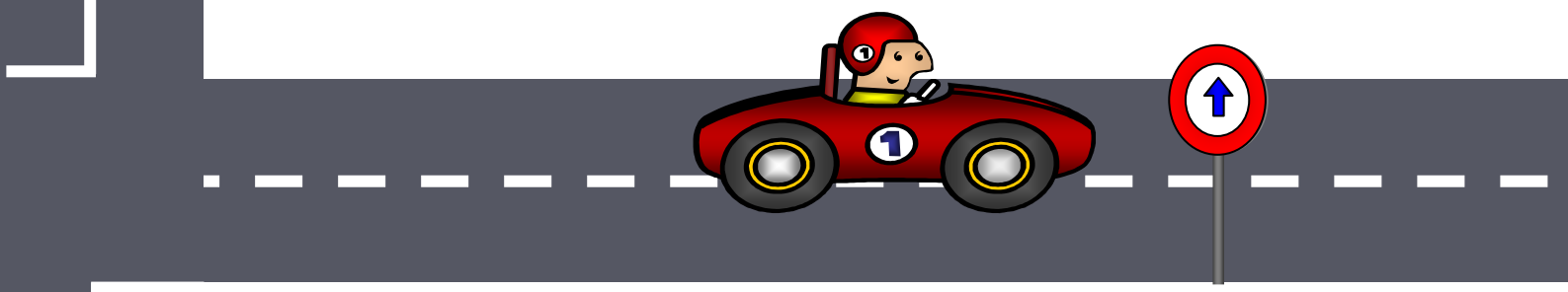
INFORMATICA



MODULO 9

Algoritmi e Linguaggio Java

A cura di Mimmo Corrado

Maggio 2012

FINALITÁ

Il Modulo Algoritmi e linguaggio Java introduce lo studente nel mondo della programmazione nel linguaggio Java.

1. FLOW-CHART E PSEUDOLINGUAGGIO

1.1 Definire il termine “algoritmo”


Al-Khwarizmi

Il termine “algoritmo” deriva dal nome del matematico arabo *Al-Khwarizmi*, vissuto nell’800 d.C., ritenuto l’ideatore del procedimento che consente di effettuare il calcolo della moltiplicazione tra due numeri mediante l’incolonnamento delle cifre (quella che ancora oggi usiamo).

Nel senso più ampio della parola, un "algoritmo" è una sequenza finita di operazioni, come ad esempio una ricetta di cucina, o le istruzioni di funzionamento di una lavatrice.

In informatica, con il termine algoritmo si intende: un procedimento (sequenza finita di istruzioni elementari) per la risoluzione di un problema, rappresentato in un linguaggio comprensibile all’uomo e adatto ad essere tradotto in un programma eseguibile da un computer.

Un algoritmo deve:

- essere finito: la sequenza di istruzioni deve essere finita e portare ad un risultato avere un punto di *Inizio*, dove si avvia l’esecuzione delle azioni, e un punto di *Fine*, dove si interrompe l’esecuzione
- essere eseguibile: le istruzioni devono poter essere eseguite materialmente dall’esecutore
- essere non ambiguo: le istruzioni devono essere espresse in modo tale da essere interpretate da tutti allo stesso modo
- essere generale: deve essere valido non solo per un particolare problema, ma per una classe di problemi
- essere deterministico: partendo dagli stessi dati iniziali deve portare sempre allo stesso risultato finale indipendentemente dall’esecutore
- essere completo: deve contemplare tutti i casi che si possono verificare durante l’esecuzione

1.2 Descrivere in forma algoritmica la procedura risolutiva di semplici problemi

Come accennato precedentemente, gli algoritmi li incontriamo e li eseguiamo quotidianamente:

- nella preparazione di un uovo al tegamino
- nella messa in moto dell’auto
- nel calcolo dell’area del rettangolo
- nel calcolo dell’M.C.D. fra due numeri naturali
- nel trasporto del lupo, della pecora e del cavolo da una sponda all’altra di un fiume
- ecc…

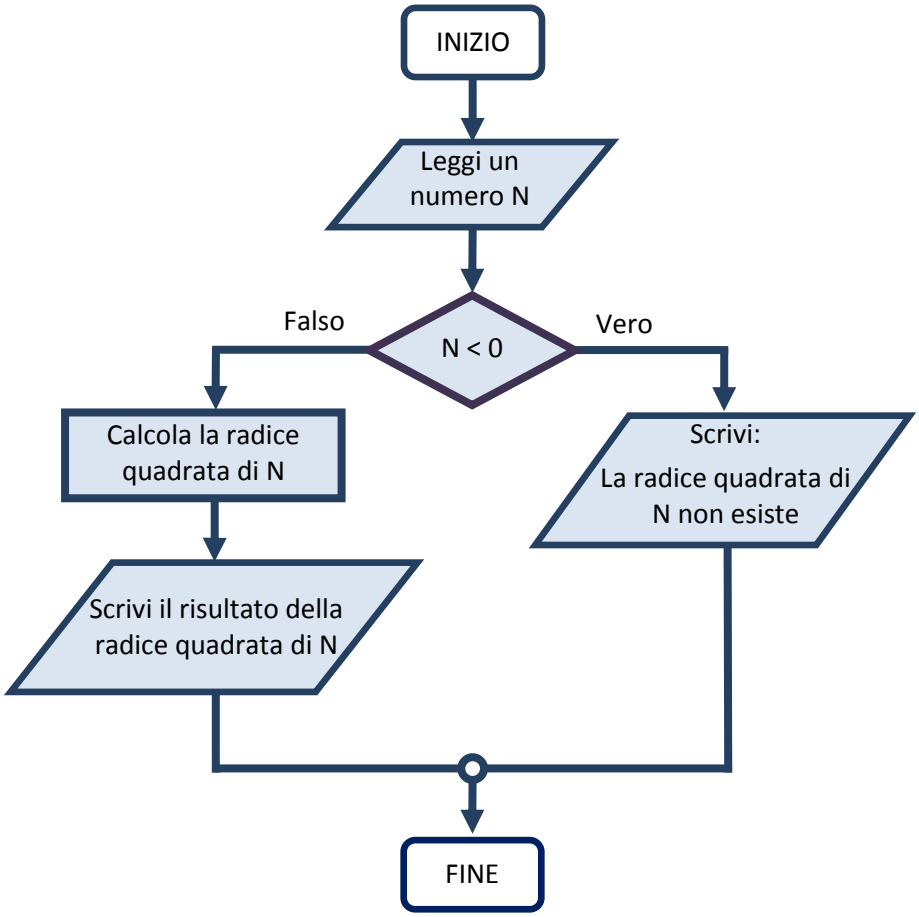
Ad esempio l’algoritmo risolutivo dell’ultimo problema, ricordando che il lupo non mangia il cavolo, è il seguente:

1. Inizio
2. traghettare la pecora sulla sponda B, lasciando assieme il lupo ed il cavolo
3. ritornare con la barca vuota sulla sponda A, lasciando la pecora da sola sulla sponda B
4. traghettare il cavolo sulla sponda B, lasciando il lupo da solo sulla sponda A
5. riportare la pecora sulla sponda A, lasciando il cavolo da solo sulla sponda B
6. traghettare il lupo sulla sponda B, lasciando da sola la pecora sulla sponda A
7. ritornare con la barca vuota sulla sponda A
8. traghettare la pecora sulla sponda B
9. Fine

1.3 Rappresentare algoritmi mediante diagrammi

Un modo chiaro per descrivere un algoritmo è rappresentato dai diagrammi di flusso (flow chart o diagrammi a blocchi).

I diagrammi di flusso sono grafici creati utilizzando una successione di figure (ognuna delle quali identifica una particolare azione) che rappresentano un ragionamento logico di immediata comprensione.



I simboli utilizzati in un diagramma di flusso sono i seguenti:

Oggetto grafico	Denominazione	Significato
INIZIO	Punto di partenza	Rappresenta un'azione che avvia il processo
FINE	Punto di fine	Rappresenta un'azione che conclude il processo
Leggi	Leggi	Rappresenta un'azione di ingresso dati
Scrivi	Scrivi	Rappresenta un'azione di uscita dei risultati
$S = b \cdot h$	Elaborazione	Rappresenta il comando o calcolo da eseguire
Test, Vero, Falso	Test	Rappresenta la scelta fra due possibili percorsi
→	Linea di flusso	Indica la direzione del percorso del flusso
○	Connessione	Rappresenta il punto d'inserimento nel grafico (generalmente contiene una lettera o un numero d'ordine)

1.4 Scrivere un semplice programma con l'uso del pseudolinguaggio

Un altro modo per rappresentare gli algoritmi consiste nell'uso di uno pseudolinguaggio. Questo tipo di linguaggio descrive le istruzioni con frasi rigorose anziché con i simboli grafici: si utilizzano parole chiave, operatori e nomi di variabili.

Tuttavia, questo tipo di linguaggio non è direttamente comprensibile dai programmi compilatori e interpreti e dovrà quindi essere tradotto in linguaggio di alto livello.

Non esiste un unico pseudo linguaggio. Analisti e programmatori di una azienda utilizzano uno pseudo linguaggio diverso da quello usato dai colleghi di un'altra.

Esempio

Pseudolinguaggio che traduce l'algoritmo per il calcolo dell'area di un rettangolo

```
PROGRAMMA Area del Rettangolo
    INIZIO
        LEGGI (base)
        LEGGI (altezza)
        AREA = base · altezza
        SCRIVI (area)
    FINE
```

2.0 PROGRAMMAZIONE STRUTTURATA

2.1 La programmazione strutturata

La *programmazione strutturata* è un paradigma (metodologia) di programmazione emerso nella seconda metà degli anni '60, che ha introdotto i concetti fondamentali che sono alla base di tutti gli altri paradigmi successivi, compresi gli ultimi linguaggi orientati agli oggetti.

Un paradigma di programmazione è uno stile fondamentale di programmazione, ovvero un insieme di strumenti concettuali forniti da un linguaggio di programmazione per la stesura dei programmi.

Esso è nato dalla necessità di portare ordine e criteri di lavoro più efficienti nella produzione dei programmi. Occorreva trovare una metodologia alternativa alla programmazione basata sul *salto incondizionato* (o goto) dei primi linguaggi di programmazione, che rendeva il codice scritto praticamente incomprensibile dallo stesso autore, tanto da essere chiamato *spaghetti code*, per la sua natura ingarbugliata.

La programmazione strutturata è una tecnica di programmazione che limita l'utilizzo delle sole tre strutture fondamentali:

- Struttura di sequenza
- Struttura di selezione *(o alternativa)*
- Struttura di iterazione *(o ciclo, ripetizione).*

Con questa tipologia di programmazione, la descrizione degli algoritmi viene effettuata in modo chiaro, facilmente leggibile e comprensibile: tale codice può essere agevolmente compreso e modificato in un secondo tempo anche da un eventuale programmatore che non sia il suo creatore.

I linguaggi di programmazione strutturati iniziarono a emergere nei primi anni 70, facendo tesoro delle idee dei due matematici italiani Corrado Bohm e Giuseppe Jacopini che per primi sottolinearono, in un teorema, come le tre strutture fondamentali offrivano un insieme di strutture di controllo completo, che garantivano la possibilità di descrivere tutti gli algoritmi.

Teorema di Jacopini-Bohm (1966)

Un qualsiasi algoritmo può essere espresso utilizzando esclusivamente le tre strutture di controllo: sequenza, selezione e iterazione.

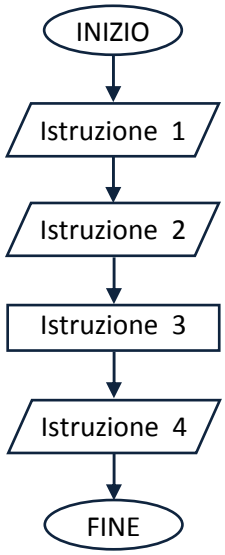
Fra i linguaggi tipici del paradigma strutturato si possono citare:

- il linguaggio *Pascal*, elaborato nel 1968 dal professor Wirth del Politecnico di Zurigo, che è tutt'oggi il linguaggio più diffuso e utilizzato per affrontare lo studio della programmazione
- il linguaggio *C*, messo a punto da Dennis Ritchie per implementare i primi sistemi operativi negli anni '70, che è il linguaggio di riferimento sia per i programmatori "più esperti" sia per i moderni linguaggi di programmazione dell'ambiente Web (*C++, Java*, *PHP* ecc. usano la stessa sintassi propria del linguaggio C).
- il Cobol
- l'algol

2.2 La struttura di sequenza

La struttura di sequenza è l'elenco ordinato delle istruzioni da eseguire.

In un ogni algoritmo, in quanto tale, non manca mai la struttura di sequenza.

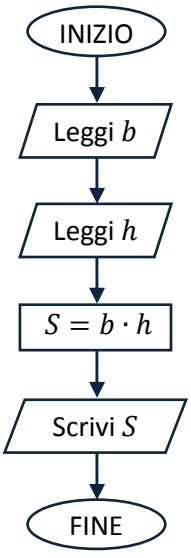
Struttura di sequenza	
Flow-chart	**Pseudolinguaggio**
	INIZIO Istruzione 1 Istruzione 2 Istruzione 3 Istruzione 4 FINE

Esempio – *Struttura di sequenza*

Dati la misura della base ***b*** e dell'altezza ***h*** di un rettangolo, calcola l'area del rettangolo ***S***.

Area del rettangolo

Flow-chart



Pseudolinguaggio

n°	Istruzione
1	INIZIO
2	Leggi la base *b*
3	Leggi l'altezza *h*
4	Assegna a *S* il valore *b·h*
5	Scrivi *S*
6	FINE

Trace table (Input: b=3 ; h=2)

n°	b	h	S
1	INIZIO		
2	3		
3		4	
4			3 · 2 = 6
5			**6**
6	FINE		

Linguaggio Java

```
import system.IO;
class rettangolo
{
      public static void main(String[] args)
            {
            int base, altezza, area;

            System.out.print("Base = ");
            base=IO.in.readInt();
            System.out.print ("Altezza = ");
            altezza = IO.in.readInt();

            area = base * altezza;

            System.out.println("Area = "+area);
            }
}
```

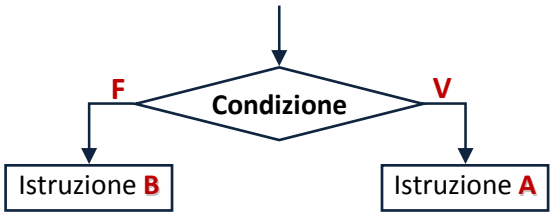
2.3 La struttura di selezione

In un algoritmo si può verificare che le istruzioni da eseguire siano diverse a seconda dei dati elaborati.

L'esecutore deve effettuare una scelta fra due alternative. Per effettuare tale scelta l'esecutore controlla il grado di verità della *condizione* posta in un test.

Se la condizione risulta *Vera* l'esecutore esegue l'istruzione A.

Se la condizione risulta *Falsa* l'esecutore segue l'istruzione B.

Struttura di selezione	
Flow-chart	**Pseudolinguaggio**
	*Se* Condizione *allora* Istruzione A *altrimenti* Istruzione B

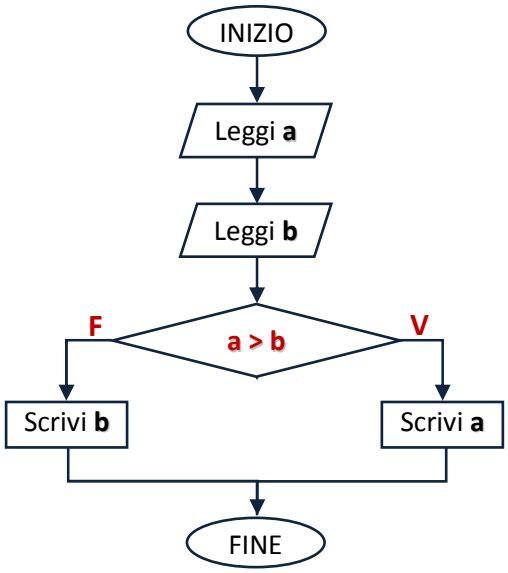
Codifica in linguaggio Java	
Selezione a due uscite	*Selezione a una uscita*
if (Condizione = Vera) Istruzione A; *else* Istruzione B;	*if* (Condizione = Vera) Istruzione A;

Esempio – *Struttura di selezione*

Dati due numeri ***a*** e ***b***, calcola il maggiore.

Max fra due numeri

Flow-chart



Pseudolinguaggio

n°	Istruzione
1	INIZIO
2	Leggi il numero *a*
3	Leggi il numero *b*
4	*Se* a > b
5	*allora* Scrivi *a*
6	*altrimenti* Scrivi *b*
7	FINE

Trace table (Input: a=3 ; b=5)

n°	a > b	a	b
1	INIZIO		
2		3	
3			5
4	Falso		
5			
6			**5**
7	FINE		

Linguaggio Java

```
import system.IO;
class max
{
      public static void main (String args[])
            {
            int a, b;

      System.out.print("Introduci il primo numero: ");
      a = IO.in.readInt();
      System.out.print("Introduci il secondo numero: ");
      b = IO.in.readInt();

      if (a > b)
            System.out.println("Il MAX e' " +a);
      else
            System.out.println("Il MAX e' " +b);
      }
}
```

2.4 La struttura di iterazione

In un algoritmo può capitare che alcune istruzioni debbano essere ripetute, in modo identico, più volte.

La ripetizione di un insieme di istruzioni prende il nome di *iterazione* o *ciclo* (loop).

Il gruppo di istruzioni ripetute prende il nome di *corpo del ciclo*.

L'istruzione di iterazione può essere definita o indefinita.

L'iterazione è detta *definita* quando è noto a priori il numero di ripetizioni.

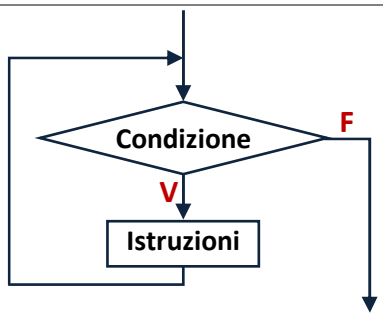
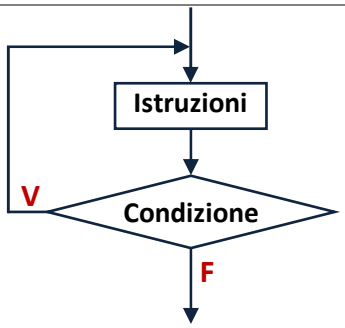
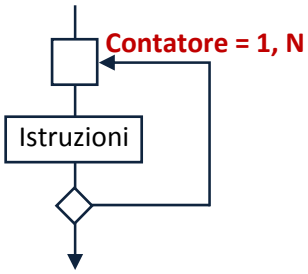
L'iterazione è detta *indefinita* quando il ciclo viene ripetuto un numero di volte sconosciuto a priori e termina quando si verifica una determinata condizione.

Esempi

- Ripeti 10 volte la preghiera "Atto di dolore" *(Iterazione definita)*
- Bevi un bicchiere di vino finché non cadi a terra ubriaco *(Iterazione indefinita)*

L'iterazione indefinita è detta *precondizionale* (o iterazione per vero) se il controllo per l'arresto dell'iterazione è posto prima del gruppo di istruzioni da ripetere.

L'iterazione indefinita è detta *postcondizionale* (o iterazione per falso) se il controllo per l'arresto dell'iterazione è posto dopo del gruppo di istruzioni da ripetere.

Strutture di Iterazione		
Iterazione indefinita precondizionale	*Iterazione indefinita postcondizionale*	*Iterazione definita enumerativa*
		
Mentre Condizione = Vera esegui Istruzioni	Ripeti Istruzioni finchè *Condizione = Falsa*	Ripeti Istruzioni N volte

Codifica in linguaggio Java		
Iterazione indefinita precondizionale	*Iterazione indefinita postcondizionale*	*Iterazione definita enumerativa*

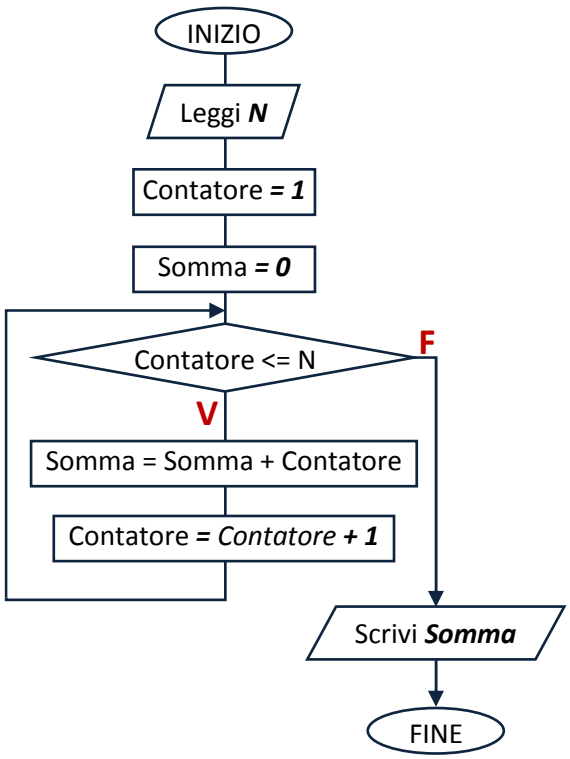
while (Condizione = Vera)
{
Istruzioni
}
	do
	{
	Istruzioni
	}
	while (Condizione = Falsa)
		for (contatore=1; contatore < N°Iterazioni; contatore++)
		{
		Istruzioni
		}

Esempio 1 - *Iterazione indefinita precondizionale*

Calcola la ***Somma*** dei primi ***N*** numeri naturali, senza utilizzare la formula di Gauss $S_n = \frac{n(n+1)}{2}$.

Somma dei primi N numeri naturali

Flow-chart



Pseudolinguaggio

n°	Istruzione
1	INIZIO
2	Leggi il numero *N*
3	Assegna al *Contatore* il valore *1*
4	Assegna alla *Somma* il valore *0*
5	*Mentre* il *Contatore <= N fai*
6	*Inizio*
7	Aggiungi alla *somma* il *contatore*
8	Incrementa di *1* il *contatore*
9	*Fine*
10	Scrivi *Somma*
11	FINE

Trace table (Input N=5)

n°	Cont < N	Contatore	Somma
3		1	
4			0
5	Vero		
7			0+1=1
8		2	
5	Vero		
7			1+2=3
8		3	
5	Vero		
7			3+3=6
8		4	
5	Vero		
7			6+4=10
8		5	
5	Vero		
7			10+5=15
8		6	
5	Falso		
10			**15**
11	FINE		

Linguaggio Java

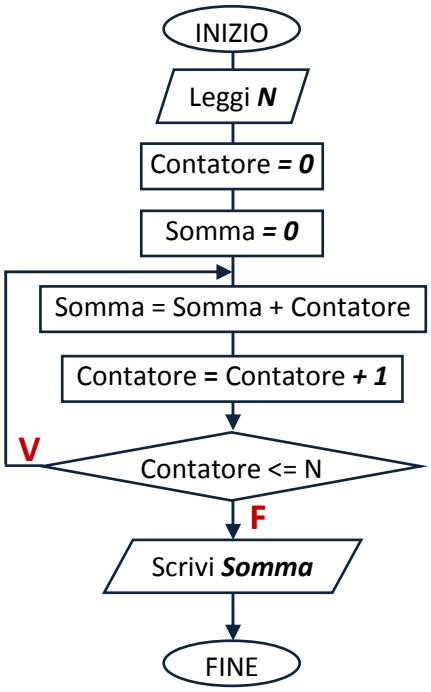
```
import system.IO;
class sommaNumeri_While
{
      public static void main (String [] args)
            {
            int contatore, somma, n;

            IO.out.println("            SOMMA DEI PRIMI N NUMERI NATURALI");

            IO.out.print("Introduci un numero naturale: ");
            n = IO.in.readInt();

            somma = 0;
            contatore = 1;

            while (contatore <= n)
                  {
                  somma = somma + contatore;
                  contatore++;
                  }

            IO.out.println("La somma dei primi "+n+" numeri naturali e' "+somma);
            }
}
```

Esempio 2 - *Iterazione indefinita postcondizionale*

Calcola la ***Somma*** dei primi ***N*** numeri naturali, senza utilizzare la formula di Gauss $S_n = \frac{n(n+1)}{2}$.

Somma dei primi N numeri naturali

Flow-chart



Pseudolinguaggio

n°	Istruzione
1	INIZIO
2	Leggi il numero *N*
3	Assegna al *Contatore* il valore *0*
4	Assegna alla *Somma* il valore *0*
5	*Ripeti*
6	*Inizio*
7	Aggiungi alla *somma* il *contatore*
8	Incrementa di *1* il *contatore*
9	*Fine*
10	*Finchè Contatore <= N*
11	Scrivi *Somma*
12	FINE

Trace table (Input N=5)

n°	Cont < N	Contatore	Somma
1		INIZIO	
2			
3		0	
4			0
7			0+0=0
8		1	
10	Vero		
7			1+2=3
8		2	
10	Vero		
7			3+3=6
8		3	
10	Vero		
7			6+4=10
8		4	
10	Vero		
7			10+5=15
8		5	
10	Falso		
11			**15**
12		FINE	

Linguaggio Java

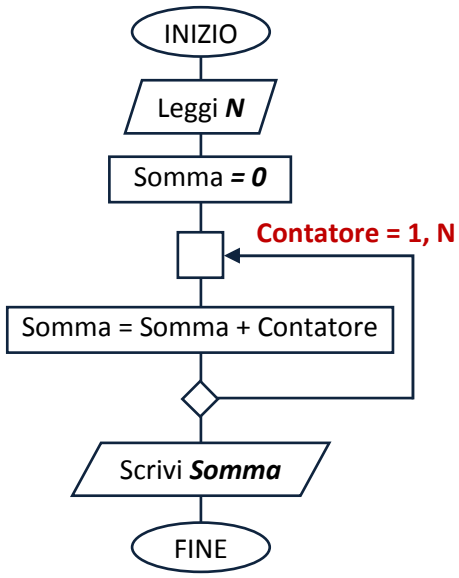
```
import system.IO;
class sommaNumeri_Do
{
      public static void main (String [] args)
            {
            int contatore, somma, n;

            IO.out.println("            SOMMA DEI PRIMI N NUMERI NATURALI");

            IO.out.print("Introduci un numero naturale: ");
            n = IO.in.readInt();

            contatore=0;
            somma = 0;

            do
                  {
                  somma = somma + contatore;
                  contatore++;
                  }
            while (contatore <= n);

            IO.out.println("La somma dei primi "+n+" numeri naturali e' "+somma);
      }
}
```

Esempio 3 - *Iterazione definita enumerativa*

Calcola la ***Somma*** dei primi ***N*** numeri naturali, senza utilizzare la formula di Gauss $S_n = \frac{n(n+1)}{2}$.

Somma dei primi N numeri naturali

Flow-chart



Pseudolinguaggio

n°	Istruzione
1	INIZIO
2	Leggi il numero *N*
3	Assegna alla *Somma* il valore *0*
4	Ripeti con *Contatore da 1 a N*
5	*Inizio*
6	Aggiungi alla *somma* il *contatore*
7	*Fine*
8	
9	Scrivi *Somma*
10	FINE

Trace table (N=5)

n°		Contatore	Somma
3			0
4		1	
6			1+1=1
4		2	
6			1+2=3
4		3	
6			3+3=6
4		4	
6			6+4=10
4		5	
6			10+5=15
9			**15**
10		FINE	

Linguaggio Java

```
import system.IO;
class sommaNumeri_For
{
      public static void main (String [] args)
            {
            int contatore, somma, n;

            IO.out.println("            SOMMA DEI PRIMI N NUMERI NATURALI");

            IO.out.print("Introduci un numero naturale: ");
            n = IO.in.readInt();

            somma = 0;

            for (contatore=1; contatore <= n; contatore++)
                  {
                  somma = somma + contatore;
                  }

            IO.out.println("La somma dei primi "+n+" numeri naturali e' "+somma);
      }
}
```

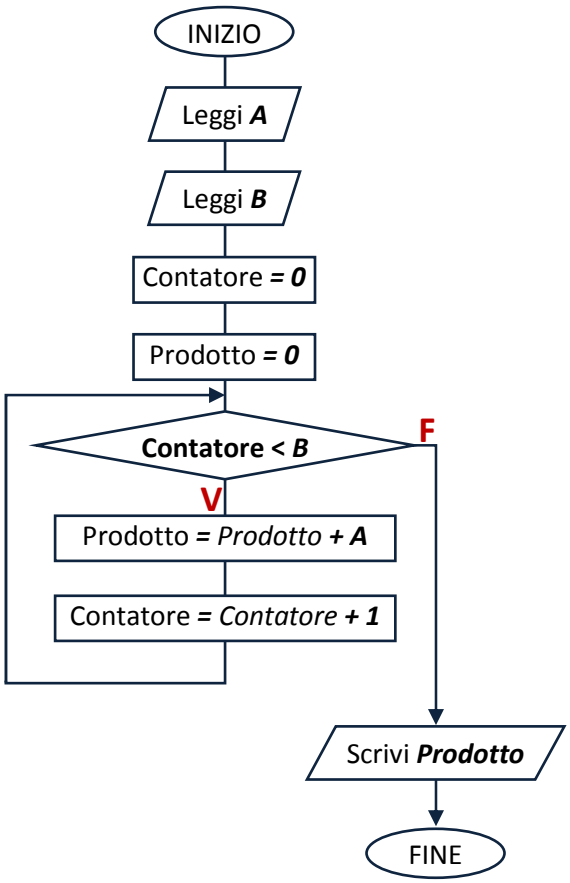
2.5 Algoritmi vari

Esempio 1 – *Prodotto di due numeri*

Dati due numeri ***A*** e ***B***, calcola il ***Prodotto*** dei due numeri utilizzando solo l'operazione di addizione.

Prodotto di due numeri

Flow-chart



Pseudolinguaggio		Trace table (Input: A=6 ; B=4)			
n°	Istruzione	n°	Cont < B	Contatore	Prodotto
1	INIZIO	4		0	
2	Leggi il numero *A*	5			0
3	Leggi il numero *B*	6	Vero		
4	Assegna al *Contatore* il valore *0*	8		1	
5	Assegna al *Prodotto* il valore *0*	9			0+6=6
6	*Mentre* il *Contatore < B fai*	6	Vero		
7	*Inizio*	8		2	
8	Incrementa di *1* il *contatore*	9			6+6=12
9	Incrementa di *A* il *prodotto*	6	Vero		
10	*Fine*	8		3	
11	Scrivi *Prodotto*	9			12+6=18
12	FINE	6	Vero		
		8		4	
		9			18+6=24
		6	Falso		
		11			**24**
		12	FINE		

Linguaggio Java

```
class prodotto
{
      public static void main (String args[])
            {
            int A, B, contatore, prodotto;

            System.out.print("Introduci il primo fattore: ");
            A = IO.in.readInt();
            System.out.print("Introduci il secondo fattore: ");
            B = IO.in.readInt();
            contatore = 0;
            prodotto = 0;

            while (contatore < B)
                        {
                        prodotto = prodotto + A;
                        contatore++;
                        }

            System.out.println("");
            System.out.println("Il prodotto e' "+prodotto);
            }
}
```

Esempio 2 - *M.C.D. fra due numeri (Algoritmo di Euclide)*

Per determinare il M.C.D. fra due numeri naturali m e n si può 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 che di n si ha che $m = kd$ e $n = hd$

La differenza $m - n$ è allora: $m - n = kd - hd$

Raccogliendo il fattore d si ottiene $m - n = d(k - h)$.

Si deduce che anche $m - n$ ha d come divisore.

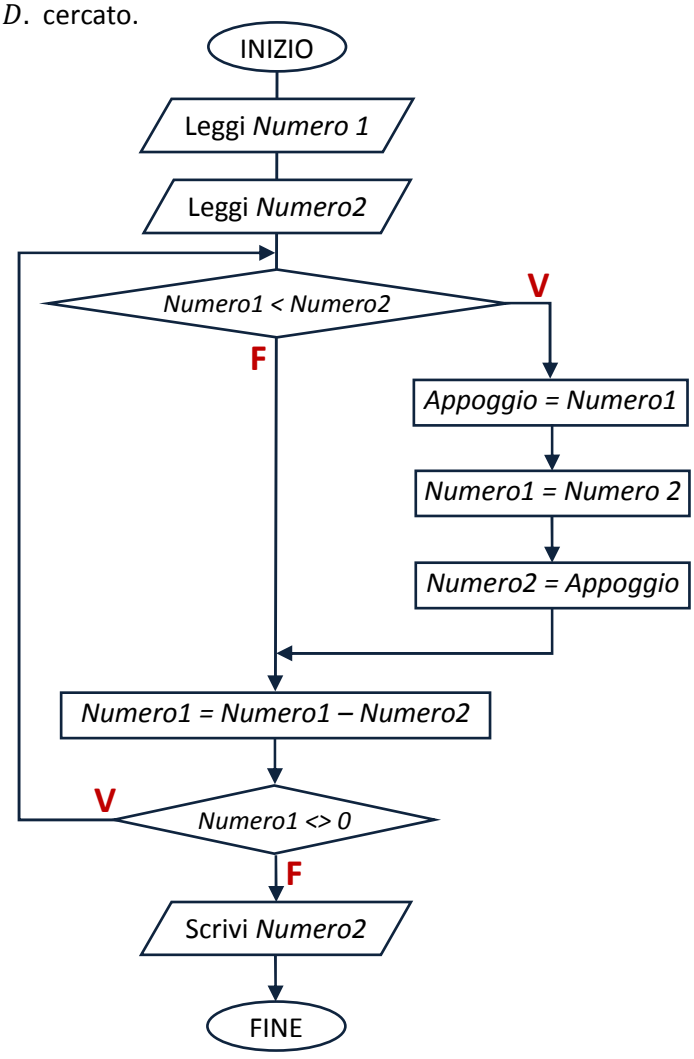
Pertanto i divisori comuni a m e n sono comuni anche a $m - n$ e n.

Cioè: $M.C.D.(m,n) = M.C.D.(m-n,n)$.

Si può allora determinare il M.C.D. fra due numeri per sottrazioni successive.

L'algoritmo è il seguente:

- si confrontano i due numeri, se il primo è minore del secondo si scambiano
- si esegue la sottrazione fra i due numeri
- si confronta poi il secondo numero con la differenza, se è necessario si scambiano
- si esegue la sottrazione fra i due numeri
- Si prosegue in questo modo fino ad ottenere una sequenza di numeri *(sempre più piccoli)* che hanno il medesimo M.C.D.
- Così facendo si giunge a 0, e a questo punto, essendo $M.C.D.(0,n) = n$, si conclude che il numero precedente è il $M.C.D.$ cercato.



M.C.D. di Euclide (Sottrazioni successive)								
Pseudolinguaggio		**Trace table** (Input: Dividendo =6 ; Divisore=15)						
n°	**Istruzione**	**n°**	**Test Se**	**Test Finchè**	**Numero1**	**Numero2**	**Appoggio**	**M.C.D.**
1	INIZIO	1						
2	Leggi *Numero1*	2			6			
3	Leggi *Numero2*	3				15		
4	*Ripeti*	5	Vero					
5	*Se* Numero1 < Numero2 *Allora*	7					6	
6	*Inizio*	8			15			
7	Appoggio = Numero1	9				6		
8	Numero1 = Numero2	11			15 – 6 = 9			
9	Numero2 = Appoggio	12		Falso				
10	*Fine*	5	Falso					
11	Numero1 = Numero1 - Numero2	11			9 – 6 = 3			
12	*Finchè Numero1 = 0*	12		Falso				
13	*M.C.D. = Numero2*	5	Vero					
14	Scrivi *M.C.D.*	7					3	
15	FINE	8			6			
		9				3		
		11			6 – 3 = 3			
		12		Falso				
		5	Falso					
		11			3 – 3 = 0			
		12		Vero				
		13						3
		14						**3**
		15	Fine					

Linguaggio Java

```
import system.IO;
class mcd
{
      public static void main (String args[])
            {
            int numero1, numero2, temporaneo;         int mcd = 1;

            System.out.print("Introduci il primo numero: ");
            numero1 = IO.in.readInt();
            System.out.print("Introduci il secondo numero: ");
            numero2 = IO.in.readInt();

            do
                  {
                  if (numero1 < numero2)
                        {
                        temporaneo = numero1;         // scambia i numeri
                        numero1    = numero2;
                        numero2    = temporaneo;
                        }
                  numero1 = numero1 - numero2;
                  }
            while (numero1 != 0);

            mcd = numero2;

            System.out.println("");
            System.out.println("Il M.C.D. = "+mcd);
            }
}
```

Esempio 3 – *Divisione con resto*

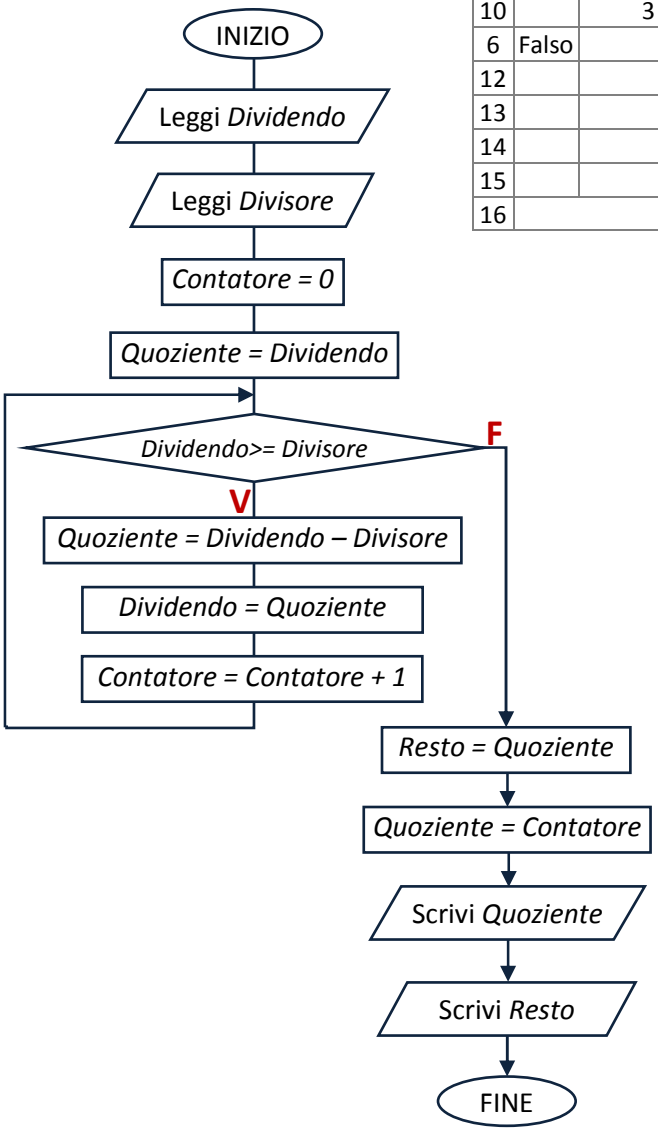
Divisione con resto fra due numeri

Pseudolinguaggio

n°	Istruzione
1	INIZIO
2	Leggi il *Dividendo*
3	Leggi il *Divisore*
4	Assegna al *Contatore* il valore *0*
5	Assegna al *Quoziente* il valore *Dividendo*
6	*Mentre* il *Dividendo >= Divisore fai*
7	*Inizio*
8	*Quoziente = Dividendo - Divisore*
9	*Dividendo = Quoziente*
10	Incrementa di *1* il *Contatore*
11	*Fine*
12	*Resto = Quoziente*
13	*Quoziente = Contatore*
14	Scrivi *Quoziente*
15	Scrivi *Resto*
16	FINE

Trace table (Input: Dividendo =19 ; Divisore=5)

n°	Test	Contatore	Dividendo	Divisore	Quoziente	Resto
1						
2			19			
3				5		
4		0				
5					19	
6	Vero					
8					19 – 5 = 14	
9			14			
10		1				
6	Vero					
8					14 – 5 = 9	
9			9			
10		2				
6	Vero					
8					9 – 5 = 4	
9			4			
10		3				
6	Falso					
12						4
13					3	
14					**3**	
15						**4**
16	Fine					



Linguaggio Java

```
import system.IO;
class divisioneConResto
{
      public static void main (String [] args)
            {
            int dividendo, divisore, quoziente, resto, contatore;

            IO.out.print("Introduci il dividendo: ");
            dividendo = IO.in.readInt();
            IO.out.print("Introduci il divisore: ");
            divisore = IO.in.readInt();

            contatore = 0;
            quoziente = dividendo;

            while (dividendo >= divisore)
                  {
                  quoziente = dividendo - divisore;
                  dividendo = quoziente;
                  contatore++;
                  }

            resto = quoziente;
            quoziente = contatore;

            IO.out.println("Quoziente = " +quoziente);
            IO.out.println("Resto = " +resto);
            }
}
```

Esempio 4 – *Scambia*

Scambia il contenuto della variabile A con il contenuto della variabile B

Linguaggio Java

```
class scambia
{
      public static void main (String args[])
            {
            int A, B, temporaneo;

            System.out.println("                  SCAMBIA DUE NUMERI");
            System.out.println("");

            System.out.print("Introduci A: ");
            A = IO.in.readInt();
            System.out.println("");

            System.out.print("Introduci B: ");
            B = IO.in.readInt();

            temporaneo = A;                    // scambia i numeri
            A = B;
            B = temporaneo;

            System.out.println("");
            System.out.println("A = "+A);
            System.out.println("B = "+B);
      }
}
```

Esempio 5 – *Valore assoluto*

Calcola il valore assoluto di un numero *N* .

Linguaggio Java

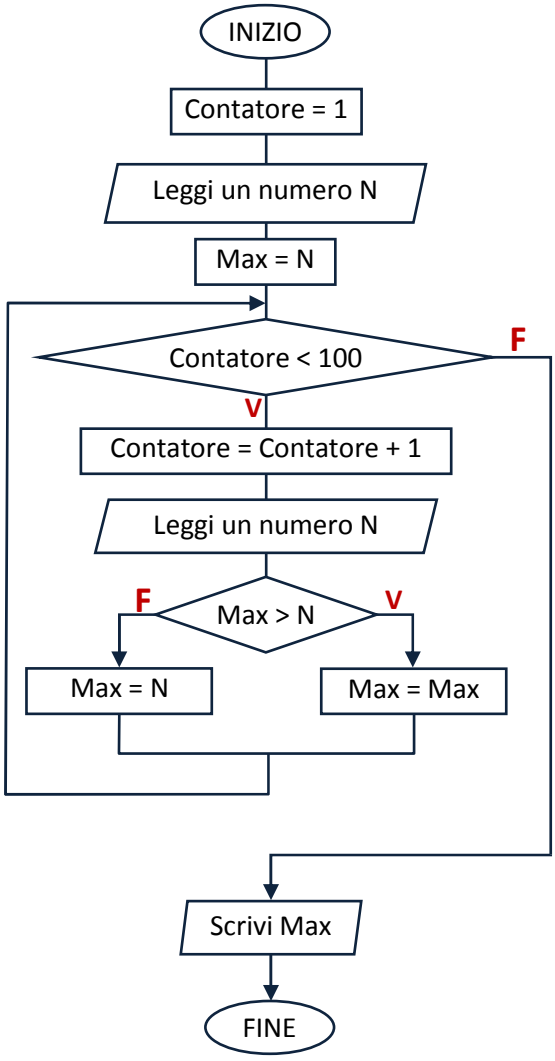
```
import system.IO;

class valoreAssoluto
{
      public static void main (String args[])
            {
            int A;

            System.out.print("Introduci A: ");
            A = IO.in.readInt();

            if (A < 0) A = -A;

            System.out.println("|A| = "+A);
      }
}
```

Esempio 6 – *Max fra n numeri*

Calcola il max fra *N* numeri .

Linguaggio Java

```
import system.IO;
class max10n
{
      public static void main (String args[])
            {
            int A, B, max, contatore;

            System.out.print("Introduci un numero: ");
            A = IO.in.readInt();
            max = A;

            for (contatore=1; contatore<=10; contatore++)
                  {
                  System.out.print("Introduci un numero: ");
                  B = IO.in.readInt();
                  if (B > max)    max = B;
                  }

            System.out.println("");
            System.out.println("Il MAX e' "+max);
            }
}
```



Altri esercizi

Esercizio 1

Calcola la media fra *N* numeri .

Esercizio 2

Calcola i primi 7 multipli del numero 2 .

Esercizio 3

Dati in ingresso due numeri A e B, scambia il valore del numero A con il valore del numero B.

Esercizio 4

Parchimetro

Esercizio 5

Controllo sui dati di input (numero positivo, 1° numero > del 2°, …)