in un intervallo [a,b]

$$E_x = \varepsilon \int_a^b |x(t)|^2 dt$$

Dato che eseguiamo il calcolo numericamente, dobbiamo utilizzare un algoritmo per e:

$$\int_a^b f(x)dx \approx \sum_{i=0}^{i=n-1} (x_{i+1}-x_i)\frac{1}{2}[f(x_{i+1})+f(x_i)]$$

Se dividiamo [a,b] in parti n parti uguali

$$x_{i+1}-x_i = \Delta x = \frac{b-a}{n}$$

Otteniamo

$$\int_a^b f(x)dx \approx \frac{1}{2}\cdot\frac{b-a}{n}\sum_{i=0}^{i=n-1}[f(x_{i+1})+f(x_i)]$$

$$\sum_{i=0}^{i=n-1}[f(x_{i+1})+f(x_i)] = \sum_{i=0}^{i=n-1} f(x_{i+1}) + \sum_{i=0}^{i=n-1} f(x_i) = \sum_{i=0}^{i=n-1} f(x_i) + f(b) + \sum_{i=0}^{i=n-1} f(x_i)$$

$$-f(a)+2\sum_{i=0}^{i=n-1} f(x_i) = -f(a)+2\left(\sum_{i=0}^{i=n} f(x_i) - f(b)\right) = -f(a)-2f(b)+2\left(\sum_{i=0}^{i=n} f(x_i)\right)$$

$$\int_a^b f(x)dx \approx \frac{(b-a)}{n}\left(\sum_{i=0}^{i=n} f(x_i) - \frac{1}{2}f(a) - f(b)\right)$$

- tri.m
- % Create a triangular pulse with width 0.4.
- % fs = 10000; % Sampling frequency (samples/sec)
- % t = -1:1/fs:1; % Time Vector
- % w = .4; % Triangle Width

```
fs = 10000;      % Sampling frequency (samples/sec)
t = -1:1/fs:1;   % Time Vector
w = .4;          % Triangle Width
x = tripuls(t-0.5,w);  % Sampled aperiodic triangle
plot(t,x);
xlabel('Time (sec)');ylabel('Amplitude')
```

- % See also SQUARE, S
- % RECTPULSE,