ALGORITMI

1. GLI ALGORITMI

Un **algoritmo** è la descrizione del percorso risolutivo di un problema per giungere dai dati iniziali ai risultati finali.
Scriviamo l'algoritmo pensando di rivolgerci a un **esecutore**, capace di svolgere azioni descritte da **istruzioni**, scritte in un particolare **linguaggio**. Ipotizziamo poi che l'esecutore sia a disposizione di un **utente** (non necessariamente chi ha scritto l'algoritmo) che si serve dell'esecuzione dell'algoritmo.

SUPPONIAMO CHE L'ESECUTORE SAPPIA COMPRENDERE ED ESEGUIRE	
Istruzioni di	**per**
input	ricevere dati (numeri, espressioni, testi....)
output	mandare messaggi e comunicare risultati
assegnazione	memorizzare un dato associandolo al nome di una variabile
calcolo	svolgere operazioni fra dati

ESEMPIO

1. Un modo di comunicare a un esecutore un possibile algoritmo relativo al **calcolo del perimetro di un rettangolo**, note le misure b della base e h dell'altezza, è il seguente.

   Inizia; acquisisci il valore di b e quello di h; calcola il doppio di b e assegnalo a $D1$; calcola il doppio di h e assegnalo a $D2$; somma i valori di $D1$ e $D2$ e assegna il risultato a P; rendi noto il valore di P; hai finito.

2. Una possibile descrizione dell'algoritmo per ottenere la **misura di un cateto** c in un triangolo rettangolo, conoscendo quelle dell'ipotenusa a e dell'altro cateto b, è la seguente.

   Inizia; acquisisci a e b; controlla se a è maggiore di b: se è vero calcola $\sqrt{a^2 - b^2}$, assegna il valore dell'espressione a c e comunica il suo valore, se è falso scrivi un messaggio di errore; hai finito.

3. Algoritmo per trovare il **massimo fra quattro numeri**.

   Inizia; leggi il primo numero e memorizzalo nella variabile M; ripeti queste istruzioni per 3 volte: [leggi un numero e memorizzalo nella variabile N, se N è maggiore di M, assegna il valore di N a M]; comunica M; hai finito.

Abbiamo scritto gli algoritmi precedenti nel linguaggio di tutti i giorni, ma è opportuno utilizzare un linguaggio convenzionale, che elimini le possibili ambiguità del linguaggio comune e metta in evidenza la struttura degli algoritmi.

2. IL LINGUAGGIO DI PROGETTO

Esaminiamo gli algoritmi degli esempi precedenti, scritti in un **linguaggio di progetto**.

ESEMPIO

1. Calcolo del perimetro di un rettangolo

◀ Figura 1

```
Inizio
SCRIVI "Fornisci la misura della base"
LEGGI b
SCRIVI "Fornisci la misura dell'altezza"
LEGGI h
D1 := 2 · b
D2 := 2 · h
P := D1 + D2
SCRIVI "La misura del perimetro è"
SCRIVI P
Fine
```

Notiamo che:

- il simbolo := indica l'assegnazione;
- con le istruzioni del tipo SCRIVI "…" chiediamo all'esecutore di rivolgersi all'utente;
- le tre istruzioni di assegnazione possono essere sostituite da un'unica istruzione, $P := 2 \cdot b + 2 \cdot h$, se l'esecutore sa utilizzare le espressioni.

In questo algoritmo le istruzioni sono da eseguire sempre e una sola volta, nell'ordine in cui si presentano. Una struttura come questa è detta *sequenza*.

2. La misura di un cateto

◀ Figura 2

```
Inizio
SCRIVI "Dai la misura dell'ipotenusa"
LEGGI a
SCRIVI "Dai la misura di un cateto"
LEGGI b
SE a > b
    ALLORA
        c := √(a² − b²)
        SCRIVI "L'altro cateto misura"
        SCRIVI c
    ALTRIMENTI
        SCRIVI "Errore"
Fine
```

In questo algoritmo l'istruzione $c := \sqrt{a^2 - b^2}$ è eseguita soltanto se la condizione $a > b$ è vera, l'istruzione SCRIVI "Errore" solo se la condizione è falsa.
Una struttura come questa è detta *selezione*.

3. Il massimo fra quattro numeri

Possiamo scrivere l'algoritmo in due modi diversi.

```
Inizio
SCRIVI "Fornisci il primo numero"
LEGGI M
CONT := 1
RIPETI
    CONT := CONT + 1
    SCRIVI "Fornisci un altro numero"
    LEGGI N
    SE N > M
        ALLORA M := N
FINCHÉ CONT = 4
SCRIVI "Il massimo è"
SCRIVI M
Fine
```

a

```
Inizio
SCRIVI "Fornisci il primo numero"
LEGGI M
CONT := 1
MENTRE CONT < 4 FAI
    CONT := CONT + 1
    SCRIVI "Fornisci un altro numero"
    LEGGI N
    SE N > M
        ALLORA  M := N
FINE MENTRE
SCRIVI "Il massimo è"
SCRIVI M
Fine
```

b

▲ **Figura 3**

Notiamo che:

- RIPETI *istruzioni* FINCHÉ *condizione*
  continua a far eseguire le *istruzioni* fintanto che la *condizione* è falsa; quando la *condizione* diventa vera l'esecutore prosegue con l'istruzione successiva alla riga di FINCHÉ;
- MENTRE *condizione* FAI *istruzioni* FINE MENTRE
  continua a far eseguire le *istruzioni* fintanto che la *condizione* è vera; quando la *condizione* diventa falsa l'esecutore non esegue le *istruzioni*, ma passa all'istruzione successiva alla riga di FINE MENTRE;
- CONT ha la funzione di variabile contatore: assume i valori 1, 2, 3, 4.

Una struttura descritta da RIPETI…FINCHÉ… o da MENTRE… FAI…, in cui una o più istruzioni vengono eseguite in modo ciclico, viene detta *iterazione*.

In linguaggio di progetto lo stesso algoritmo può anche essere scritto come nella figura 4.

◀ **Figura 4**

```
Inizio
SCRIVI "Fornisci il primo numero"
LEGGI M
PER i CHE VA DA 1 A 3 CON PASSO 1 FAI
    SCRIVI "Fornisci un altro numero",
    LEGGI N
    SE N > M
        ALLORA M := N
PROSSIMO i
SCRIVI "Il massimo è"
SCRIVI M
Fine
```

In questo caso supponiamo che l'esecutore abbia la capacità di gestire un contatore in modo autonomo.

Una struttura descritta da PER… CHE VA DA… A… CON PASSO… è detta **iterazione enumerativa**.

In generale, abbiamo le seguenti strutture.

Sequenza

```
Inizio
<istruzione>
<istruzione>
<istruzione>
    …
Fine
```

Selezione

```
SE <condizione>
        ALLORA
        <istruzioni1>
        ALTRIMENTI
        <istruzioni2>
```

dove il blocco <istruzioni1> viene eseguito se <condizione> è vera, il blocco <istruzioni2> viene eseguito se <condizione> è falsa.

È possibile anche la seguente struttura di selezione in cui non è presente ALTRIMENTI:

```
SE <condizione>
        ALLORA
        <istruzioni1>
```

Iterazione

```
RIPETI
    <istruzioni>
FINCHÉ <condizione>
```

dove si esegue il blocco <istruzioni>, se <condizione> è vera si esce dal ciclo, se invece <condizione> è falsa si continua il ciclo, tornando a eseguire <istruzioni>; e così via.

Nella struttura RIPETI… FINCHÉ… il blocco <istruzioni> viene eseguito almeno una volta, anche se <condizione> è già vera all'inizio del ciclo.

```
MENTRE <condizione> FAI
    <istruzioni>
FINE MENTRE
```

dove se <condizione> è vera si esegue <istruzioni>, se invece <condizione> è falsa esce dal ciclo.

Per comprendere meglio la differenza delle due strutture puoi farne un confronto nel seguente esempio.

ESEMPIO

RIPETI mangia un biscotto FINCHÉ sei sazio	MENTRE non sei sazio FAI mangia un biscotto FINE MENTRE

◄ Figura 5

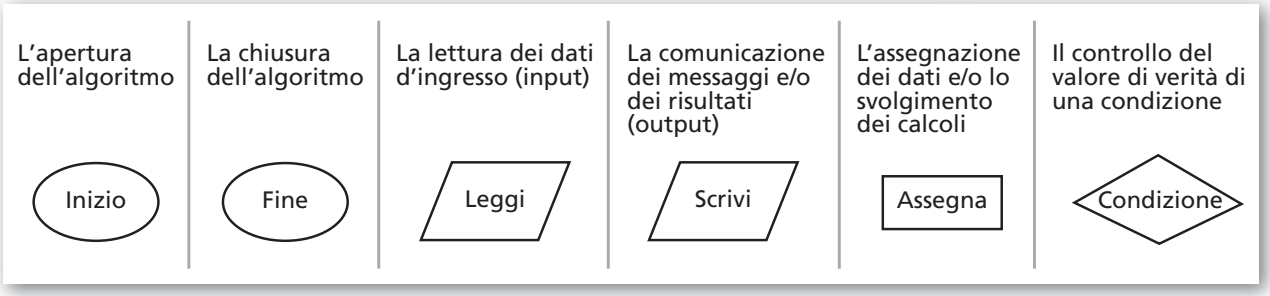
PER <indice> CHE VA DA <iniziale> A <finale> CON PASSO <delta> FAI
<istruzioni>
PROSSIMO <indice>

dove la variabile indicata in <indice>, partendo dal valore espresso in <iniziale>, aumenta della quantità <delta> per ogni ciclo; quando <indice> supera il valore indicato da <finale> si esce dal ciclo.

3. I DIAGRAMMI A BLOCCHI

Un altro modo per descrivere gli algoritmi è quello, di tipo grafico, dei diagrammi a blocchi.

Qui di seguito vediamo le forme convenzionali dei blocchi.

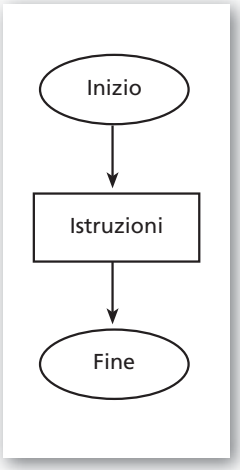


▲ Figura 6

Le tre strutture fondamentali degli algoritmi possono essere descritte così con i diagrammi a blocchi. I blocchi vengono rappresentati uniti da frecce, che indicano il flusso di esecuzione dell'algoritmo.

Sequenza

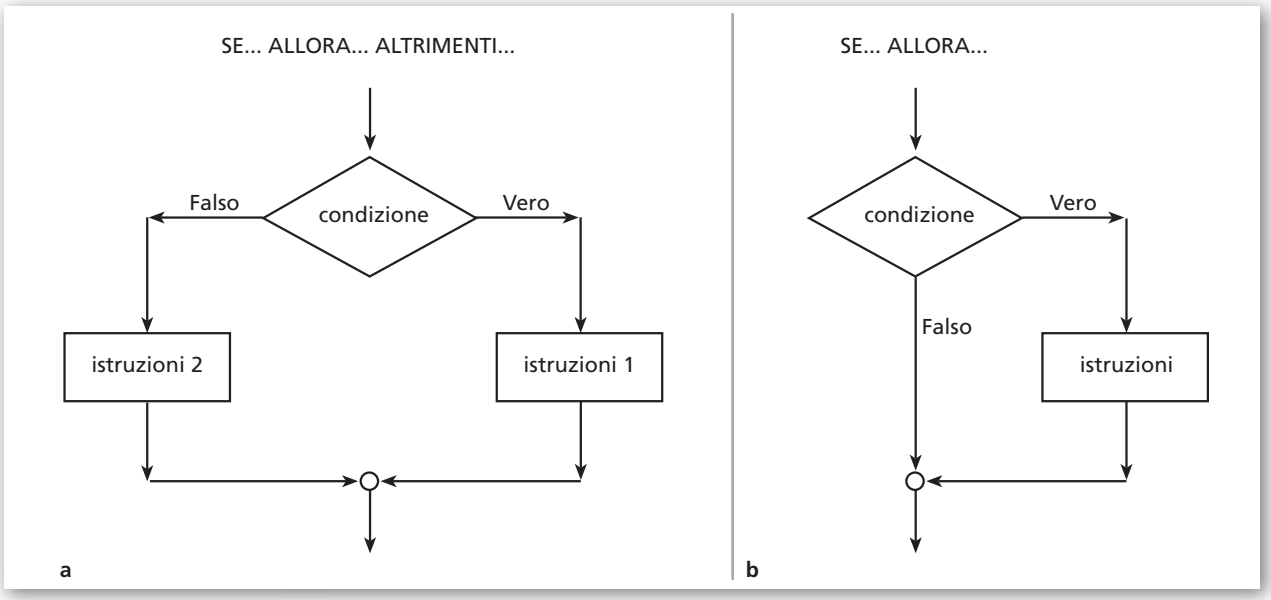
I blocchi di apertura e chiusura dell'algoritmo hanno una sola sola freccia, rispettivamente uscente ed entrante, mentre i blocchi intermedi hanno generalmente una freccia entrante e una uscente (figura 7).



► Figura 7

Selezione

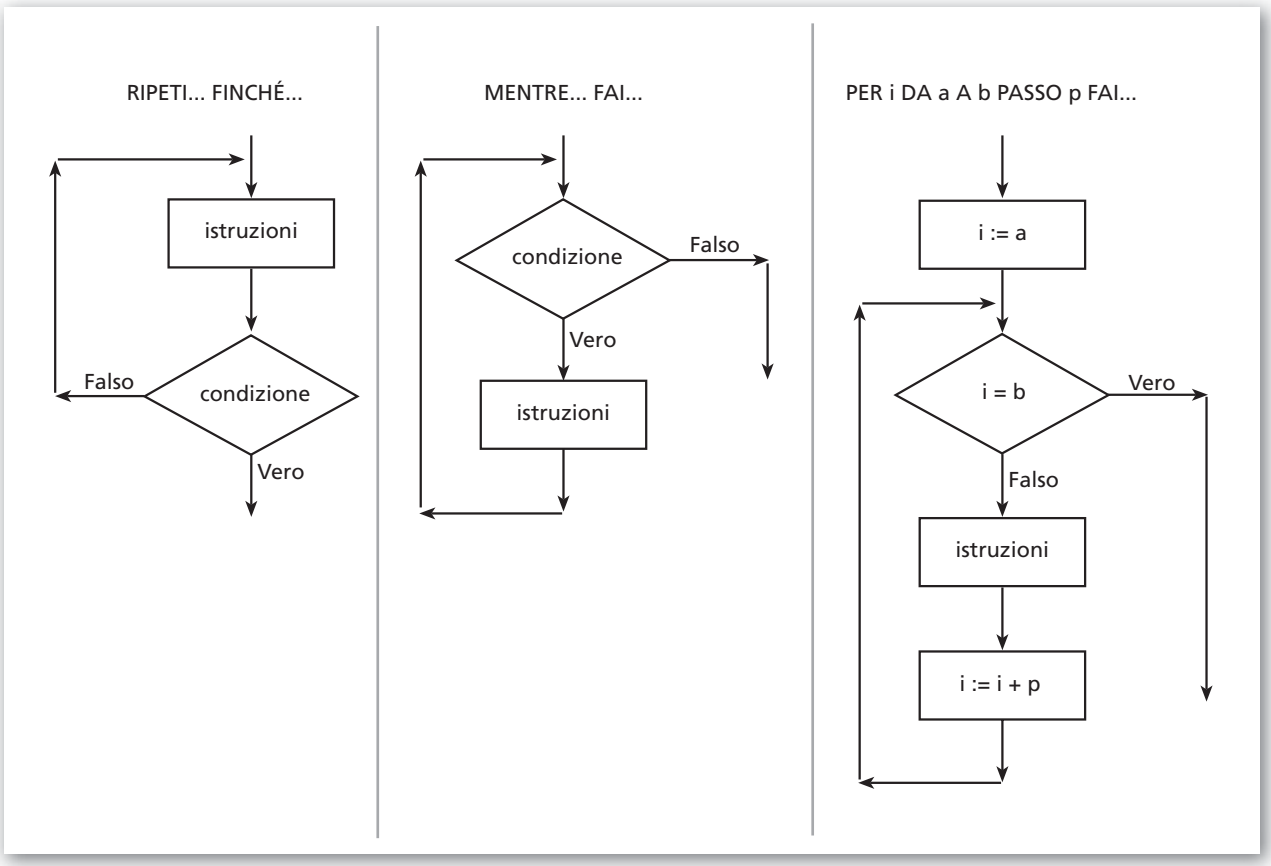
Il blocco di controllo, nel caso più semplice, ha una freccia entrante e due uscenti, corrispondenti ai due valori di verità della condizione che viene valutata (figura 8).



▲ Figura 8

▼ Figura 9

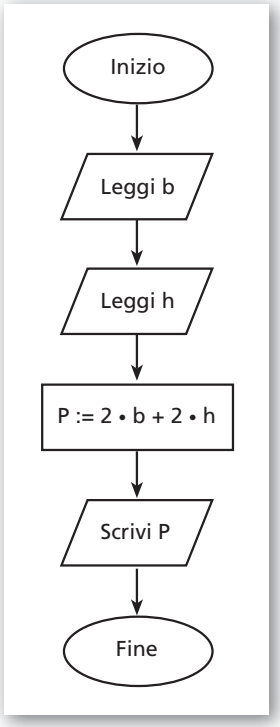
Iterazione



Costruiamo i diagrammi a blocchi degli esempi precedenti.

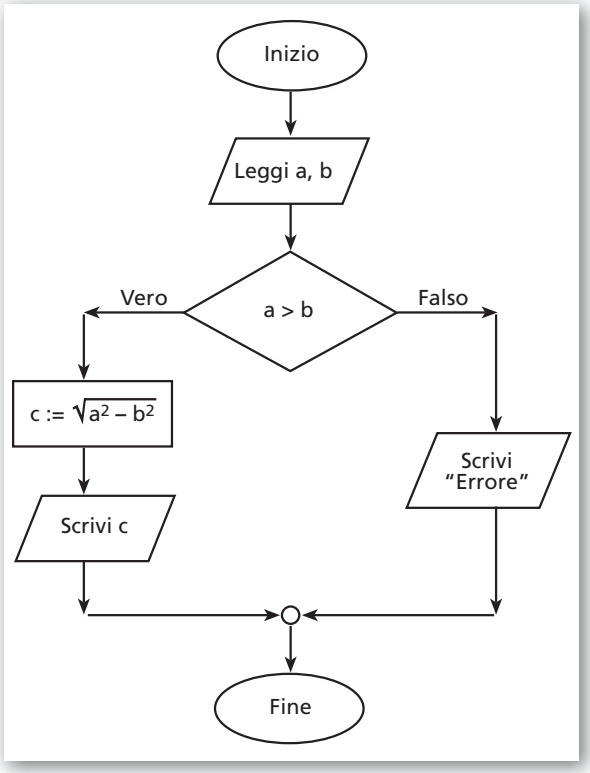
ESEMPIO

1. Il perimetro di un rettangolo



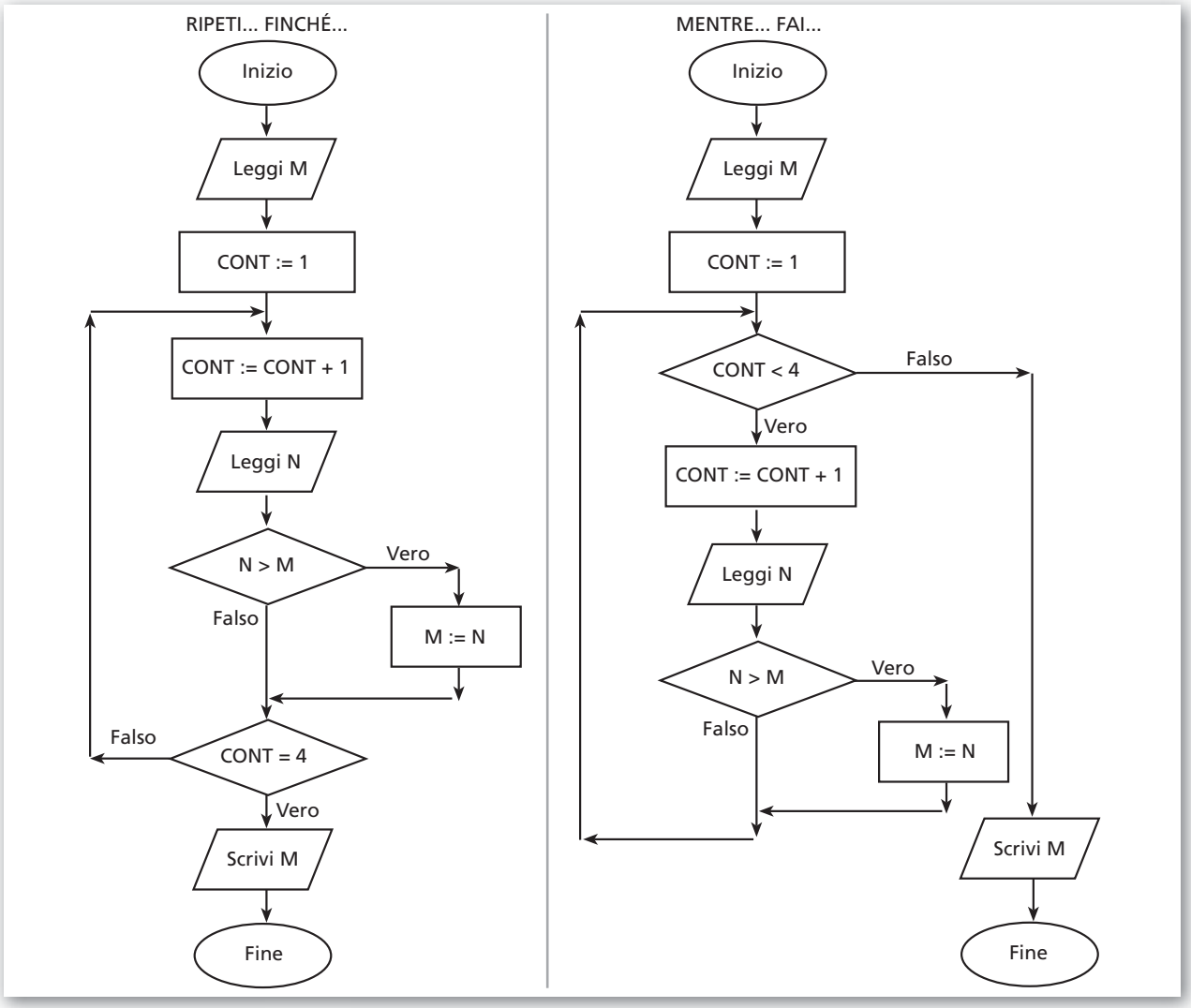
◀ Figura 10

2. La misura di un cateto



◀ Figura 11

3. Il massimo fra quattro numeri



▲ Figura 12

4. L'ALGORITMO DI EUCLIDE

Nell'insieme dei numeri naturali, cerchiamo il massimo comune divisore fra due numeri m e n che indichiamo con M.C.D.(m, n).
Possiamo sfruttare il seguente teorema.

TEOREMA

Supposto $m \geq n$, se m e n hanno un divisore d comune, d è divisore anche di $m - n$.

DIMOSTRAZIONE

Poiché d è divisore sia di m sia di n, abbiamo che

$$m = kd \quad \text{e} \quad n = hd.$$

La differenza $m - n$ è allora:

$$m - n = kd - hd.$$

Raccogliamo d al secondo membro:

$$m - n = d(k - h).$$

Quindi anche $m - n$ ha d come divisore.

Per il teorema precedente, i divisori comuni a m e n sono gli stessi comuni a $m - n$ e n e quindi:

$$\text{M.C.D.}(m, n) = \text{M.C.D.}(m - n, n).$$

Possiamo allora determinare il M.C.D. fra due numeri per sottrazioni successive. Iniziamo confrontando i due numeri, se il primo è più piccolo scambiamo il primo con il secondo, quindi eseguiamo la sottrazione fra i due numeri. Confrontiamo poi il secondo numero con la differenza, se è necessario li scambiamo, eseguiamo la sottrazione… Proseguiamo ottenendo una sequenza di numeri che hanno il medesimo M.C.D. e, per come sono calcolati, sono numeri naturali sempre più piccoli. Giungiamo quindi a 0 e a quel punto, essendo M.C.D.$(0, n) = n$, diciamo che il numero precedente è il M.C.D. cercato.

Scriviamo questo **algoritmo per la ricerca del M.C.D. con le sottrazioni successive** in linguaggio di progetto, sia con RIPETI… FINCHÉ… sia con MENTRE… FAI.

▼ Figura 13

```
Inizio
SCRIVI "PRIMO NUMERO:"
LEGGI m
SCRIVI "SECONDO NUMERO:"
LEGGI n
RIPETI
      SE m < n
          ALLORA
                temp := m
                m := n
                n := temp
      m := m − n
FINCHÉ m = 0
SCRIVI n
Fine
```

a

```
Inizio
SCRIVI "PRIMO NUMERO:"
LEGGI m
SCRIVI "SECONDO NUMERO:"
LEGGI n
MENTRE m > 0  FAI
      SE m < n
          ALLORA
                temp := m
                m := n
                n := temp
      m := m − n
SCRIVI n
Fine
```

b

Osserva che:

- per effettuare lo scambio fra i numeri m e n abbiamo bisogno di una terza variabile che chiamiamo *temp*;
- l'istruzione $m := m - n$ richiede il calcolo della differenza fra m e n e la memorizzazione del risultato in m. Il precedente valore di m va perso.

La tavola di traccia

Per capire meglio quali azioni fa compiere un algoritmo e in quale sequenza, possiamo compilare la **tavola di traccia**, cioè una tabella dei valori assunti dalle variabili e dei risultati dei test logici legati alle condizioni, percorrendo l'algoritmo passo a passo.

Compiliamo la tavola di traccia dell'algoritmo precedente, utilizzando la versione con RIPETI… FINCHÉ…, e commentiamola.

► Tabella 1

Passo	m	n	$m < n$	$temp$	m	n	$m := m - n$	$m = 0$	M.C.D.
1	15	9	Falso				6	Falso	
2	6	9	Vero	6	9	6	3	Falso	
3	3	6	Vero	3	6	3	3	Falso	
4	3	3	Falso				0	Vero	3

Passo 1 Partiamo con i numeri $m = 15$ e $n = 9$. La condizione $m < n$ è falsa, quindi calcoliamo la differenza e l'assegniamo a m. La condizione $m = 0$, è falsa, quindi ripetiamo le istruzioni del ciclo.

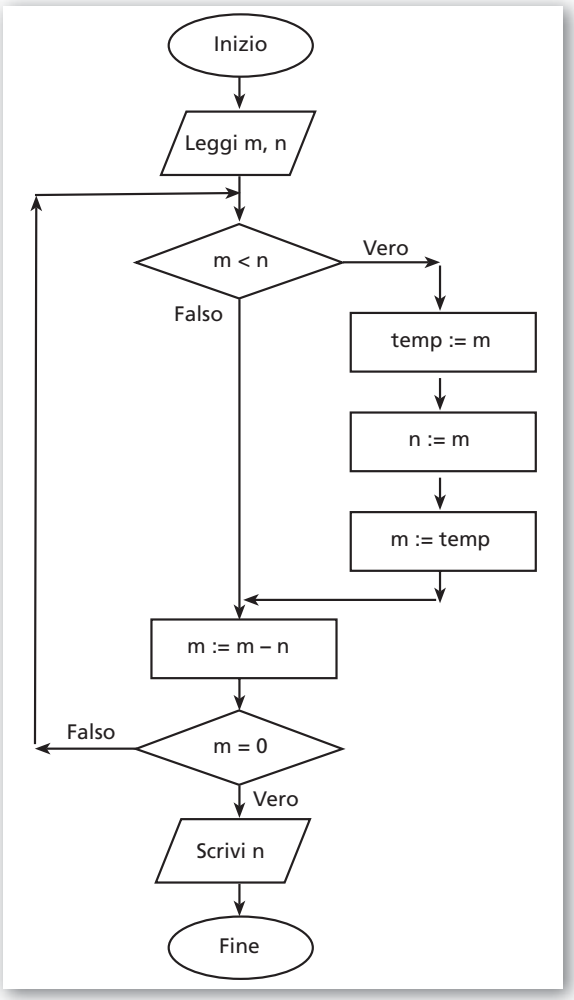
Passo 2 La condizione $m < n$ è vera, pertanto scambiamo i valori di m e di n. Effettuiamo la differenza e l'assegniamo a m. Poiché m è diverso da 0, ripetiamo il ciclo.

Passo 3 Proseguiamo in modo simile.

Passo 4 Troviamo che m vale 0, quindi usciamo dal ciclo e scriviamo il valore contenuto in n, che è il M.C.D. cercato.

L'algoritmo di Euclide con il diagramma a blocchi

Se usiamo un diagramma a blocchi, l'algoritmo delle differenze successive è quello rappresentato nella figura 14.

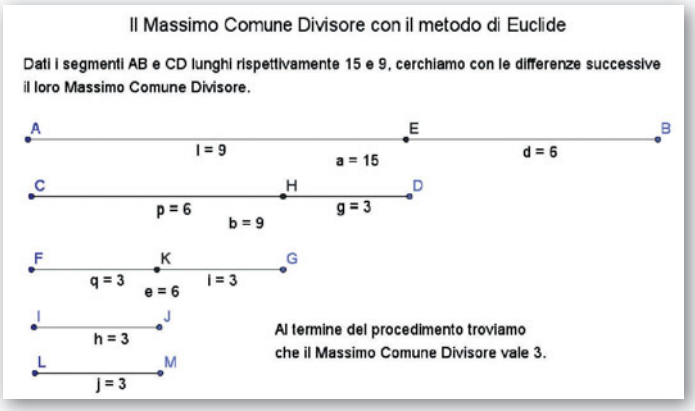


► Figura 14

L'ALGORITMO DI EUCLIDE CON GEOGEBRA

Euclide illustrò l'algoritmo che prende il suo nome in modo geometrico. Partiva da due segmenti, la cui lunghezza era espressa da due numeri naturali, e procedeva sottraendo ripetutamente il più corto dal più lungo sino a pervenire a un segmento di misura nulla.
La lunghezza del segmento immediatamente precedente era contenuta in modo esatto in quelle dei segmenti dati ed era la più grande a godere di tale proprietà, cioè rappresentava geometricamente il numero naturale M.C.D. di quelli relativi alle due lunghezze iniziali.

Per esempio, seguiamo il procedimento geometrico di Euclide con GeoGebra (figura a fianco).
Inseriamo nell'area del disegno con *Rette_Segmento di data misura* i due segmenti *AB* e *CD* di misure intere rispettivamente 15 e 9.
Con *Circonferenza_Compasso* riportiamo sul segmento *AB* il segmento *CD*, trovando il segmento differenza *EB*, che spostiamo in *FG* con *Rette_Segmento di data misura*. Riportiamo *FG* su *CD*, ricavando il segmento differenza *HD*, che spostiamo in *IJ*.



Riportiamo *IJ* su *FG* ricavando il segmento differenza *KG*, che spostiamo in *LM*. Riportiamo *LM* su *IJ* ricavando il segmento nullo; pertanto la misura di *LM*, uguale a 3, è il M.C.D.

5. L'ALGORITMO PER LA RICERCA DEL M.C.D. CON LE DIVISIONI SUCCESSIVE

Ci sono algoritmi poco efficienti, che richiedono un alto numero di passaggi e moltissimo tempo per essere eseguiti. Per esempio, l'algoritmo delle differenze successive per la ricerca del M.C.D., se i due numeri forniti sono grandi, può richiedere numerosissimi passaggi.
Abbiamo un'esecuzione più rapida se il nostro esecutore sa fare la divisione con resto e sa quindi calcolare il resto fra la divisione di due numeri naturali m e n.
In questo caso possiamo sfruttare il seguente teorema.

TEOREMA

Supposto $m \geq n$, se m e n hanno un divisore d comune, d è divisore anche di r e n, dove r è il resto della divisione di m per n.

DIMOSTRAZIONE

Poiché d è divisore sia di m sia di n, abbiamo:

$$m = kd \quad \text{e} \quad n = hd.$$

Per la definizione di divisione:

$r = m - qn$, dove q è il quoziente e r è il resto della divisione di m per n.

Sostituendo le espressioni di m e n ricaviamo:

$r = kd - qhd \quad \rightarrow \quad r = d(k - hq)$,

da cui deduciamo che d è un divisore di r.

Per il teorema precedente, m e n hanno gli stessi divisori comuni di r e n e quindi anche lo stesso M.C.D.
Possiamo allora applicare questa proprietà più volte per determinare per mezzo di divisioni successive il M.C.D. di due numeri.

Partiamo dai numeri dati e determiniamo il resto della loro divisione. Calcoliamo poi il resto della divisione fra il secondo numero e il resto precedente e così via. Proseguiamo ottenendo una successione di numeri che hanno lo stesso M.C.D. Osserviamo che i numeri sono sempre più piccoli, in quanto il resto di ogni divisione è minore del dividendo. Quindi, prima o poi si raggiunge lo 0. A quel punto, essendo M.C.D.$(r, 0) = r$, diciamo che l'ultimo resto diverso da 0 è il M.C.D. cercato.

Scriviamo l'**algoritmo per la ricerca del M.C.D. con le divisioni successive**, in linguaggio di progetto, indicando con *mod* l'operatore che restituisce il resto della divisione fra due numeri.

▼ Figura 15

```
Inizio
SCRIVI "PRIMO NUMERO:"
LEGGI m
SCRIVI "SECONDO NUMERO:"
LEGGI n
RIPETI
  r := mod(m, n)
  m := n
  n := r
FINCHÉ r = 0
SCRIVI m
Fine
```

a

```
Inizio
SCRIVI "PRIMO NUMERO:"
LEGGI m
SCRIVI "SECONDO NUMERO:"
LEGGI n
MENTRE n > 0 FAI
   r := mod(m, n)
  m := n
  n := r
SCRIVI m
Fine
```

b

Compiliamo la tavola di traccia dell'algoritmo precedente, utilizzando la versione con RIPETI… FINCHÉ…, e commentiamola.

► Tabella 2

Passo	m	n	r	m	n	$r = 0$	M.C.D.
1	15	9	6	9	6	Falso	
2	9	6	3	6	3	Falso	
3	6	3	0	3	0	Vero	3

Passo 1 15 diviso 9 ha resto 6, diamo il valore 9 contenuto in n a m e il valore 6 contenuto in r a n; poiché il resto non è 0, continuiamo il ciclo.

Passo 2 Procediamo in modo analogo, tenendo presente che adesso m vale 9 e n vale 6.

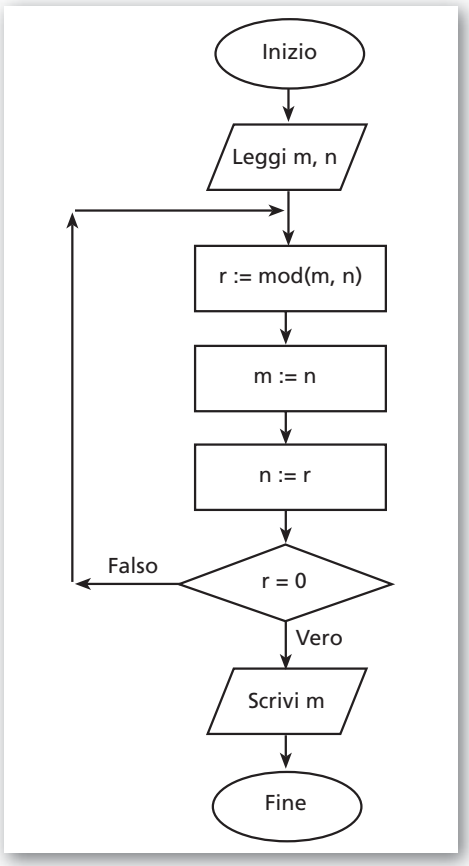
Passo 3 Il resto vale 0, quindi usciamo dal ciclo e abbiamo il M.C.D. in m.

È da notare che l'algoritmo funziona anche se all'inizio forniamo i dati scambiati: $m = 9$ e $n = 15$, perché il resto della divisione fra 9 e 15 è proprio 9.
La prima riga della tavola di traccia cambia, ma poi si riprende con la tavola di traccia precedente.

◄ Tabella 3

Passo	m	n	r	m	n	$r = 0$	M.C.D.
1	9	15	9	15	9	Falso	
2	15	9	6	9	6	Falso	
3	9	6	3	6	3	Falso	
4	6	3	0	3	0	Vero	3

Se usiamo un diagramma a blocchi, l'algoritmo è quello rappresentato nella figura 16.



► Figura 16

6. DALL'ALGORITMO AL PROGRAMMA

Se traduciamo un algoritmo con un **linguaggio di programmazione** utilizzabile da un computer, che è un particolare esecutore, otteniamo un **programma**.
Forniamo un esempio di traduzione dell'algoritmo di Euclide, sia con sottrazioni sia con divisioni successive, con il linguaggio Python.

Per scrivere il programma, facciamo riferimento alla versione 2.7 di Python, che puoi scaricare gratuitamente dal sito ufficiale del linguaggio e installare sul tuo computer.

ESEMPIO

Dal menu *Start* di Windows, scegli l'opzione IDLE di Python. Nel menu *File*, scegli *New Window*. Nel foglio che compare, scrivi le istruzioni seguenti (quelle di sinistra per l'algoritmo delle differenze successive, quelle di destra per quello delle divisioni). Fai attenzione a rispettare le indentazioni.

▼ Figura 17

```
m = input ("primo numero:")
n = input ("secondo numero:")
while m > 0:
        if m < n:
                temp = m
                m = n
                n = temp
        m = m − n
print "M.C.D.:", n
```

a

```
m = input ("primo numero:")
n = input ("secondo numero:")
while n > 0:
    r = m % n
    m = n
    n = r
print "M.C.D.:", m
```

b

Scritte le istruzioni, scegli nel menu *Run* l'opzione *Run Module*, e compare una finestra in cui devi scegliere OK, scrivere poi il nome con il quale salvare il programma, terminandolo con l'estensione .py, per esempio *miomcd.py*., e salvare scegliendo *Salva*. A questo punto il programma viene eseguito.

Le procedure

Per organizzare meglio i programmi e renderli più leggibili, si possono utilizzare le **procedure**, ossia insiemi di istruzioni a cui si può dare un nome e che possono essere richiamati all'interno dei programmi. Vediamo, utilizzando ancora Python, come possiamo riscrivere i programmi dell'esempio precedente.

▼ Figura 18

ESEMPIO

```
def mcd(m, n):
    while m > 0:
        if m < n:
            temp = m
            m = n
            n = temp
        m = m − n
    print "MCD =", n
    return

m = input ("primo numero:")
n = input ("secondo numero:")
mcd(m, n)
```

a

```
def mcd(m, n):
    while n > 0:
        r = m % n
        m = n
        n = r
    print "MCD =", m
    return


m = input ("primo numero:")
n = input ("secondo numero:")
mcd(m, n)
```

b

Come vedi, l'algoritmo è sempre lo stesso, ma preferiamo prima «far imparare» al computer cosa deve fare quando gli diamo l'istruzione *mcd* e poi utilizzarla. I valori di *m* e *n*, dopo essere stati letti con le istruzioni di input, vengono forniti alla procedura assegnandoli alle variabili scritte fra le parentesi tonde, separate da una virgola.

Riportiamo anche le procedure scritte con due linguaggi diversi, quello di Derive e quello di Wiris, questa volta utilizzando RIPETI… FINCHÉ…
Nota che qui abbiamo fornito direttamente i valori alla procedura, senza istruzioni di input. Ciò è possibile anche con le procedure che abbiamo definito con Python.

ESEMPIO

1. Derive

◀ Figura 19

```
        MasComDiv_1(m, n) :=
          Prog
            Loop
              If m < n
                 Prog
#1:                 temp := m
                    m := n
                    n := temp
              m := m - n
              If m = 0 exit
            RETURN n

#2:    MasComDiv_1(15, 12)

#3:    3
```

a

```
        MasComDiv_2(m, n) :=
          Prog
            Loop
              r := MOD(m, n)
#1:           m := n
              n := r
              If r = 0 exit
            RETURN m

#2:    MasComDiv_2(15, 12)

#3:    3
```

b

2. Wiris

◀ Figura 20

```
mcd_1(m, n) := inizio
                 ripetere
                   se m < n allora
                     temp = m
                     m = n
                     n = temp
                   fine
                   m = m - n
                 fino_a_quando m = 0
                 n
               fine ;
mcd_1(15, 12) → 3
mcd_1(3015, 248) → 1
mcd_1(78, 150) → 6
```

a

```
mcd_2(m, n) := inizio
                 ripetere
                   r = resto(m, n)
                   m = n
                   n = r
                 fino_a_quando r = 0
                 m
               fine ;
mcd_2(15, 12) → 3
mcd_2(3015, 248) → 1
mcd_2(78, 150) → 6
```

b

La ricorsività

Le procedure possono utilizzare se stesse all'interno della loro definizione. Questa caratteristica si indica con il termine **ricorsività.**
Per farti comprendere come funziona, scriviamo le procedure ricorsive in Python. Puoi utilizzarle sostituendole nei programmi precedenti.

► Figura 21

```
def mcd(m, n):
    if m < n:
        mcd(n, m)
    else:
        if n == 0:
            print "MCD =", m
            return
        else:
            mcd(m − n, n)
```

a

```
def mcd(m, n):
    if n == 0:
        print "MCD =", m
        return
    else:
        mcd(n, m % n)
```

b

Nota che in Python il segno di uguale viene scritto == per distinguerlo da quello di assegnazione che è =.

Esprimiamo nel linguaggio usuale quello che abbiamo scritto nella definizione a sinistra.

Per calcolare il M.C.D. di m e n, se $m < n$, allora calcola il M.C.D. con m e n scambiati, altrimenti procedi controllando se n è uguale a 0. Se lo è, scrivi che il valore di m è il massimo comune divisore e hai finito, altrimenti calcola il M.C.D. fra $m - n$ e n.

La ricorsività non ha bisogno della struttura MENTRE… FAI… e il modo con cui la procedura richiama se stessa risulta molto simile al procedimento che abbiamo descritto a parole quando, esaminando il teorema, abbiamo indicato come potevamo sfruttarlo.

Esprimiamo nel linguaggio usuale anche la definizione a destra.

Per calcolare il M.C.D. di m e n, se n è uguale a 0, scrivi che il valore di m è il massimo comune divisore e hai finito, altrimenti calcola il M.C.D. fra n e il resto della divisione fra m e n.

ESERCIZI

Per ognuno dei seguenti problemi, relativi a numeri naturali:
- **svolgi l'analisi matematica per pervenire alla sua soluzione,**
- **scrivi con le convenzioni del linguaggio di progetto o dei diagrammi a blocchi l'algoritmo risolutivo,**
- **compila la tavola di traccia dell'algoritmo con i valori d'ingresso indicati.**

1 Dato un numero n, stabilisci il numero v dei suoi divisori, compresi 1 e il numero stesso.
Prova con $n = 19$, con $n = 54$. [2, 8]

2 Dato un numero n, stabilisci il numero φ dei numeri minori di n e primi con n, unità inclusa.
Prova con $n = 4$, con $n = 7$, con $n = 15$. [2, 6, 8]

3 Dati due numeri n e f, con f numero primo, trova la molteplicità m con la quale f appare fra i divisori di n.
Prova con $n = 162$ e con $f = 3$, con $n = 162$ e con $f = 6$. [4, f non è primo]

4 Scrivi un algoritmo che, dati 5 numeri, calcoli la loro somma. Prova con 4, 2, 6, 7, 3. [22]

5 Tre rappresentanti di commercio di una medesima ditta partono contemporaneamente dalla sede e impiegano per completare il loro giro, il primo p giorni, il secondo s giorni, il terzo g giorni. Dopo aver assegnato rispettivamente un valore a p e uno a s, stabilisci, per ogni valore di g da 1 a 6, il numero dei giorni che passano affinché si ritrovino tutti assieme nella località di partenza.
Prova con $p = 3$ giorni e $s = 5$ giorni.

6 Nella successione dei numeri di Fibonacci i primi due sono 1 e 1 e gli altri si ottengono ognuno come somma dei due che lo precedono, trova:

a) il numero n di posto i,

b) l'indice i del primo numero della successione che supera il numero i^2,

c) il numero s della successione che sia il p-esimo a essere primo.

Prova con $i = 8$ e con $p = 4$. [$n = 21$, $i = 13$, $s = 13$]

Per verificare che l'addizione e la moltiplicazione godono della proprietà associativa e che la sottrazione e la divisione non godono di tale proprietà, assegna n valori equidistanti fra loro ai parametri a e b e mostra in colonne affiancate i valori che assumono in corrispondenza le seguenti coppie di espressioni. Prova con a variabile in {8, 9, 10} e b variabile in {3, 6, 9}.

7 $(a + b) + 7$ e $a + (b + 7)$.

8 $(a \cdot 324) \cdot b$ e $a \cdot (b \cdot 324)$.

9 $(877 - a) - b$ e $877 - (a - b)$.

10 $(a : b) : 3$ e $a : (b : 3)$.

11 Se inserisci i valori indicati, quali uscite dà il seguente algoritmo? A quale domanda potrebbe rispondere l'algoritmo? L'operatore logico *or* fornisce il risultato VERO se almeno una delle due condizioni che lega è vera.

```
Inizio
LEGGI d
    SE d < 0 or d > 90
        ALLORA
            SCRIVI "Il dato non è accettabile"
        ALTRIMENTI
            α := (90 + d)/2
            β := 90 − α
            SCRIVI α e β
Fine
```

Prova con $d = 0$, $d = 10$, $d = 90$, $d = 30$, $d = 120$.

[45 e 45; 50 e 40; 90 e 0; 60 e 30; *d* non è acc.; trova due angoli complementari la cui differenza è *d*]

A quale quesito rispondono i seguenti algoritmi? Segui la traccia partendo dai valori sotto indicati.

12

```
Inizio
LEGGI n
flag = 1
cont = 0
RIPETI
    SE mod(n, 3) = 0
        ALLORA
            cont := cont + 1
            n := n/3
        ALTRIMENTI
            flag = 0
FINCHÉ flag = 0
SCRIVI cont
Fine
```

Prova con $n = 162$ e con $n = 223$. [dato il numero *n*, stabilisci la molteplicità del divisore 3; 4, 0]

13

```
Inizio
LEGGI n
PER i CHE VA DA 1 a n CON PASSO 1
LEGGI a_i
PROSSIMO i
cont = 0
PER i CHE VA DA 1 a n CON PASSO 1
    SE mod(a_i, 2) = 0
        ALLORA
            cont := cont + 1
PROSSIMO i
SCRIVI cont
Fine
```

Prova con $n = 5$ e $a_i = \{4, 8, 3, 7, 11\}$.

[determina quanti numeri pari appaiono in un insieme di *n* numeri dati; 2]

Che cosa c'è che non va nelle seguenti descrizioni di algoritmi?

14

Inizio
LEGGI a
$d := a - b$
SE $d > 0$
ALLORA
SCRIVI a, "è maggiore di", b
ALTRIMENTI
SCRIVI a, "non è maggiore di", b
Fine

[il numero b non è noto all'esecutore]

15

Inizio
LEGGI a e b
RIPETI
$m := a - b$
$n := a + b$
$a := a + 1$
$b := b + 1$
FINCHÉ $m > n$
SCRIVI m, n
Fine

[il ciclo prosegue all'infinito se $b > 0$]

Trova e correggi gli errori sostanziali contenuti nei seguenti algoritmi, in relazione a quello che il commento promette.

16 L'algoritmo trova il perimetro di un triangolo isoscele di lato l e altezza h.

Inizio
SCRIVI "Dai la misura del lato obliquo"
LEGGI l
SCRIVI "Dai la misura dell'altezza"
LEGGI h
$b := 2 \cdot \sqrt{l^2 - h^2}$
$duep := 2 \cdot b + l$
SCRIVI $duep$
Fine

[occorre fare un controllo su l e h; il calcolo del perimetro è sbagliato]

17 L'algoritmo calcola l'area di un rettangolo di base b e diagonale d.

Inizio
SCRIVI "Dai la misura della base"
LEGGI b
SCRIVI "Dai la misura della diagonale"
LEGGI d
$h := \sqrt{b^2 + d^2}$
$S := b \cdot h$
SCRIVI S
Fine

[occorre fare un controllo su b e d; il calcolo di h è sbagliato]