



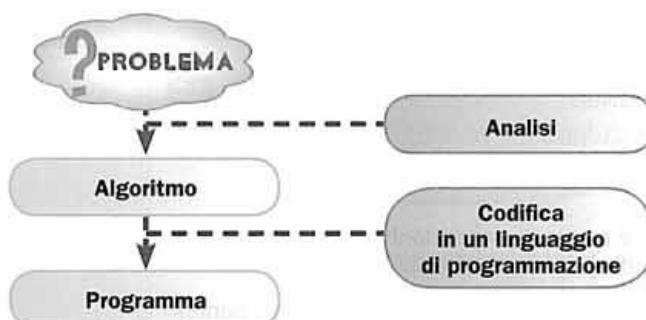
## LEZIONE



### La programmazione

Per risolvere un problema abbiamo imparato a individuare la strategia risolutiva e a descriverla in una sequenza di passi elementari, costruendo un algoritmo. L'algoritmo, però, non è direttamente eseguibile dall'esecutore. Per renderlo eseguibile, è necessario rappresentarlo in un sistema formale che consenta di comunicarlo al computer. Questo significa trasformare l'algoritmo concettuale in un insieme di istruzioni ben definite, che vanno scritte in modo chiaro e non ambiguo. Nel caso dei computer, il sistema formale che consente di **codificare**, cioè di tradurre nel linguaggio adottato l'algoritmo risolutivo. L'attività di codifica viene anche indicata con il termine **implementazione**, molto usato in informatica con il preciso significato di concretizzazione di un'astrazione.

L'algoritmo così tradotto prende il nome di **programma**. La disciplina che si occupa dell'automatizzazione dei processi risolutivi dei problemi prende il nome di **programmazione**.



Il paradigma di programmazione è strettamente legato al linguaggio: non si fa programmazione logica in C o in Java, non si programma a oggetti in Prolog. L'algoritmo risolutivo viene scelto in base al paradigma e al linguaggio adottato.

### La documentazione del lavoro

Per uno studio corretto della programmazione è importantissimo abituarsi a documentare ogni fase del lavoro con grafici, tabelle e commenti. Mentre si programma, si ha spesso la necessità di tornare indietro per rivedere alcuni punti, per correggere errori o per sviluppare meglio un problema già risolto. Anche a distanza di tempo, un lavoro ben documentato sarà sicuramente più semplice da rileggere e da correggere. Realizzare una **buona documentazione**, quindi, è una regola fondamentale per uno sviluppo completo ed esauriente della soluzione di un problema. Per questo motivo suggeriamo alcuni accorgimenti.

- Realizzare la documentazione contemporaneamente alla progettazione e mai successivamente.
- Riferire tutti i chiarimenti indicati nella documentazione ai **punti fondamentali del problema**, indicando dettagliatamente che cosa fare e, subordinatamente, come farlo.
- Attribuire alle variabili nomi significativi, che favoriscano la comprensione dello scopo per cui le variabili vengono utilizzate.

L'identificatore	Esempio
deve suggerire il contenuto	Massimo, Minimo, Media
può essere composto da più parole	ValoreMassimo, MediaGeometrica
può essere composto da un solo carattere	X, Y, Z
non dovrebbe avere un nome scelto a caso	Abc, Xk, Aa
non deve contenere spazi	Valore Massimo, Media Geometrica
non deve contenere simboli speciali	%IVA, Giorno/Mese
non dovrebbe contenere caratteri accentati	Età, Città, Paternità



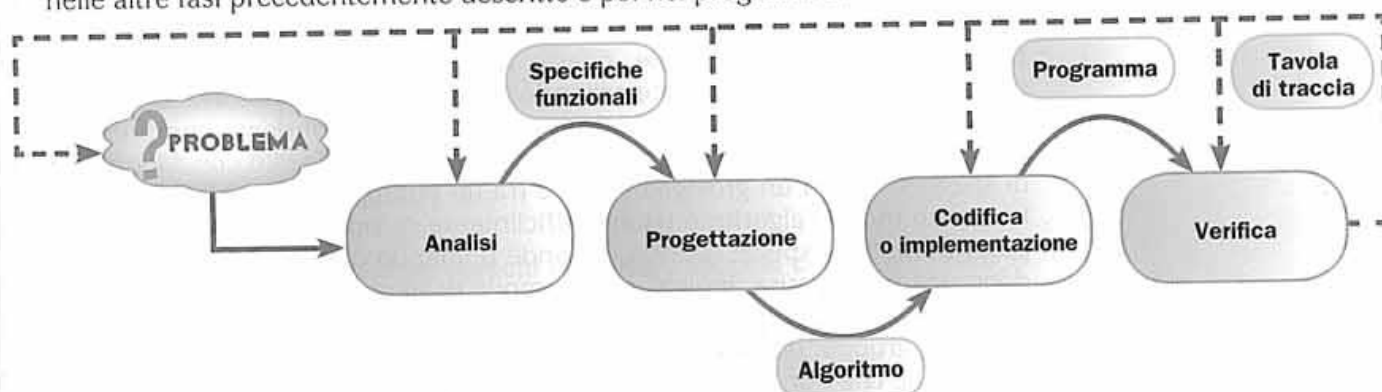
- Commentare in modo semplice le istruzioni, evitando annotazioni superflue. Durante la lettura di un algoritmo, i **commenti** alle istruzioni rappresentano la prima e più immediata documentazione disponibile. Essi aiutano a comprendere e a giustificare la presenza di alcune istruzioni e a favorire la verifica della correttezza formale dell'algoritmo. Non tutte le istruzioni hanno bisogno di essere commentate: in generale saranno trascurate tutte le istruzioni il cui contenuto è facilmente deducibile dal contesto.
- Aggiornare la documentazione immediatamente appena dovessero essere necessarie modifiche all'algoritmo o al programma.

Per ottenere un lavoro ben documentato, quindi, è consigliabile procedere affrontando i seguenti passaggi.

- **Lettura del testo del problema.** È indispensabile leggere attentamente e interpretare correttamente la traccia del problema per poter passare alle fasi successive.
- **Analisi del problema.** È una delle fasi più delicate. Consiste nell'esporre l'interpretazione che segue la lettura del testo del problema e nel mettere in evidenza tutti gli aspetti fondamentali. Durante questa fase occorre individuare con grande attenzione di quali dati di input **input** è necessario servirsi e quali dati di **output** occorre produrre;
- **Analisi dei dati.** Consiste nell'analizzare i dati individuati durante l'analisi del problema e nel descriverli dettagliatamente. Utilizzeremo la **tabella delle variabili** con il nome, la funzione (I = input, O = output, L = lavoro), il tipo e la descrizione (il significato e l'utilizzo) di ogni variabile coinvolta nell'elaborazione. Le variabili che hanno funzione di input e di output costituiranno l'area di interesse del problema, ossia le **specifiche funzionali**. È buona norma compilare la tabella indicando dapprima le variabili che descrivono le specifiche funzionali e, di seguito, le altre. Quando i dati di output sono rappresentati da messaggi e non da valori di variabili, compileremo la tabella scrivendo "Messaggio" all'interno della colonna del nome e completeremo solo la colonna *Funzione* (scrivendo al suo interno la lettera O) e la colonna *Descrizione* (riportando il contenuto del messaggio).
- **Formalizzazione dell'algoritmo.** È la fase di dettaglio. Consiste nello sviluppo dell'algoritmo usando lo pseudolinguaggio, i DaB o qualsiasi altro metodo di descrizione formale conosciuto.
- **Test (o trace).** Consiste nella rappresentazione dell'esecuzione dell'algoritmo, ossia nel compimento del test per verificare se i passi dettagliati risolvono il problema dato. Per far ciò ci serviremo di un'altra tabella, la **tavola di traccia**. Inserendo dati opportunamente scelti, seguiremo, passo dopo passo, i valori assunti dalle variabili, così controlleremo l'esattezza delle istruzioni e il raggiungimento dell'obiettivo.
- **Codifica.** L'algoritmo così testato potrà, a questo punto, essere codificato nel linguaggio di programmazione scelto. Tuttavia, se durante questa fase dovessero risultare necessarie delle modifiche, queste andranno riportate prima nelle altre fasi precedentemente descritte e poi nel programma.

NOME	FUNZIONE	TIPO	DESCRIZIONE

NOME	FUNZIONE	TIPO	DESCRIZIONE
Istruzione 1			
Istruzione 2			
.....			
Istruzione N			





# La programmazione strutturata e il costrutto sequenza

## La programmazione strutturata

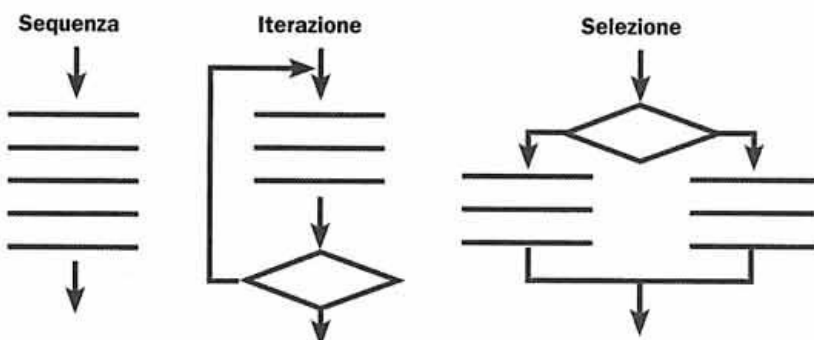
Analizziamo un metodo di strutturazione degli algoritmi che ha l'obiettivo di rendere più facile la loro costruzione, la loro lettura e la loro eventuale modifica. Tale metodo di programmazione si basa sull'utilizzo di tre costrutti sintattici fondamentali, noti come **strutture di controllo**:

- sequenza;
- selezione;
- iterazione.

La metodologia di programmazione che considera un algoritmo come un insieme di blocchi di istruzioni, ognuno fornito di un solo ingresso e una sola uscita e organizzati tra di loro secondo le strutture di controllo di sequenza, selezione e iterazione, prende il nome di **programmazione strutturata**.

Ciascun blocco è isolato dagli altri, nel senso che non è possibile, per esempio, saltare dall'interno di uno all'interno di un altro (cioè non è prevista un'istruzione del tipo **VAI A...**). A tale proposito, due matematici italiani, Corrado Bohm e Giuseppe Jacopini, nel 1966 enunciarono il seguente teorema:

un algoritmo scritto secondo le regole della programmazione a salti, per quanto complesso, può essere sempre trasformato in un algoritmo a esso equivalente che utilizzi esclusivamente tre costrutti sintattici fondamentali: **sequenza, selezione e iterazione**.



Dal teorema seguono due considerazioni:

- gli algoritmi che analizzeremo non conterranno al loro interno la famigerata istruzione di salto (**VAI A...**);
- utilizzando opportunamente i tre costrutti sintattici fondamentali possiamo scrivere qualsiasi algoritmo.

## Perché nasce la programmazione strutturata?

Quando l'arte della programmazione muoveva i primi passi, i programmatori consideravano dimostrazione di bravura il programmare utilizzando il minor numero di istruzioni possibile, questo faceva sì che si tendesse a riutilizzare pezzi di codice semplicemente saltando al loro inizio. Ne risultavano complicati collegamenti tra le varie parti del programma, non facilmente comprensibili a chi non lo aveva scritto. L'aspetto grafico dell'algoritmo era il cosiddetto "piatto di spaghetti", con un groviglio di linee tra un punto e l'altro. Il problema è che in questo modo l'algoritmo risulta difficilmente comprensibile anche allo stesso programmatore, specie quando è grande o quando questi lo riprende in esame dopo una lunga pausa. Inoltre, questo modo di lavorare può essere ancora praticabile da un singolo programmatore, ma diventa del tutto inconcepibile per un lavoro di gruppo, quando è necessario che il codice sia facilmente capito anche da persone che non lo hanno scritto.



Si è dunque arrivati, verso la fine degli anni Sessanta, a cambiare completamente la visione di che cosa fosse un programma ben scritto, e a non considerare più la capacità di districarsi in mezzo ad aggrovigliate matasse di VAI A come principale caratteristica del buon programmatore.

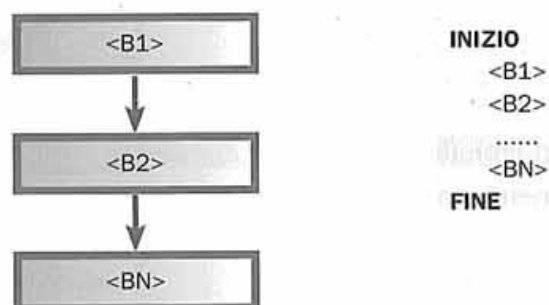
Alla fine è emerso proprio lo schema della programmazione strutturata che, come abbiamo visto, si proponeva di ottenere l'aspetto di un flusso ordinato tra un inizio e una fine, per programmi che di per sé sarebbero intricati. Il modello è quello di una struttura sequenziale, nella quale si applicano le varie operazioni una di seguito all'altra, in modo ordinato, senza alternative possibili. La semplice sequenza non può esprimere tutta la potenza degli algoritmi, poiché questa è essenzialmente contenuta nella capacità di scegliere tra due alternative. Come si possono dunque conciliare queste due esigenze?

L'idea è semplicissima: basta imitare il linguaggio naturale. Il VAI A è concepibile solo per una macchina i cui "pensieri" hanno dei ben determinati indirizzi, mentre in un linguaggio naturale un algoritmo è già espresso in una forma strutturata che corrisponde allo "svolgimento temporale" di una particolare computazione, il cosiddetto **processo**. In un linguaggio naturale i costrutti utilizzati sono "fai questo ...", "poi fai quello ...", "se ... fai questo ... altrimenti fai quello ...", "finché ... fai così ..." oppure "ripeti 5 volte questo ...", che sono proprio i costrutti di **controllo del flusso, sequenza, alternativa e ciclo**, al centro di questa unità formativa.

Forma non strutturata	Forma strutturata
1) leggi x, y	leggi x, y
2) se $x < y$ scambiali	se $x < y$ scambiali
3) dividi x per y, chiama r il resto	<b>ripeti</b>
4) poni $x = y$	dividi x per y, chiama r il resto
5) poni $y = r$	poni $x = y$
6) se $y \neq 0$ vai a 3)	poni $y = r$
	<b>fino a che <math>y = 0</math></b>
7) stampa x	stampa x
8) fine	fine

## Il costrutto sequenza

La sequenza è il più semplice fra i tre costrutti fondamentali. Si utilizza quando le azioni devono essere eseguite ordinatamente una dopo l'altra senza alcuna possibilità di scelta. Le istruzioni vengono scritte una dopo l'altra separandole in modo opportuno (per esempio con un punto e virgola ";", o semplicemente scrivendo un'istruzione per riga); esse verranno poi eseguite una dopo l'altra nell'ordine con cui sono scritte. In generale tale struttura si rappresenta nel seguente modo:



dove B1, B2 e BN possono essere **blocchi semplici** o **blocchi composti**. Un blocco semplice è, per esempio, un'istruzione di assegnazione, un'istruzione di I/O o uno degli altri costrutti fondamentali. Il blocco composto, invece, è un insieme di più blocchi semplici. In base a quest'ultimo concetto, anche un intero algoritmo strutturato è una struttura sequenziale.



### Scambiare il contenuto di due variabili A e B.

#### Analisi del problema

A prima vista la soluzione immediata sembrerebbe:

- prendi il contenuto di B e mettilo in A;
- prendi il contenuto di A e mettilo in B.

Naturalmente questa soluzione è errata, in quanto l'esecuzione della prima azione comporta la perdita immediata del contenuto della variabile A. Per risolvere questo problema dobbiamo servirci di una terza variabile C in cui "salvare temporaneamente" il contenuto di A. È un po' come se volessimo scambiare il contenuto di due bottiglie contenenti una del vino e l'altra del latte: non possiamo versare il vino contenuto nella bottiglia all'interno dell'altra! Così facendo, otterremmo soltanto un "cocktail poco gradevole", ma non risolveremmo il problema: per farlo, è necessario utilizzare una terza bottiglia. In dettaglio, quindi, le azioni da compiere sono le seguenti:

- prendi il contenuto di A e mettilo in C;
- prendi il contenuto di B e mettilo in A;
- prendi il contenuto di C e mettilo in B.

Vediamole graficamente ricorrendo all'analogia con le scatole. Supponendo che l'ambiente di valutazione sia il seguente:

$\{(A, 2, \text{Intero}), (B, 6, \text{Intero}), (C, 0, \text{Intero})\}$

inizialmente avremo:



Prendi il contenuto di A e mettilo in C.



Nuovo ambiente di valutazione:

$\{(A, 2, \text{Intero}), (B, 6, \text{Intero}), (C, 2, \text{Intero})\}$

Prendi il contenuto di B e mettilo in A.



Nuovo ambiente di valutazione:

{(A, 6, Intero), (B, 6, Intero), (C, 2, Intero)}

Prendi il contenuto di C e mettilo in B.

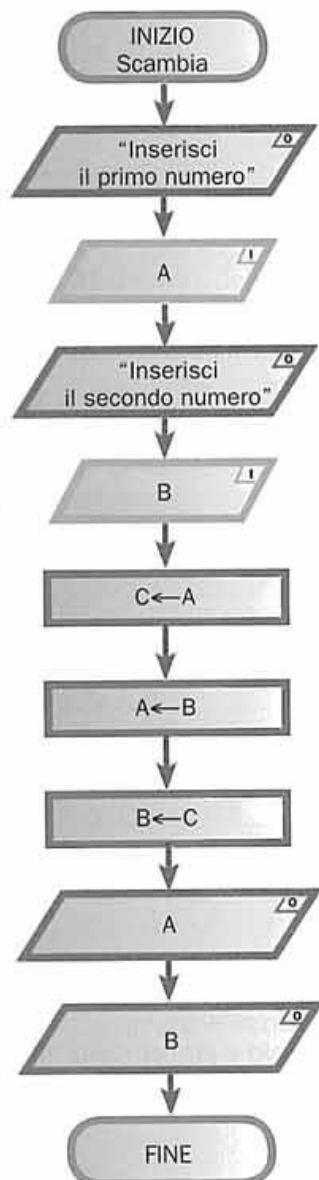


Ambiente di valutazione terminale:

{(A, 6, Intero), (B, 2, Intero), (C, 2, Intero)}

### Analisi dei dati

NOME	FUNZIONE	TIPO	DESCRIZIONE
A	I/O	Intero	Primo valore da scambiare. Dopo lo scambio diviene variabile di output.
B	I/O	Intero	Secondo valore da scambiare. Dopo lo scambio diviene variabile di output.
C	L	Intero	Variabile che consentirà di effettuare lo scambio.



### Formalizzazione dell'algoritmo

ALGORITMO Scambia

VARIABILI

A, B, C: INTERO

INIZIO

SCRIVI("Inserisci il primo numero")

LEGGI(A)

SCRIVI("Inserisci il secondo numero")

LEGGI(B)

C ← A

A ← B

B ← C

SCRIVI(A)

SCRIVI(B)

FINE

### Test

Istruzione	A	B	C	Output
LEGGI(A)	2			
LEGGI(B)		6		
C ← A			2	
A ← B	6			
B ← C		2		
SCRIVI(A)				6
SCRIVI(B)				2

Con un diagramma a blocchi si vogliono cogliere gli aspetti essenziali di un algoritmo; questo significa, per esempio, "trascurare" alcuni messaggi ovvi che nel programma scritto in linguaggio di programmazione saranno invece indicati. È questo il caso dei messaggi "Inserisci il primo numero" e "Inserisci il secondo numero" che precedono la lettura dei dati. Per questo motivo, in tutti gli algoritmi presenti in questo libro, i messaggi "ovvi" potranno anche non essere presenti.





1. Qual è il risultato prodotto dal seguente algoritmo? Enuncia il problema risolto.

```
▶ ALGORITMO Uno
▶ VARIABILI
▶   X, Y, Z : INTERO
▶ INIZIO
▶   SCRIVI("Inserisci due numeri")
▶   LEGGI(X, Y)
▶    $Z \leftarrow X + Y$ 
▶   SCRIVI(Z)
▶ FINE
```



2. Qual è il risultato prodotto dal seguente algoritmo? Enuncia il problema risolto.



```
▶ ALGORITMO Due
▶ VARIABILI
▶   A, B, P1, P2, Somma : INTERO
▶ INIZIO
▶   SCRIVI("Inserisci due numeri")
▶   LEGGI(A, B)
▶    $P1 \leftarrow 3 * A$ 
▶    $P2 \leftarrow 2 * B$ 
▶    $Somma \leftarrow P1 + P2$ 
▶   SCRIVI(Somma)
▶ FINE
```



3. Qual è il risultato prodotto dal seguente algoritmo? Enuncia il problema risolto.



```
▶ ALGORITMO Tre
▶ VARIABILI
▶   A, B, Sorpresa : INTERO
▶ INIZIO
▶   SCRIVI("Inserisci due numeri")
▶   LEGGI(A, B)
▶    $Sorpresa \leftarrow (A + B) / 2$ 
▶   SCRIVI(Sorpresa)
▶ FINE
```



4. Qual è il risultato prodotto dal seguente algoritmo? Enuncia il problema risolto. ☐☐☐

▶ ALGORITMO Quattro

▶ COSTANTI

▶ PIGRECO ← 3.14

▶ VARIABILI

▶ R, Circ, Area : INTERO

▶ INIZIO

▶ SCRIVI("Inserisci la misura del raggio")

▶ LEGGI(R)

▶ Circ ←  $2 * R * \text{PIGRECO}$

▶ Area ←  $R * R * \text{PIGRECO}$

▶ SCRIVI(Circ, Area)

▶ FINE



5. Qual è il risultato prodotto dal seguente algoritmo? Realizza il diagramma a ☐☐☐  
blocchi e compila la tabella delle variabili.

▶ ALGORITMO Cinque

▶ VARIABILI

▶ Somma, Interesse : REALE

▶ INIZIO

▶ SCRIVI("Inserisci la somma di denaro")

▶ LEGGI(Somma)

▶ Interesse ←  $\text{Somma} * 0.30$

▶ SCRIVI("Interesse calcolato = ", Interesse)

▶ FINE

All'interno dell'algoritmo esiste un valore che potrebbe essere dichiarato come costante? Se la risposta è affermativa, individua e correggi lo pseudocodice di questa nuova dichiarazione.

6. Il seguente algoritmo richiede in input la misura della base e dell'altezza di ☐☐☐  
un rettangolo, quindi calcola e visualizza il perimetro e l'area.

▶ ALGORITMO Sei

▶ VARIABILI

▶ Base, Altezza, Perimetro, Area : REALE

▶ INIZIO

▶ SCRIVI("Inserisci la misura della base ")

▶ LEGGI( )

▶ SCRIVI("Inserisci la misura dell'altezza ")

▶ LEGGI( )

▶  $= (\text{Base} + \text{ )} * 2$

▶  $\text{Area} = \text{ } * \text{Altezza}$

▶ SCRIVI( , Perimetro)

▶ SCRIVI("Area = ", Area)

▶ FINE

Completa le istruzioni riportate parzialmente e, successivamente, realizza il diagramma a blocchi e la tabella delle variabili.

# LEZIONE

## 17



## Il costrutto selezione

### Quando si utilizza il costrutto selezione

Questo costrutto permette di effettuare una scelta fra due possibili alternative. Per effettuare una scelta, però, dobbiamo valutare una condizione. Un esempio tratto dalla vita di tutti i giorni potrebbe essere il seguente:

SE pioverà ALLORA prenderò l'autobus ALTRIMENTI farò una passeggiata

In termini più generali possiamo scrivere:

SE SI VERIFICA LA CONDIZIONE  
ALLORA ESEGUI ISTRUZIONE1  
ALTRIMENTI ESEGUI ISTRUZIONE2

La sintassi di questo costrutto, chiamato più precisamente **selezione binaria**, è la seguente:

SE <(Condizione)>

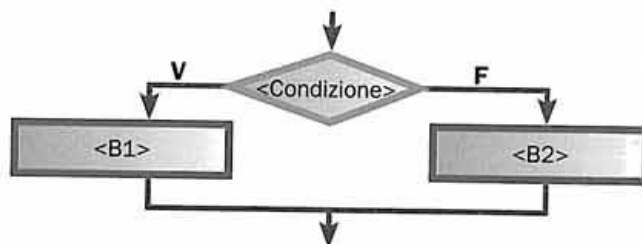
ALLORA

<B1>

ALTRIMENTI

<B2>

FINESE



dove B1 e B2, anche in questo caso, possono essere blocchi semplici o composti. Quindi, se la Condizione è Vera si esegue il blocco B1 altrimenti, cioè se la Condizione è Falsa, si esegue il blocco B2.

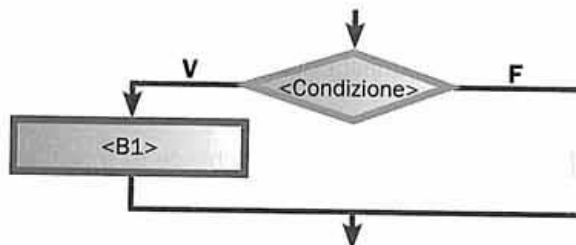
Il costrutto selezione può presentarsi anche con un solo ramo, cioè senza l'alternativa **ALTRIMENTI**. Questo caso, chiamato **selezione unaria**, si rappresenta sintatticamente nel seguente modo:

SE <(Condizione)>

ALLORA

<B1>

FINESE



### L'indentazione

Hai notato i "rientri" utilizzati per i SE, gli ALLORA, gli ALTRIMENTI? E hai osservato che ogni SE ha un suo FINESE corrispondente? E che la parola chiave ALTRIMENTI inizia esattamente nella stessa colonna in cui inizia la parola chiave ALLORA? Le istruzioni vengono così incolonnate per indicare la loro dipendenza. Si evidenzia, in tal modo, che un'istruzione è contenuta in un'altra. È buona abitudine curare questi incolonnamenti, poiché così facendo si garantisce una buona leggibilità dello pseudocodice. Questa tecnica è conosciuta come **indentazione**, è estremamente diffusa e il suo utilizzo viene considerato come una norma fondamentale di buona programmazione. La tecnica è basata sull'idea di usare gli spazi bianchi (ossia i rientri) allo scopo di separare più chiaramente le istruzioni e, in particolare, di rappresentare esplicitamente le relazioni di annidamento.





Consideriamo il seguente frammento di pseudocodice:

```

1.  SE(Età < MaggioreEtà)
2.      ALLORA
3.          SE(Età < EtàMinima)
4.              ALLORA
5.                  SCRIVI("Mi dispiace, non hai l'età per lavorare")
6.              ALTRIMENTI
7.                  SCRIVI("Puoi lavorare solo come apprendista")
8.          FINESE
9.      ALTRIMENTI
10.         SCRIVI("Tutto ok! Puoi lavorare")
11.     FINESE
    
```

L'indentazione del codice rende chiaro il fatto che la verifica di riga 3 viene eseguita *solo se* ha avuto esito positivo quella di riga 1, ovvero che la seconda selezione è *annidata* nella prima.

## Il costrutto selezione multipla

Questo costrutto deriva dal costrutto selezione, anzi ne rappresenta un'estensione. Capita molto spesso di dover fare delle scelte orientandosi fra più possibilità. Quotidianamente ci poniamo delle domande alle quali possiamo dare più di una risposta. Per esempio: quale maglione indosserò questa mattina? Quale film vedrò stasera? Quali libri porterò domani a scuola? E così via.

Per risolvere i problemi in cui si opera una scelta tra più di due alternative, in dipendenza del valore assunto da un certo parametro, è molto utile utilizzare il costrutto selezione multipla. Esso è presente in quasi tutti i linguaggi di programmazione, anche se il comportamento non è sempre lo stesso. La sintassi è la seguente:

**NEL CASO CHE <Selettore> SIA**

<Valore1>: <B1>

<Valore2>: <B2>

.

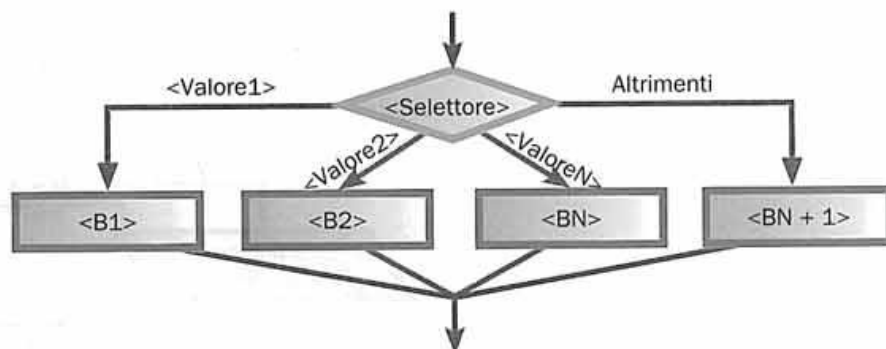
.

<ValoreN>: <BN>

**[ALTRIMENTI**

<BN+1>]

**FINECASO**



dove:

- <Selettore> può essere un valore numerico intero o un singolo carattere, non una costante; costituisce l'elemento che discrimina la scelta;
- <Valore1> deve essere dello stesso tipo di <Selettore> e può essere:
  - un valore unico (per esempio 1, 2, 3 ecc., oppure "a", "b", "c" ecc.);
  - una lista di valori (per esempio 1, 10, 100 – 2, 3, 6 – "a", "A", – "s", "S" e così via);
  - un intervallo chiuso di valori (per esempio 1...100 per comprendere tutti i numeri tra 1 e 100, "a"... "n" per indicare tutte le lettere tra "a" e "n" e così via).



# LEZIONE

# 17



**Dato in input un numero intero N, comunicare se è positivo o negativo.**

## Analisi del problema

Si parte dal concetto che un numero positivo è maggiore o uguale a zero. Quindi, non dobbiamo far altro che acquisire un numero dall'esterno e confrontarlo con zero: se il numero è maggiore o uguale a zero, allora è positivo, altrimenti è negativo.

## Analisi dei dati

NOME	FUNZIONE	TIPO	DESCRIZIONE
N	I	Intero	Valore da verificare
Messaggio	O		Numero positivo oppure numero negativo

## Formalizzazione dell'algoritmo

**ALGORITMO** PosNeg

**VARIABILI**

N: INTERO

**INIZIO**

LEGGI(N)

SE( $N \geq 0$ )

ALLORA

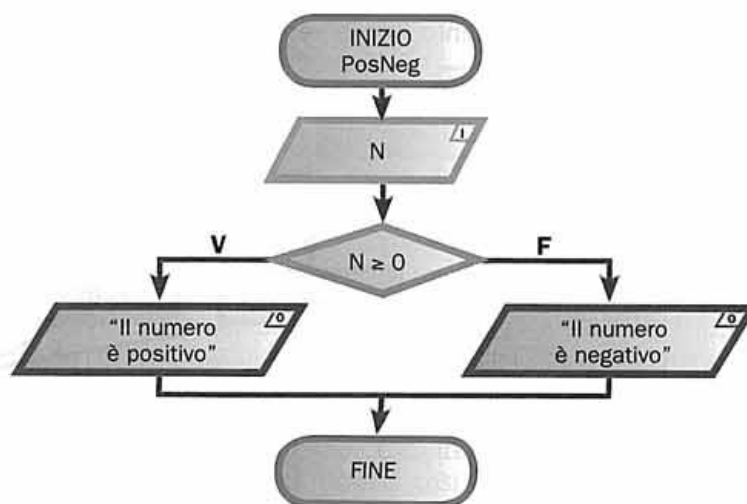
SCRIVI("Il numero è positivo")

ALTRIMENTI

SCRIVI("Il numero è negativo")

**FINESE**

**FINE**



## Test

È superfluo corredare l'esercizio della tavola di traccia: non abbiamo alcun bisogno di controllare la correttezza di questo algoritmo. È fin troppo chiaro!



Dato in input un numero intero N, comunicare se è negativo, positivo o nullo.

### Analisi del problema

L'analisi del problema è simile alla precedente, con un controllo in più. Se il numero richiesto in input è maggiore di zero, allora è positivo, ma se così non è, non possiamo affermare che il numero è negativo: potrebbe essere nullo. Controlliamo, quindi, se il numero è minore di zero. Se il numero non è maggiore o minore di zero, allora è nullo.

### Analisi dei dati

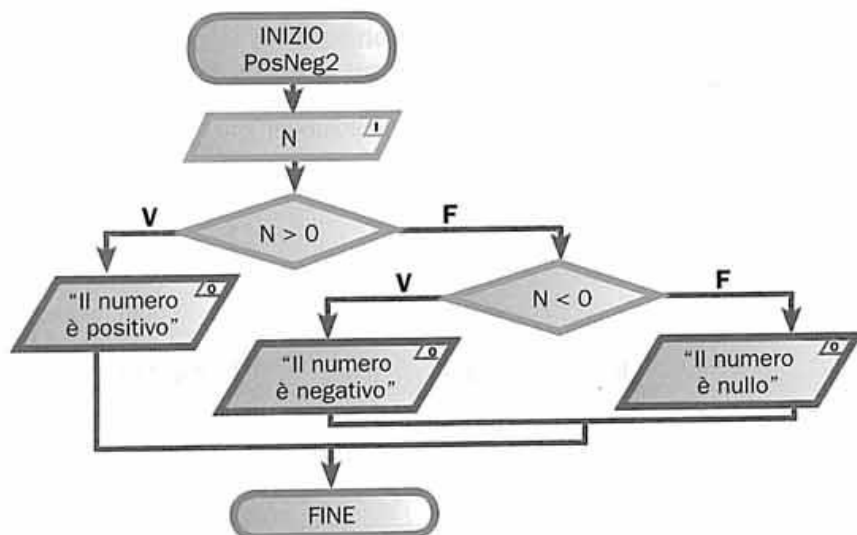
NOME	FUNZIONE	TIPO	DESCRIZIONE
N	I	Intero	Valore da verificare
Messaggio	O		Numero positivo, negativo o nullo

### Formalizzazione dell'algoritmo

```

ALGORITMO PosNeg2
VARIABILI
  N: INTERO
INIZIO
  LEGGI(N)
  SE(N > 0)
    ALLORA
      SCRIVI("Il numero è positivo")
    ALTRIMENTI
      SE(N < 0)
        ALLORA
          SCRIVI("Il numero è negativo")
        ALTRIMENTI
          SCRIVI("Il numero è nullo")
      FINESE
    FINESE
  FINE

```



### Test

Anche per questo problema è superfluo compilare la tavola di traccia. La semplicità e la correttezza dell'algoritmo sono fin troppo evidenti!

Questo esercizio è caratterizzato dalla presenza di un costrutto selezione all'interno di un altro. Quindi, nel ramo "altrimenti" del primo costrutto di selezione, se ne esegue un altro. Un costrutto inserito in un altro dello stesso tipo si dice **annidato**. Si parla, invece, di costrutti **in cascata** quando due o più costrutti di ugual tipo devono essere eseguiti in successione.





# LEZIONE



**Realizzare un algoritmo che risolva l'equazione lineare  $ax=b$  e comunichi all'utente se è un'equazione determinata, indeterminata o impossibile.**

## Analisi del problema

Un'equazione lineare si risolve separando i termini che contengono l'incognita dagli altri termini mediante l'applicazione delle proprietà dell'uguaglianza. In nostro caso la soluzione si ottiene dividendo il termine noto per quello associato all'incognita. Nel caso in cui si abbia a che fare con equazioni del tipo  $0 = 0$ , avremo equazioni indeterminate, mentre con equazioni del tipo  $0 = B$  oppure  $B \neq 0$  avremo equazioni impossibili.

## Analisi dei dati

NOME	FUNZIONE	TIPO	DESCRIZIONE
N	I	Intero	Valore del coefficiente A
B	I	Intero	Valore del coefficiente B
Messaggio	O		Equazione ammette soluzioni o non ammette soluzioni
Soluzione	L	Reale	Soluzione dell'equazione

## Formalizzazione dell'algoritmo



### ALGORITMO Equazione

#### VARIABILI

A, B: **INTERO**

Soluzione: **REALE**

#### INIZIO

**LEGGI**(A, B)

**SE** (A  $\neq$  0)

#### ALLORA

Soluzione  $\leftarrow B / A$

**SCRIVI**("La soluzione è ", Soluzione)

#### ALTRIMENTI

**SE**(B  $\neq$  0)

#### ALLORA

**SCRIVI**("L'equazione non ammette soluzioni")

#### ALTRIMENTI

**SCRIVI**("L'equazione ammette infinite soluzioni")

#### FINESE

#### FINESE

#### FINE

## Test

ISTRUZIONE	A	B	Soluzione	?	OUTPUT
<b>LEGGI</b> (A, B)	2	6			
A $\neq$ 0				V	
Soluzione $\leftarrow B/A$			3		
<b>SCRIVI</b> ("La soluzione è ", Soluzione)					3
<b>LEGGI</b> (A, B)	0	5			
A $\neq$ 0				F	
B $\neq$ 0				V	
<b>SCRIVI</b> ("L'equazione non ammette soluzioni")					L'equazione non ammette soluzioni
<b>LEGGI</b> (A, B)	0	0			
A $\neq$ 0				F	
B $\neq$ 0				F	
<b>SCRIVI</b> ("L'equazione ammette infinite soluzioni")					L'equazione ammette infinite soluzioni



**Realizzare un algoritmo che scriva il nome dei poligoni in base al numero dei lati (il numero dei lati non può essere superiore a 12).**

#### Analisi del problema

Il problema è semplice: occorre prima leggere il numero dei lati; quindi, in base al valore acquisito, se è minore o uguale a 12, scrivere a quale tipo di poligono corrisponde un certo numero di lati, altrimenti segnalare l'input non valido.

#### Analisi dei dati

NOME	FUNZIONE	TIPO	DESCRIZIONE
lati	1	Intero	Numero di lati del poligono
messaggio	0		Tipo di poligono o messaggio di errore

#### Formalizzazione dell'algoritmo

**ALGORITMO** Poligoni

**VARIABILI**

Lati: **INTERO**

**INIZIO**

**SCRIVI**("Inserisci il numero dei lati")

**LEGGI** (Lati)

**NEL CASO CHE** Lati **SIA**

    3: **SCRIVI**("Triangolo")

    4: **SCRIVI**("Quadrilatero")

    5: **SCRIVI**("Pentagono")

    6: **SCRIVI**("Esagono")

    7: **SCRIVI**("Ettagono")

    8: **SCRIVI**("Ottagono")

    9: **SCRIVI**("Ennagono")

    10: **SCRIVI**("Decagono")

    12: **SCRIVI**("Dodecagono")

**ALTRIMENTI**

**SCRIVI**("Inserimento errato")

**FINECASO**

**FINE**

**Lo stipendio base di un dipendente varia a seconda del livello di appartenenza; realizzare un algoritmo che, dato il nome e il livello di un dipendente, stampi il suo stipendio base.**

#### Formalizzazione dell'algoritmo

**ALGORITMO** Stipendi

**VARIABILI**

Nome: **STRINGA**[20]

Livello: **INTERO**

Stipendio: **REALE**

**INIZIO**

**SCRIVI**("Inserisci il nome dell'impiegato ")

**LEGGI**(Nome)

**SCRIVI**("Inserisci il livello (un numero da 1 a 4) ")

**LEGGI**(Livello)

**NEL CASO CHE** Livello **SIA**

    1: Stipendio ← 1500,75

    2: Stipendio ← 2000,50

    3: Stipendio ← 2500,82

    4: Stipendio ← 2950,45

**ALTRIMENTI**

**SCRIVI**("Livello digitato errato!")

**FINECASO**

**SE**(Livello >= 1)

**ALLORA**

**SE**(Livello <= 4)

**ALLORA**

**SCRIVI**("Il signor ", Nome, " percepisce uno stipendio base di euro ", Stipendio)

**FINESE**

**FINESE**

**FINE**



1. Considera il seguente frammento di pseudocodice:

```
▶ LEGGI(I)
▶ SE(J < 0)
▶   ALLORA
▶     J ← J + 1
▶   ALTRIMENTI
▶     J ← 0
▶ FINESE
▶ SCRIVI(J)
```

Che cosa viene stampato?

2. Modifichiamo il precedente frammento di pseudocodice nel seguente modo:

```
▶ LEGGI(J)
▶ SE(J > 0)
▶   ALLORA
▶     J ← J + 1
▶   ALTRIMENTI
▶     J ← 0
▶ FINESE
▶ SCRIVI(J)
```

Che cosa succede ora? Qual è il valore finale di J?

3. Modifichiamolo ancora:

```
▶ LEGGI(I)
▶ LEGGI(J)
▶ SE(I > 0)
▶   ALLORA
▶     SE(J > 0)
▶       ALLORA
▶         J ← 1
▶     FINESE
▶   ALTRIMENTI
▶     I ← 1
▶ FINESE
▶ SCRIVI(I)
▶ SCRIVI(J)
```

Che cosa succede ora? Qual è il valore finale di I e di J?



Il seguente pseudocodice risolve il problema di trovare il massimo tra due numeri interi. In esso, però, è presente un errore; sei in grado di trovarlo? ☐☐☐

ALGORITMO Massimo

VARIABILI

Num1, Num2, Max: INTERO

NIZIO

SCRIVI("Inserisci il primo numero")

LEGGI(Num1)

SCRIVI("Inserisci il secondo numero")

LEGGI(Num2)

SE(Num1 > Num2)

ALLORA

Max ← Num2

ALTRIMENTI

Max ← Num1

FINESE

SCRIVI("Il valore massimo è ", Max)

FINE

Il seguente pseudocodice risolve il seguente problema: dati in input due numeri interi e il simbolo dell'operazione, fornire in output il risultato. In esso, però, è presente un grave errore. Riesci a individuarlo? Inoltre, avresti potuto utilizzare un altro costrutto per la risoluzione del problema? ☐☐☐

ALGORITMO Operazione

VARIABILI

Num1, Num2, Ris: INTERO

Op: CARATTERE

NIZIO

SCRIVI("Inserisci il primo valore")

LEGGI(Num1)

SCRIVI("Inserisci il secondo valore")

LEGGI(Num1)

SCRIVI("Inserisci il simbolo della operazione")

LEGGI(Op)

SE(Op = '+')

ALLORA

Ris ← Num1 + Num2

FINESE

SE(Op = '-')

ALLORA

Ris ← Num1 - Num2

FINESE

SE(Op = '\*')

ALLORA

Ris ← Num1 \* Num2

FINESE

SE(Op = '/')

ALLORA

Ris ← Num1 / Num2

FINESE

SCRIVI("Il risultato è ", Ris)

FINE



# Algebra booleana e logica: introduzione

## Proposizioni ed enunciati

Nel pensiero greco antico, la **logica** era parte della filosofia ed era intesa come studio del modo di ragionare. **Aristotele**, il grande filosofo greco vissuto tra il 384 e il 322 a.C., se ne occupò organicamente, esaminando il discorso come traduzione verbale del pensiero. Successivamente **George Boole** (1815-1864), un matematico e logico inglese, pose le basi per la moderna logica matematica, fondamentale per il funzionamento del computer e per la sua programmazione. Boole intuì la mancanza di un formalismo matematico che permettesse di esprimere i concetti della logica simbolica e lavorò per colmare questa lacuna. Il risultato fu la costruzione di un sistema algebrico conosciuto con il nome di **algebra booleana**, che ha trovato poi applicazioni più vaste di quelle che lo stesso Boole poteva allora immaginare.

Le operazioni che un computer esegue, infatti, non sono unicamente di tipo aritmetico. In fase di programmazione capita spesso di incontrare istruzioni di tipo logico, per l'esecuzione delle quali occorre fare riferimento all'**algebra di Boole**. Questa è costituita da un insieme di operazioni definite su oggetti astratti, le **proposizioni** e gli **enunciati**, per cui è detta anche **algebra proposizionale**.

Una **proposizione** è un costrutto linguistico autonomo di senso compiuto, composto per lo meno da un soggetto e da un predicato.

Un **enunciato** è una particolare proposizione che può assumere solo due stati possibili (Vero/Falso, V/F). La verità o falsità di un enunciato rappresentano i valori di verità.



Oggi il sole splende – Londra è la capitale d'Italia – 4 è un numero pari

Le tre frasi sono enunciati:

- il primo ha valore Vero se oggi il sole effettivamente splende e il valore Falso in caso contrario;
- il secondo ha valore decisamente Falso;
- il terzo ha valore decisamente Vero.

È facile intuire che un enunciato può essere esclusivamente vero o falso, mai entrambi. Attenzione, le seguenti sono proposizioni ma non enunciati:

- "Prendimi il libro"
- "Il fiume Po scorre in Italia"
- "Che ore sono?"

Per essi non è infatti possibile stabilire i rispettivi valori di verità.

Gli enunciati per i quali si può immediatamente affermare se sono veri o falsi prendono il nome di **enunciati semplici** (o **atomici**). Una combinazione di enunciati legati da particolari operatori detti **connettivi logici** prende il nome di **enunciato composto** (o **molecolare**). I connettivi vengono così definiti proprio perché connettono i vari enunciati.



L'enunciato:

"Stasera studierò o andrò al cinema"

è un enunciato composto, poiché è formato dai sottoenunciati "Stasera studierò" e "andrò al cinema" legati dal connettivo "o"; il suo valore di verità sarà definito dalla valutazione dei sottoenunciati e dal connettivo che li congiunge.



## e funzioni logiche elementari

Analizzare gli enunciati composti significa associare a ognuno il valore Vero o Falso, partendo dall'analisi degli enunciati semplici che li compongono e applicando semplici regole di composizione. Accenniamo brevemente ai seguenti **connettivi logici**:

**AND** (congiunzione)    **OR** (disgiunzione)    **NOT** (negazione).

Costruiamo, per ognuno di essi, particolari tabelle dette **tavole di verità**. In esse  $p$  e  $q$  rappresentano due enunciati e sono illustrate tutte le possibili combinazioni di valori in funzione delle quali l'enunciato composto assume un certo valore. In alcuni linguaggi i valori Vero e Falso sono rappresentati rispettivamente con 1 e 0.

### Congiunzione AND (\*): prodotto logico

Questo connettivo viene anche detto **prodotto logico** poiché il risultato si può ottenere effettuando il prodotto algebrico delle due variabili che rappresentano gli enunciati in ingresso. Il valore di verità di  $p$  and  $q$  è dato dalla tabella qui a destra.

$p$	$q$	$p$ AND $q$
F	F	F
F	V	F
V	F	F
V	V	V



L'enunciato  $p$ :

"Io ascolto un CD"

e l'enunciato  $q$ :

"Io mangio un panino"

danno origine all'enunciato  $p$  and  $q$ :

"Io ascolto un CD e mangio un panino"

che è vero, effettivamente, se sto facendo entrambe le cose.

Come si vede, l'unico caso in cui  $p$  and  $q$  è vero è quando sia  $p$  che  $q$  sono veri.

### Disgiunzione OR (+): somma logica o disgiunzione inclusiva

Questo connettivo è anche detto **somma logica** poiché il risultato si può ottenere effettuando la somma algebrica delle due variabili che rappresentano gli enunciati in ingresso. Il valore di verità di  $p$  or  $q$  è dato dalla seguente tabella.

$p$	$q$	$p$ OR $q$
F	F	F
F	V	V
V	F	V
V	V	V



L'enunciato  $p$ :

"Io vado al cinema"

e l'enunciato  $q$ :

"Io ascolto un CD"

danno origine all'enunciato  $p$  OR  $q$ :

"Io vado al cinema o ascolto un CD"

che è vero se faccio almeno una delle due cose.

Basta che  $p$  sia vero oppure che  $q$  sia vero, perché  $p$  or  $q$  sia anch'esso vero.

### Negazione NOT (!): complemento

Questo connettivo è anche detto **negazione logica** poiché inverte, cioè "complementa" il valore della variabile in ingresso. Fornisce in uscita il valore "opposto" al valore della variabile in ingresso, che per tale motivo viene detto **valore complementato** o **valore negato**. Il valore di verità di  $\text{not } p$  è dato dalla tabella qui accanto.

$p$	$\text{NOT } p$
F	V
V	F



L'enunciato  $p$ :

"Io vado al cinema"

dà origine all'enunciato  $\text{NOT } p$  (scritto anche come  $\text{!}p$  in alcuni contesti):

"Io non vado al cinema"



# Algebra booleana e logica: altre funzioni e regole di precedenza

## LEZIONE

# 19

### Altre funzioni logiche

I connettivi AND, OR e NOT svolgono le funzioni logiche elementari. Ne esiste un ultimo che prende il nome di **disgiunzione esclusiva** e che nella teoria degli insiemi corrisponde alla differenza simmetrica.

#### Disgiunzione XOR: disgiunzione esclusiva

Questo connettivo *esclude* che la proposizione risultante possa essere vera se entrambe le proposizioni componenti sono vere. Un esempio di disgiunzione esclusiva è fornito dalla proposizione seguente:

**"O è giorno o è notte".**

p	q	p XOR q
F	F	F
F	V	V
V	F	V
V	V	F

Il corrispondente connettivo si indica con **XOR** e la sua tabella di verità è riportata qui a sinistra.

Oltre alle funzioni logiche AND, OR e NOT, esistono altre funzioni logiche elementari di uso comune che possono essere definite in funzione delle prime. Fra queste vi sono la funzione NAND (NOT AND) e NOR (NOT OR), che si ottengono, come suggerisce il nome stesso, come negazione logica, rispettivamente, delle funzioni AND e OR.

### Gli operatori relazionali

Nell'ambito della logica booleana è possibile utilizzare gli **operatori relazionali** ( $=$ ,  $\neq$ ,  $<$ ,  $>$ ,  $\leq$ ,  $\geq$ ). Vediamo alcune equivalenze:

Dire	Equivale a dire
$A \neq B$	$\text{NOT } (A = B)$
$A \geq B$	$(A > B) \text{ OR } (A = B)$
$A \leq B$	$(A < B) \text{ OR } (A = B)$
$A > B$	$(\text{NOT } (A = B)) \text{ AND } (\text{NOT } (A < B))$
$A < B$	$(\text{NOT } (A = B)) \text{ AND } (\text{NOT } (A > B))$



Un ultimo appunto. Spesso in matematica utilizziamo la notazione:

$$A \leq X \leq B$$

per affermare che X è compreso tra A e B. Come tradurlo con gli operatori booleani, visto che la logica dei computer non accetta tale notazione?

È presto detto:

$A \leq X \leq B$  equivale a:

$(X \geq A) \text{ AND } (X \leq B)$  oppure  $(A \leq X) \text{ AND } (X \leq B)$

La notazione:

$$2 \leq X \leq 8$$

equivale a:

$(X \geq 2) \text{ AND } (X \leq 8)$  oppure  $(2 \leq X) \text{ AND } (X \leq 8)$

Tale relazione sarà vera, ovviamente, per  $X = 2, 3, 4, 5, 6, 7, 8$ .



## Regole di precedenza

Si possono combinare più operatori logici per ottenere espressioni logiche più complesse. Come nell'algebra tradizionale, anche nell'algebra di Boole sono definite precise precedenze tra gli operatori. Pertanto, è necessario ricordare che:

- **NOT** ha precedenza più alta di **AND** e **OR**; per esempio:  
NOT a AND NOT b OR NOT c equivale a (NOT a) AND (NOT b) OR (NOT c).
- **AND** ha precedenza più alta di **OR**, per esempio:  
a AND b OR c equivale a (a AND b) OR c.

Supponiamo di voler calcolare il valore di verità del seguente enunciato composto:

$$(p \text{ AND } q) \text{ OR } ((\text{NOT } p) \text{ AND } (\text{NOT } q))$$



1. Si assegnano i valori di ingresso alle varie occorrenze di p e di q.

p	AND	q	OR	NOT p	AND	NOT q
V		V				
V		F				
F		V				
F		F				

2. Si negano i valori di p e q.

p	AND	q	OR	NOT p	AND	NOT q
V	V	V		F		F
V	F	F		F		V
F	F	V		V		F
F	F	F		V		V

3. Si calcolano i valori del primo AND e si cancellano le colonne dei valori usati.

p	AND	q	OR	NOT p	AND	NOT q
V	V	V		F		F
V	F	F		F		V
F	F	V		V		F
F	F	F		V		V

4. Si opera allo stesso modo con il secondo AND.

p	AND	q	OR	NOT p	AND	NOT q
V	V	V		F	F	F
V	F	F		F	F	V
F	F	V		V	F	F
F	F	F		V	V	V

5. Si calcola infine l'OR utilizzando come valori di ingresso le due colonne rimaste.

p	AND	q	OR	NOT p	AND	NOT q
V	V	V	V	F	F	F
V	F	F	F	F	F	V
F	F	V	F	V	F	F
F	F	F	V	V	V	V

Nella colonna OR, che è il "connettivo principale" troviamo la tavola di verità del nostro enunciato.



# Algebra booleana e logica: altre funzioni e regole di precedenza

**Realizzare un algoritmo che, date in input le misure di tre lunghezze, stabilisca se possono essere le misure dei lati di un triangolo.**

## Analisi del problema

Come è noto, in un triangolo la somma di due lati qualsiasi è maggiore del terzo. Perciò, affinché tre numeri  $a$ ,  $b$ , e  $c$  possano essere considerati misure dei lati di un triangolo, dovranno essere contemporaneamente soddisfatte le tre condizioni:  $a + b > c$ ,  $a + c > b$ ,  $b + c > a$ . Per risolvere il problema faremo uso di una variabile booleana (che chiameremo *TuttoOK*) che avrà la funzione di "interruttore": all'inizio dell'algoritmo si inizializza la variabile *TuttoOK* al valore VERO. Se le condizioni  $a + b > c$ ,  $a + c > b$ ,  $b + c > a$  sono soddisfatte, e quindi le misure inserite possono essere le misure dei lati di un triangolo, la variabile *TuttoOK* mantiene il suo valore VERO, ma se una delle condizioni non viene verificata, alla variabile *TuttoOK* viene assegnato il valore FALSO. Al termine della verifica delle tre condizioni, occorre controllare il valore della variabile *TuttoOK*: se il valore è VERO allora significa che le tre lunghezze possono essere le misure dei lati di un triangolo, altrimenti no.

## Analisi dei dati

NOME	FUNZIONE	TIPO	DESCRIZIONE
A	I	Intero	Valore della prima lunghezza
B	I	Intero	Valore della seconda lunghezza
C	I	Intero	Valore della terza lunghezza
Messaggio	O		Lunghezze possono essere lati del triangolo o non possono essere

## Formalizzazione dell'algoritmo

### ALGORITMO Triangolo

#### VARIABILI

A, B, C: **INTERO**

TuttoOK: **BOOLEANO**

#### INIZIO

TuttoOK ← **VERO**

**SCRIVI**("Inserisci le tre lunghezze")

**LEGGI**(A, B, C)

**SE**( $A + B \leq C$ )

**ALLORA**

TuttoOK ← **FALSO**

**FINESE**

**SE**( $A + C \leq B$ )

**ALLORA**

TuttoOK ← **FALSO**

**FINESE**

**SE**( $C + B \leq A$ )

**ALLORA**

TuttoOK ← **FALSO**

**FINESE**

**SE**(TuttoOK = **VERO**) // Si poteva anche scrivere **SE**(TuttoOK)

**ALLORA**

**SCRIVI**("Le tre lunghezze possono essere lati di un triangolo")

**ALTRIMENTI**

**SCRIVI**("Le tre lunghezze non possono essere lati di un triangolo")

**FINESE**

**FINE**





e un algoritmo che, letti tre valori da tastiera  $a$ ,  $b$  e  $c$   $a \leq c$ , stabilisca se possono essere le misure dei lati di un triangolo e, in caso affermativo, visualizzi il tipo (scaleno, equilatero) e lo stampi a video.

### Il problema

Per questo problema inseriremo tre valori che rispettino la condizione affinché sappiamo che tre segmenti rappresentano i lati di un triangolo se il più grande è minore della somma degli altri due. Per questo accertamento ci servirà ancora di una variabile booleana che fungerà da "interruttore". Accertato che le tre misure possono essere quelle dei lati di un triangolo, stabiliremo il tipo di triangolo che possono formare. In particolare:

se i tre lati sono tutti uguali si tratta di un triangolo equilatero;  
se due lati sono uguali si tratta di un triangolo isoscele;  
altrimenti si tratta di un triangolo scaleno.

### I dati

FUNZIONE	TIPO	DESCRIZIONE
I	Intero	Valore della prima lunghezza
I	Intero	Valore della seconda lunghezza
I	Intero	Valore della terza lunghezza
O		Tipo di triangolo oppure messaggio di errore

### Azione dell'algoritmo

**NO** TipoTriangolo

**DO** INTERO

**DO** BOOLEANO

angolo: STRINGA[10]

**DO** ← FALSO

I("Inserisci le tre lunghezze")

(A, B, C)

- B > C)

**LORA**

Trovato ← VERO

SE((A=B) AND (B=C))

ALLORA

TipoTriangolo ← "Equilatero"

ALTRIMENTI

SE((A=B) OR (B=C) OR (A=C))

ALLORA

TipoTriangolo ← "Isoscele"

ALTRIMENTI

TipoTriangolo ← "Scaleno"

FINESE

FINESE

E

avato) // Significa SE(Trovato = VERO)

**LORA**

SCRIVI("Le misure digitate formano un triangolo ", TipoTriangolo)

**ALTRIMENTI**

SCRIVI("Le tre lunghezze non sono lati di un triangolo")

E





# Il costrutto iterativo precondizionale

## LEZIONE

# 20

### Iterazione = potenza di calcolo

Pur avendo a disposizione strutture sequenziali e di selezione, rimangono molte le situazioni "intrattabili" con tali strumenti. In tutti gli esempi visti nelle lezioni precedenti, la sequenza di istruzioni eseguita dall'algoritmo è sempre stata la stessa a ogni esecuzione: il risultato finale dipendeva solo dal valore delle variabili e dai pochissimi dati inseriti dall'utente.

I problemi risolti sono stati del tipo: *dati in input tre numeri A, B e C, fornire in output la loro somma*. Non abbiamo fatto altro che acquisire dall'esterno i valori di A, B, C, sommarli e visualizzare il risultato.

E se avessimo dovuto fare la somma di 100 numeri? Avremmo utilizzato 100 variabili? Avremmo realizzato l'algoritmo seguente?

#### INIZIO

```
LEGGI(A)
LEGGI(B)
LEGGI(C)
LEGGI(D)
LEGGI(E)
.
```

Certamente no! E ancora: se al posto di 100 numeri avessimo voluto sommare N numeri, con N richiesto in input e che, quindi, può variare da esecuzione a esecuzione?

Prendiamo proprio in esame questo problema: *dati in input N numeri, fornire in output la loro somma*. L'algoritmo risolutivo di questo problema deve essere **flessibile**, ossia deve adattarsi a gestire situazioni diverse: occorre, infatti, rendere **indeterminata** la quantità di numeri da elaborare, consentendo all'utente di inserire la quantità desiderata. L'algoritmo, pertanto, deve conoscere **quanti** e **quali** numeri di cui effettuare la somma. Per fare questo si possono seguire due strade:

- specificare all'inizio la quantità di numeri di cui effettuare la somma e, successivamente, inserirli;
- inserire in sequenza i vari numeri e fissare un valore prestabilito (il cosiddetto **dato tappo**) che ha il compito di chiudere la sequenza. Per esempio, se si fissa il valore zero come dato tappo, l'inserimento dei numeri terminerà quando si digiterà il valore zero.

Entrambi i casi sono accomunati dal fatto che *un gruppo di azioni deve essere ripetuto per un certo numero di volte*: occorre, quindi, utilizzare il **costrutto iterazione**. Il **costrutto iterazione** o **costrutto iterativo** (detto anche **ciclo**) viene utilizzato quando un'istruzione (o un gruppo di istruzioni) deve essere eseguita finché non si verifica una determinata **condizione**.

Le istruzioni che determinano l'esecuzione dei cicli rivestono particolare importanza nella programmazione in quanto consentono l'esecuzione ripetitiva di gruppi di istruzioni per un determinato numero di volte o fino al raggiungimento di un determinato risultato. Sono queste istruzioni, in definitiva, che consentono principalmente di utilizzare in modo efficace la potenza di calcolo degli elaboratori elettronici.

### Totalizzatori e contatori

Nell'elaborazione ciclica sono spesso utilizzati i **totalizzatori** (o **accumulatori**) e i **contatori**. Per chiarire meglio il concetto di totalizzatore, pensiamo alle azioni eseguite dal cassiere di un supermercato quando si presenta un cliente con il carrello pieno di merce. Il cassiere effettua un'elaborazione ciclica sulla merce acquistata: ogni oggetto viene esaminato per acquisirne il prezzo. Lo scopo dell'elaborazione è quello di cumulare i prezzi dei prodotti acquistati per stabilire il totale che il cliente dovrà corrispondere.



al punto di vista informatico si tratta di utilizzare una variabile (nell'esempio potrebbe essere rappresentata dal totalizzatore di cassa) che viene aggiornata per ogni prezzo acquisito: ogni nuovo prezzo acquisito non deve sostituire il precedente, ma aggiungersi ai prezzi già acquisiti precedentemente. Tale variabile dovrà essere azzerata quando si passa a un nuovo cliente (ogni cliente dovrà pagare solamente il prezzo dei prodotti da lui acquistati), si aggiornerà per ogni prodotto esaminato (ogni nuovo prezzo acquisito verrà cumulato ai precedenti) e una volta terminato l'esame dei prodotti acquistati conterrà il valore totale da pagare.

Nella programmazione, la variabile che assume tale compito viene definita totalizzatore o accumulatore: cioè una variabile nella quale ogni nuovo valore non sostituisce ma si accumula a quelli già presenti in precedenza. Se la variabile si aggiorna sempre di una quantità costante (per esempio, viene sempre aggiunta un'unità), viene chiamata contatore. Nel nostro caso il cassiere, oltre a totalizzare la somma spesa dal cliente, deve anche contare quanti articoli ha acquistato: ogni volta che passa un prodotto sotto il lettore ottico, il contatore si incrementa di una unità.

Totalizzatori e contatori devono essere inizializzati prima del loro utilizzo: nel nostro caso, il totalizzatore e il contatore devono essere azzerati (cioè inizializzati a zero), prima di cominciare l'elaborazione, affinché contengano un valore che rispecchi esattamente tutto ciò che è stato acquistato dal cliente esaminato. L'aggiornamento, invece, viene effettuato all'interno di un ciclo: il totalizzatore verrà incrementato del prezzo del singolo prodotto acquistato, mentre il contatore verrà incrementato di una unità ogni volta che il prodotto viene registrato in cassa. Quando i valori da esaminare terminano, il totalizzatore contiene il totale da pagare e il contatore contiene il numero di prodotti acquistati.

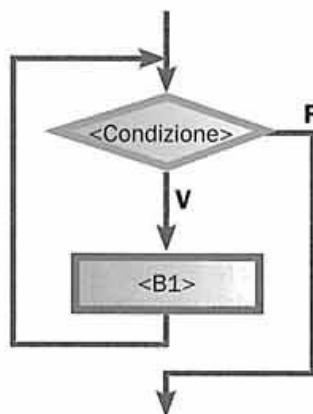
## Costrutto iterativo precondizionale

Nella programmazione strutturata vengono utilizzati due costrutti iterativi; cominciamo ad analizzare il primo, chiamato **costrutto iterativo precondizionale** (oppure **costrutto iterativo con controllo in testa**) poiché, come dice lo stesso nome, il controllo della condizione viene effettuato prima di eseguire le istruzioni che compongono il ciclo.

**MENTRE** (<Condizione>) **ESEGUI**

<B1>

**FINEMENTRE**



durante l'esecuzione di questo costrutto si valuta la <Condizione>. Se è vera si segue il blocco <B1> e si ritorna a valutare la <Condizione>. Questa esecuzione continua fino a quando la <Condizione> non risulta falsa e, in tal caso, si passa a eseguire l'istruzione successiva al FINEMENTRE.

L'espressione logica presente in <Condizione> viene detta **guardia del ciclo**, mentre il blocco di istruzioni <B1> viene detto **corpo del ciclo**.

L'uso del MENTRE richiede alcune precauzioni: nel progetto di un ciclo MENTRE necessario, infatti, considerare con attenzione le condizioni di ingresso e quelle di uscita. Se le condizioni di ingresso sono mal formulate, il computer, in fase di esecuzione, non entrerà mai nel ciclo. D'altra parte, se esiste un errore di logica all'interno del ciclo, si corre il rischio che l'esecuzione "entri in loop", ossia che si trovi intrappolati all'infinito all'interno del ciclo.



# Il costrutto iterativo precondizionale

**Realizzare un algoritmo che, dati in input N numeri interi, determini la loro somma e la media.**

## Analisi del problema

Con un problema così formulato, l'utilizzo del **costrutto sequenza** per la sua realizzazione è praticamente impossibile. Non conosciamo a priori quanti sono i numeri di cui fare la somma e, quindi, non sappiamo quante variabili utilizzare. Il costrutto iterazione, invece, non pone limiti e, come tale, è l'unico che ci permette la risoluzione di questo tipo di problemi. In questo esercizio però, prima di impostare l'iterazione, dobbiamo chiedere all'esterno quanti sono i numeri che bisogna sommare. È questa una richiesta essenziale per una valida esecuzione del costrutto: se non la facessimo, non sapremmo quando fermarci.

Alla fine del ciclo è necessario controllare che sia stato inserito almeno un numero. Se ciò non accade, le variabili *Somma* e *Conta* rimangono impostate al loro valore iniziale pari a zero e il calcolo della media avverrebbe effettuando una divisione tra due variabili che contengono zero. Per evitare questo inconveniente si effettua un controllo sulla variabile *Conta*.

## Analisi dei dati

NOME	FUNZIONE	TIPO	DESCRIZIONE
N	I	Intero	Quantità di numeri da inserire in input
Num	I	Intero	Numero richiesto in input
Somma	O	Intero	Somma dei numeri inseriti in input
Media	O	Reale	Media dei numeri inseriti in input
Conta	L	Intero	Contatore del ciclo

## Formalizzazione dell'algoritmo



### ALGORITMO SommaMedia1

#### VARIABILI

N, Num, Somma, Conta: **INTERO**

Media: **REALE**

#### INIZIO

Somma ← 0

Conta ← 0

**SCRIVI**("Quanti numeri vuoi inserire? ")

**LEGGI**(N)

**MENTRE**(Conta < N) **ESEGUI**

**SCRIVI**("Inserisci un numero ")

**LEGGI**(Num)

    Somma ← Somma + Num

    Conta ← Conta + 1

**FINEMENTRE**

**SE**(Conta = 0)

**ALLORA**

        Media ← 0

**ALTRIMENTI**

        Media ← Somma / Conta

**FINESE**

**SCRIVI**("La somma è ", Somma)

**SCRIVI**("La media è ", Media)

**FINE**

Nel blocco iterativo deve esserci l'istruzione che modifica la condizione (nel nostro caso  $Conta \leftarrow Conta + 1$ ), altrimenti, una volta iniziato, il ciclo verrà ripetuto all'infinito; in questa evenienza diremo di trovarci in un caso di **loop infinito**.



allizzare un algoritmo che, data in input una sequenza di numeri usa dallo zero, determini la loro somma e la media.

### Analisi del problema

Questo problema rappresenta una variante del precedente. In questa versione, non conoscendo a priori quanti sono i numeri di cui fare la somma, non dobbiamo preoccuparci di farlo conoscere all'inizio: si continuerà a inserire numeri finché non verrà inserito lo zero (**dato tappo**). Una volta digitato lo zero si uscirà dal ciclo e potremo visualizzare la somma e la media. Le differenze salienti con la precedente versione sono le seguenti.

Non è necessario utilizzare un contatore di ciclo poiché non è necessario contare i numeri inseriti per verificare la condizione presente nel ciclo (la **guardia del ciclo**). Nell'algoritmo, però, utilizzeremo ugualmente una variabile *Conta* che avrà sempre la funzione di contatore, ma non per consentirci di uscire dal ciclo, bensì per contare quanti numeri sono stati inseriti al fine di calcolare correttamente la media.

La lettura del numero viene inserita due volte: una **esterna al ciclo** che ci permette di entrare nel ciclo (nel caso in cui il numero inserito sia diverso dal tappo, ossia dallo zero) e una **interna** (cioè quella inserita nel corpo del ciclo) che ci permette di digitare numeri fino a quando non digitiamo lo zero.

### Analisi dei dati

VE	FUNZIONE	TIPO	DESCRIZIONE
n	I	Intero	Numero richiesto in input
suma	O	Intero	Somma dei numeri inseriti in input
media	O	Reale	Media dei numeri inseriti in input
conta	L	Intero	Contatore del ciclo

### Formalizzazione dell'algoritmo

**ALGORITMO** SommaMedia2

#### VARIABILI

Num, Somma, Conta: **INTERO**

Media: **REALE**

#### INIZIO

Somma ← 0

Conta ← 0

**SCRIVI**("Inserisci un numero")

**LEGGI**(Num)

**MENTRE**(Num ≠ 0) **ESEGUI**

Somma ← Somma + Num

Conta ← Conta + 1

**SCRIVI**("Inserisci un numero")

**LEGGI**(Num)

**FINEMENTRE**

**SE**(Conta = 0)

**ALLORA**

Media ← 0

**ALTRIMENTI**

Media ← Somma / Conta

**FINESE**

**SCRIVI**("La somma è", Somma)

**SCRIVI**("La media è", Media)

**FINE**





**Realizzare un algoritmo che, data in input una sequenza di numeri chiusa dallo zero, determini la loro somma e la media. Nel caso in cui l'utente non digiti lo zero, non sarà possibile inserire più di 50 numeri.**

### Analisi del problema

Questo problema rappresenta un'altra variante dei precedenti. Anche in questa versione continueremo a inserire numeri fino a quando non digiteremo lo zero, il cui inserimento provocherà l'uscita dal ciclo consentendo la visualizzazione della somma e della media. La differenza con il precedente problema consiste nel fatto che non possiamo inserire più di 50 numeri. Pertanto, occorre prestare attenzione al caso in cui l'utente non abbia ancora digitato lo zero ma abbia inserito 50 numeri, eventualità che porterà comunque a terminare l'inserimento. Per far questo dobbiamo apportare una modifica al ciclo che, in questo caso, prevedrà due condizioni legate dall'operatore logico AND: la prima verificherà che il numero inserito sia diverso da zero e la seconda accerterà che la quantità di numeri inseriti sia minore di 50.

### Analisi dei dati

NOME	FUNZIONE	TIPO	DESCRIZIONE
Num	I	Intero	Numero richiesto in input
Somma	O	Intero	Somma dei numeri inseriti in input
Media	O	Reale	Media dei numeri inseriti in input
Conta	L	Intero	Contatore del ciclo

### Formalizzazione dell'algoritmo



#### ALGORITMO SommaMedia3

##### VARIABILI

Num, Somma, Conta: **INTERO**

Media: **REALE**

##### INIZIO

Somma ← 0

Conta ← 0

**SCRIVI**("Inserisci un numero")

**LEGGI**(Num)

**MENTRE**((Num ≠ 0) **AND** (Conta ≠ 50)) **ESEGUI**

Somma ← Somma + Num

Conta ← Conta + 1

**SCRIVI**("Inserisci un numero")

**LEGGI**(Num)

##### FINEMENTRE

**SE**(Conta = 0)

##### ALLORA

Media ← 0

##### ALTRIMENTI

Media ← Somma / Conta

##### FINESE

**SCRIVI**("La somma è ", Somma)

**SCRIVI**("La media è ", Media)

##### FINE



Realizzare un algoritmo che, data in input una sequenza di N numeri, determini il massimo tra essi.

### Analisi del problema

Chiediamo innanzitutto il numero N di numeri e successivamente, attraverso un ciclo precondizionale, chiediamo all'utente di inserire N volte i numeri da analizzare.

Per quanto riguarda la ricerca del massimo, ipotizziamo che il primo numero inserito sia il massimo (ed effettivamente lo è, essendo il primo!). In seguito, per ogni numero della sequenza inserito, confrontiamo tale numero con l'ipotetico massimo che abbiamo stabilito. Se il numero inserito risulta maggiore del massimo, aggiorniamo il massimo, altrimenti il numero inserito viene ignorato e si passa all'analisi del successivo.

### Analisi dei dati

NOME	FUNZIONE	TIPO	DESCRIZIONE
N	I	Intero	Numero di elementi da inserire
m	I	Intero	Numero richiesto in input
Max	O	Intero	Massimo elemento presente nella sequenza di numeri inseriti
Conta	L	Intero	Contatore del ciclo

### Formalizzazione dell'algoritmo

#### ALGORITMO Massimo

##### VARIABILI

N, Num, Max, Conta: **INTERO**

##### INIZIO

Conta ← 1

**SCRIVI**("Numeri da inserire")

**LEGGI**(N)

**MENTRE**(Conta ≤ N) **ESEGUI**

**SCRIVI**("Inserisci un numero")

**LEGGI**(Num)

**SE**(Conta = 1)

**ALLORA**

            Max ← Num

**ALTRIMENTI**

**SE**(Num > Max)

**ALLORA**

                Max ← Num

**FINESE**

**FINESE**

    Conta ← Conta + 1

**FINEMENTRE**

**SCRIVI**("Il massimo è", Max)

**FINE**





**1.** Analizza il seguente pseudocodice:

```
► ALGORITMO Uno
► VARIABILI
►   I, R: INTERO
► INIZIO
►   I ← 0
►   MENTRE(I > 2) ESEGUI
►     R ← R + 1
►   FINEMENTRE
►   R ← 5
► FINE
```

- Il corpo del ciclo MENTRE (cioè l'istruzione  $R \leftarrow R + 1$ ) sarà mai eseguito?
- Se la risposta è no, per quale motivo?



**2.** Analizza il seguente pseudocodice:

```
► ALGORITMO Due
► VARIABILI
►   I, R: INTERO
► INIZIO
►   I ← 0
►   R ← 0
►   MENTRE(I ≤ 10) ESEGUI
►     R ← R + 1
►     I ← I + 1
►   FINEMENTRE
►   R ← 5
►   SCRIVI(R)
► FINE
```

- Quante volte viene eseguito il ciclo MENTRE?
- Qual è il valore della variabile R all'uscita dal ciclo (cioè subito dopo l'esecuzione dell'istruzione FINEMENTRE)?
- Qual è il valore visualizzato della variabile R?



**3.** Quale sequenza produce in output il seguente pseudocodice?

```
► ALGORITMO Tre
► VARIABILI
►   I: INTERO
► INIZIO
►   I ← 1
►   MENTRE(I > 19) ESEGUI
►     SCRIVI(I)
►     I ← I + 1
►   FINEMENTRE
► FINE
```

- Se il valore di I fosse partito da 0, quale sequenza sarebbe stata prodotta in output?



4. Analizza il seguente pseudocodice:



**ALGORITMO** Quattro

**VARIABILI**

I, R: INTERO

**INIZIO**

I ← 1

R ← 0

**MENTRE**(I ≤ 10) **ESEGUI**

R ← R + 1

I ← I + 2

**FINEMENTRE**

**SCRIVI**(R)

**FINE**

Quante volte viene eseguito il ciclo **MENTRE**?

Qual è il valore della variabile R all'uscita dal ciclo?

Qual è il valore visualizzato della variabile R?

Quale ritocco apporteresti alla variabile I affinché il ciclo non venga mai avviato?

Quale ritocco apporteresti alla variabile I affinché il ciclo venga eseguito una sola volta?

5. Considera il seguente pseudocodice:



**ALGORITMO** Sei

**VARIABILI**

Numero1, Numero2, Effetto: INTERO

**INIZIO**

**LEGGI**(Numero1, Numero2)

Effetto ← 0

**MENTRE**(Numero1 ≠ 0) **ESEGUI**

SE(Numero1 MOD 2 ≠ 0)

**ALLORA**

Effetto ← Effetto + Numero2

**FINESE**

Numero1 ← Numero1 / 2

Numero2 ← Numero2 \* 2

**FINEMENTRE**

**SCRIVI**(Effetto)

**FINE**

Qual è l'output prodotto dallo pseudocodice nel caso in cui i valori delle variabili di input siano Numero1 = 37 e Numero2 = 41?

Qual è l'output prodotto dallo pseudocodice nel caso in cui i valori delle variabili di input siano Numero1 = 31 e Numero2 = 17?

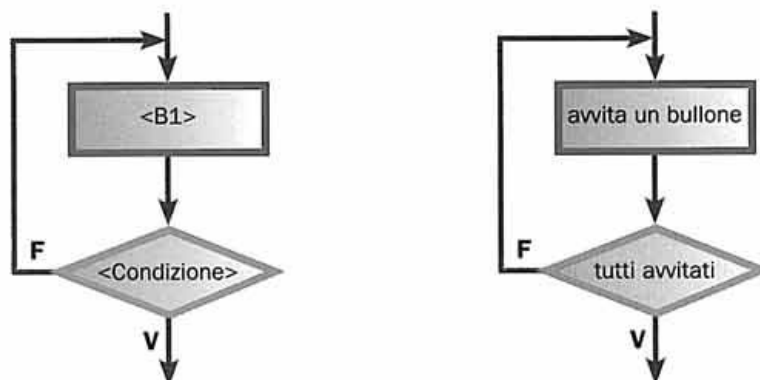
Che cosa rappresenta il risultato fornito in output?



## Il costrutto iterativo postcondizionale

Analizziamo il secondo costrutto iterativo chiamato **costrutto iterativo postcondizionale** (oppure *costrutto iterativo con controllo in coda*) poiché, come dice il nome stesso, il controllo della condizione viene effettuato dopo l'esecuzione delle istruzioni che compongono il ciclo.

Diamo uno sguardo alla descrizione sintattica illustrata nella seguente figura:



Dopo aver eseguito per la prima volta il ciclo, si valuta la *<Condizione>*. Se è falsa si riesegue il blocco *<B1>*, altrimenti, ossia se la *<Condizione>* è vera, si esce dal ciclo e si passa a eseguire l'istruzione successiva al FINCHÉ. Ovviamente, anche in questo caso l'espressione logica presente in *<Condizione>* rappresenta la **guardia del ciclo**, mentre il blocco di istruzioni *<B1>* è il **corpo del ciclo**.

Il costrutto iterativo postcondizionale presenta notevoli differenze con quello precondizionale. Esaminiamo con attenzione le informazioni riportate nei riquadri seguenti.

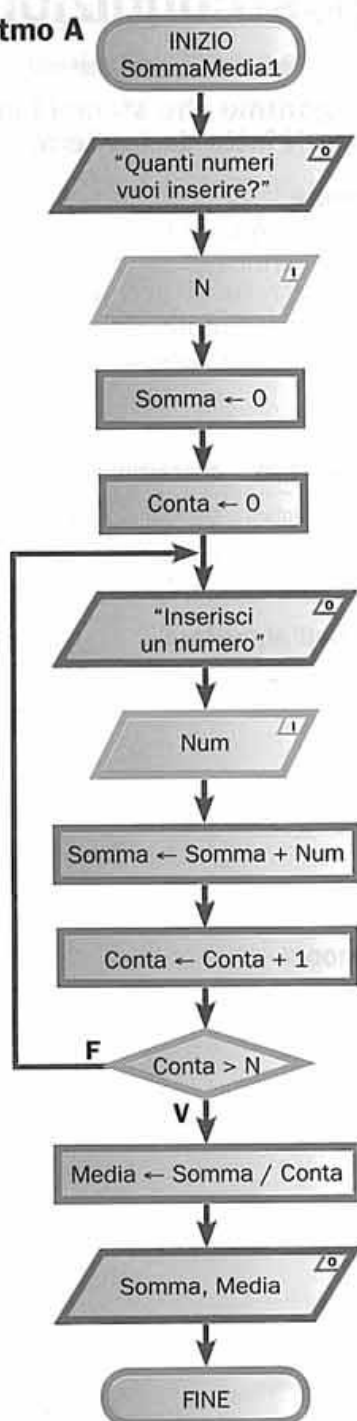
Costrutto iterativo MENTRE	Costrutto iterativo RIPETI FINCHÉ
1. La condizione viene controllata prima di eseguire il blocco B1.	1. La condizione viene controllata dopo aver eseguito il blocco B1.
2. Il blocco B1 può non essere mai eseguito.	2. Il blocco B1 è eseguito almeno una volta.
3. Il blocco B1 è eseguito fintantoché la condizione rimane vera.	3. Il blocco B1 è eseguito fintantoché la condizione rimane falsa.

### Costrutto postcondizionale e precondizionale a confronto

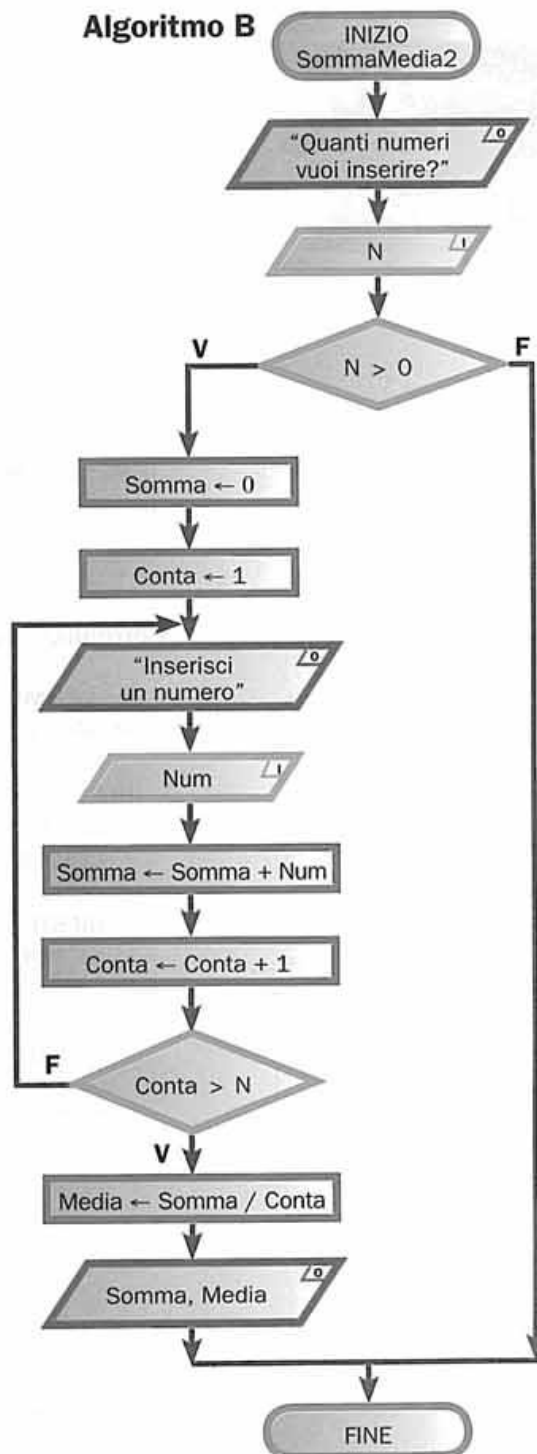
Durante la stesura di un algoritmo ci si può trovare nella situazione in cui il corpo di un ciclo potrebbe non essere eseguito neppure una volta. Con l'utilizzo di un costrutto precondizionale il problema non si pone, ma se si vuole utilizzare quello postcondizionale è necessario eseguire un test prima di entrarci. Riferiamoci al problema risolto negli esempi dell'unità precedente che richiedeva di realizzare un algoritmo per visualizzare la somma e la media di *N* numeri e proviamo a risolverlo utilizzando quest'ultimo costrutto iterativo. Nella soluzione **A** si evince facilmente che, se digitiamo il valore zero alla richiesta della variabile *N*, l'esecuzione dell'algoritmo continua poiché la condizione è presente in fondo al ciclo. Per fare in modo che il ciclo venga eseguito solo se l'utente digita un valore diverso da zero, dobbiamo apportare una modifica (così come riportato nella variante **B**) che obbliga l'utente a inserire un valore della variabile *N* maggiore di 0. Vediamo i due algoritmi.



Algoritmo A



Algoritmo B



Ci si accorge facilmente che il costrutto iterativo precondizionale risulta più semplice e immediato e soddisfa completamente le esigenze in ogni situazione garantendo, inoltre, minori possibilità di errore. D'altro canto, il ciclo postcondizionale è utile per il controllo dei dati in fase di input, quando il valore inserito dall'utente deve essere validato prima di procedere con l'elaborazione. Supponiamo, per esempio, che il problema richieda di inserire un numero corrispondente ai mesi dell'anno. Con l'utilizzo di un costrutto postcondizionale possiamo vincolare l'utente all'inserimento corretto dei dati. Il seguente frammento di algoritmo ci fornisce la conferma:

**RIPETI**

**SCRIVI**("Inserisci un numero compreso tra 1 e 12")

**LEGGI**(N)

**FINCHÉ**((N ≥ 1) AND (N < 12))

Il costrutto iterativo postcondizionale può essere trasformato in uno precondizionale semplicemente duplicando il blocco interno al ciclo e negando la condizione:

**RIPETI**

<B1>

**FINCHÉ**(<Condizione>)

<B1>

**MENTRE NOT**(<Condizione>) **ESEGUI**

<B1>

**FINEMENTRE**

È doveroso ricordare che il nostro approccio metodologico/didattico prevede un ciclo postcondizionale in cui il blocco viene ripetuto finché la condizione rimane falsa. I moderni linguaggi di programmazione mettono a disposizione costrutti iterativi postcondizionali che iterano quando la condizione è vera.





# Il costrutto iterativo postcondizionale



**Realizzare un algoritmo che stampi i numeri interi dispari minori o uguali a  $N$ , con  $N$  letto da tastiera.**

## Analisi del problema

Innanzitutto occorre acquisire in input il numero  $N$ . Successivamente si inizializza un contatore e si verifica se il suo valore è pari o dispari (attraverso la funzione MOD per il calcolo del resto di una divisione intera). Se il valore è dispari, allora si visualizza il valore del contatore, altrimenti lo si ignora. Il procedimento continua fino a che il valore del contatore si mantiene inferiore a  $N$ .

## Analisi dei dati

NOME	FUNZIONE	TIPO	DESCRIZIONE
$N$	I	Intero	Quantità di numeri da inserire in input
Conta	O	Intero	Contatore del ciclo

## Formalizzazione dell'algoritmo

### ALGORITMO PariDispari

#### VARIABILI

$N$ , Conta: INTERO

#### INIZIO

SCRIVI("Inserisci il valore di  $N$ ")

LEGGI( $N$ )

Conta  $\leftarrow 1$

#### RIPETI

SE(Conta MOD 2 = 1)

ALLORA

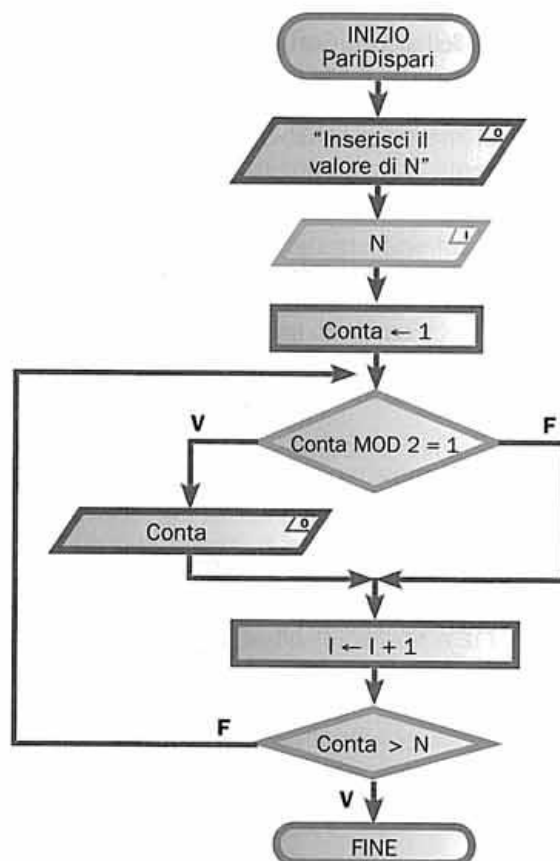
SCRIVI(Conta)

FINESE

$I \leftarrow I + 1$

FINCHÉ(Conta >  $N$ )

#### FINE





Realizzare un algoritmo che calcoli il prodotto tra due numeri interi utilizzando la sola operazione di somma.

### Analisi del problema

Dopo aver richiesto in input due numeri  $A$  e  $B$  strettamente positivi, si tratta di sommare il numero  $A$  con se stesso per un numero di volte pari al numero  $B$ . Per poter individuare il numero di volte che deve essere eseguita la somma e, quindi, per comprendere quando ci si deve fermare, decrementeremo di uno il valore del numero  $B$  ogni volta che viene eseguita la somma. Questa operazione si interromperà quando il valore di  $B$  sarà uguale a 0.

### Analisi del problema

NOME	FUNZIONE	TIPO	DESCRIZIONE
A	I	Intero	Valore del moltiplicando
B	I	Intero	Valore del moltiplicatore
Prodotto	O	Intero	Valore del prodotto

### Formalizzazione dell'algoritmo

#### ALGORITMO ProdottoAB

##### VARIABILI

A, B, Prodotto: **INTERO**

##### INIZIO

SCRIVI("Inserisci i valori di A e B")

LEGGI(A, B)

Prodotto  $\leftarrow$  0

##### RIPETI

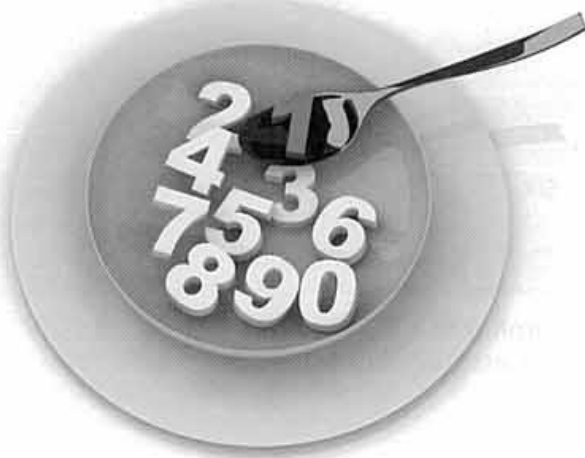
Prodotto  $\leftarrow$  Prodotto + A

B  $\leftarrow$  B - 1

FINCHÉ(B = 0)

SCRIVI("Il prodotto è ", Prodotto)

##### FINE





■ ■ ■ **1.** Analizza il seguente pseudocodice:

```
▶ ALGORITMO Uno  
▶ VARIABILI  
▶   A: INTERO  
▶ INIZIO  
▶   A ← 5  
▶   RIPETI  
▶     A ← A * 2  
▶     SCRIVI(A)  
▶   FINCHÉ(A > 50)  
▶ FINE
```

- Qual è l'output fornito?
- Quante volte viene eseguito il ciclo?

■ ■ ■ **2.** Analizza il seguente pseudocodice:

```
▶ ALGORITMO Due  
▶ VARIABILI  
▶   A: INTERO  
▶ INIZIO  
▶   A ← 5  
▶   RIPETI  
▶     A ← A * 2  
▶   FINCHÉ(A > 50)  
▶   SCRIVI(A)  
▶ FINE
```

- Qual è l'output fornito?
- Quante volte viene eseguito il ciclo?

■ ■ ■ **3.** Analizza il seguente pseudocodice:

```
▶ ALGORITMO Tre  
▶ VARIABILI  
▶   A: INTERO  
▶ INIZIO  
▶   A ← 5  
▶   RIPETI  
▶     A ← A * 2  
▶   FINCHÉ(A < 50)  
▶   SCRIVI(A)  
▶ FINE
```

- Qual è l'output fornito?
- Quante volte viene eseguito il ciclo?



4. Il seguente algoritmo presenta un loop infinito. Individua l'errore.

☐☐☐

▶ ALGORITMO Quattro

▶ VARIABILI

▶ X, Y: INTERO

▶ INIZIO

▶  $Y \leftarrow 7$

▶  $X \leftarrow 1$

▶ RIPETI

▶  $X \leftarrow Y$

▶  $Y \leftarrow Y + 1$

▶ FINCHÉ( $Y = 0$ )

▶ SCRIVI(X)

▶ FINE

5. Trova il valore delle variabili I e R alla fine dell'esecuzione del seguente pseudocodice completando la tavola di traccia:

☐☐☐

▶ ALGORITMO Cinque

▶ VARIABILI

▶ I, R: INTERO

▶ INIZIO

▶  $I \leftarrow 1$

▶  $R \leftarrow 0$

▶ RIPETI

▶  $R \leftarrow R + 1$

▶ SE( $R \neq 2$ )

▶ ALLORA

▶  $I \leftarrow 0$

▶ ALTRIMENTI

▶  $I \leftarrow 1$

▶ FINESE

▶ FINCHÉ( $I \neq 0$ )

▶ FINE

ISTRUZIONE	?	I	R
$I \leftarrow 1$		1	
$R \leftarrow 0$			0

• Quante volte viene eseguito il ciclo MENTRE?

6. Trova il valore delle variabili I e R alla fine dell'esecuzione del seguente pseudocodice completando la tavola di traccia:

☐☐☐

▶ ALGORITMO Sei

▶ VARIABILI

▶ I, R: INTERO

▶ INIZIO

▶  $I \leftarrow 1$

▶  $R \leftarrow 3$

▶ RIPETI

▶ SE( $I \neq 2$ )

▶ ALLORA

▶  $R \leftarrow R - 1$

▶ ALTRIMENTI

▶  $R \leftarrow R + 1$

▶ FINESE

▶  $I \leftarrow I + 1$

▶ FINCHÉ( $I < 0$ )

▶ FINE

ISTRUZIONE	?	I	R
$I \leftarrow 1$		1	
$R \leftarrow 3$			3



# I costrutti iterativi derivati

## LEZIONE

# 22

I costrutti iterativi pre e postcondizionali hanno una caratteristica fondamentale che li accomuna: la condizione che regola l'esecuzione del blocco iterativo. Tale condizione è rappresentata da un'espressione logica (anche complessa) che confronta alcune variabili e, prima o poi, consente di uscire dal blocco iterativo evitando il fatale *loop infinito*.

Questi due costrutti iterativi sono stati ampiamente utilizzati nelle situazioni in cui non è possibile conoscere a priori quante volte deve essere eseguito il ciclo. Sono, pertanto, dei **costrutti iterativi indeterminati** che fanno sì che *il corpo del ciclo venga eseguito per un numero indefinito di volte*. Durante la risoluzione dei problemi, però, ci si può trovare nella situazione in cui il numero di iterazioni è determinato a priori, cioè assume un valore definito e noto.

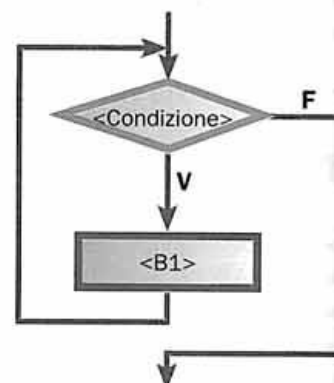
Presentiamo alcuni esempi di processi descrivibili con iterazioni indefinite e definite:

Iterazioni indefinite	Iterazioni definite
Mentre diluvia pensa a studiare	Inserisci 40 foto nell'album
Mescola la crema finché si addensa	Per 5 giorni devi prendere la medicina
Mentre la terra è secca inaffia la pianta	Invia 100 email di Buon Natale agli amici
Finché non ti senti dissetato continua a bere	Scrivi 20 lettere

Il **costrutto iterativo determinato** (chiamato anche *costrutto iterativo enumerativo*) è un costrutto iterativo derivato da quello precondizionale e permette di ripetere un blocco di istruzioni non in base al valore di verità di una condizione, ma in base al numero di volte che si vuole ripetere il blocco. In questo particolare costrutto iterativo precondizionale la condizione di arresto è indicata dal valore massimo che può raggiungere una variabile denominata **indice** utilizzata come contatore.

La sintassi di questo nuovo costrutto è la seguente:

```
PER <Indice> ← <Inizio> [INDIETRO] A <Fine> [PASSO <N>] ESEGUI
    <B1>
FINEPER
```



In questa struttura l'<Indice>, che serve da controllo per la ripetizione, assume inizialmente il valore indicato da <Inizio>. Una volta entrati nel ciclo si procede con l'esecuzione del blocco di istruzioni racchiuso tra le parole chiave PER e FINEPER. L'esecuzione di questo blocco viene ripetuta e, a ogni ripetizione, viene incrementato il valore dell'<Indice>. Il ciclo continuerà a essere eseguito fintantoché l'<Indice> si mantiene minore o uguale al valore di <Fine>. La ripetizione del blocco, di conseguenza, si arresta quando l'<Indice> diviene maggiore di <Fine>.

Diamo uno sguardo ai seguenti algoritmi risolutivi del problema: *dati in input 5 numeri visualizzare la loro somma e la loro media*. Le due versioni proposte sono state realizzate utilizzando il costrutto iterativo precondizionale e il costrutto iterativo determinato e indicizzato.

**ALGORITMO** SommaMedia1**VARIABILI**Num, Somma, Conta: **INTERO**Media: **REALE****INIZIO**Somma  $\leftarrow$  0Conta  $\leftarrow$  0**MENTRE**(Conta < 5) **ESEGUI****LEGGI**(Num)Somma  $\leftarrow$  Somma + NumConta  $\leftarrow$  Conta + 1**FINEMENTRE**Media  $\leftarrow$  Somma / Conta**SCRIVI**("La somma è", Somma)**SCRIVI**("La media è", Media)**FINE****ALGORITMO** SommaMedia2**VARIABILI**Num, Somma, Conta: **INTERO**Media: **REALE****INIZIO**Somma  $\leftarrow$  0**PER** Conta  $\leftarrow$  1 **A** 5 **ESEGUI****LEGGI**(Num)Somma  $\leftarrow$  Somma + Num**FINEPER**Media  $\leftarrow$  Somma / Conta**SCRIVI**("La somma è", Somma)**SCRIVI**("La media è", Media)**FINE**

serviamo le differenze.

In questo costrutto la variabile *Conta*, pur avendo la funzione di contatore, è più propriamente definita come **variabile indice**.

La variabile indice *Conta* non necessita dell'**inizializzazione** (cioè dell'assegnazione del suo valore di partenza) esterna al costrutto, in quanto viene inizializzata durante l'impostazione del ciclo. Non è così nel costrutto **MENTRE**, dove, invece, notiamo l'inizializzazione della variabile (*Conta*  $\leftarrow$  1).

La variabile indice non necessita dell'**incremento**. Nell'algoritmo, infatti, non troviamo l'istruzione *Conta*  $\leftarrow$  *Conta* + 1 che è sottintesa nel costrutto.

Il ciclo **PER** si comporta esattamente come il ciclo **MENTRE** in quanto, se il valore di **INIZIO** è maggiore di **FINE**, non viene mai eseguito.

Se l'algoritmo avesse richiesto in input *N* numeri, anziché 5, avremmo sempre potuto utilizzare il costrutto **PER**, pur non conoscendo anticipatamente di quanti numeri si sarebbe dovuta fare la somma. Infatti, dopo la richiesta in input del valore di *N*, saremmo riusciti a ottenere questa informazione indispensabile per applicare questo costrutto e avremmo scritto:

```
PER Conta  $\leftarrow$  1 A N ESEGUI
```

Per chiarire gli ultimi due elementi presenti nella struttura generale di questo costrutto. Nella sintassi, la parola racchiusa tra parentesi quadre **INDIETRO A** permette l'utilizzo del costrutto nel caso in cui la variabile indice debba subire dei decrementi per giungere dal valore <Inizio> al valore <Fine>. Supponiamo, per esempio, di dover stampare i numeri compresi tra 13 e 5. Utilizzando i concetti studiati si comprende che in questo caso il ciclo **PER** è il seguente:

```
PER Conta  $\leftarrow$  13 INDIETRO A 5 ESEGUI
```

Abbiamo visto che l'utilizzo di questo costrutto iterativo non prevede l'incremento della variabile indice, poiché ciò avviene automaticamente. In alcune situazioni problematiche, però, potremmo aver bisogno di incrementare l'indice non di una sola unità, ma di una quantità diversa. In queste circostanze occorre utilizzare **PASSO <N>**, dove la variabile *N* indica la quantità della quale viene incrementata la variabile indice. Per esempio, se volessimo visualizzare i primi 10 numeri dispari potremmo utilizzare il seguente ciclo:

```
PER Conta  $\leftarrow$  1 A 50 PASSO 2 ESEGUI
```

```
SCRIVI(Conta)
```

```
FINEPER
```

In questo modo, a ogni iterazione, la variabile *Conta* viene incrementata di due unità, risolvendo così il problema dato.

# Tutto in test...a

## PROVE OGGETTIVE PER LA VERIFICA DELLE CONOSCENZE

**1** Per trasformare l'algoritmo in un insieme di istruzioni comprensibili dal computer occorre:

- a) il diagramma a blocchi
- b) lo pseudocodice
- c) il linguaggio di programmazione
- d) la BNF

**2** Il teorema di Böhm-Jacopini afferma che:

- a) in ogni algoritmo è possibile utilizzare le istruzioni di salto
- b) in un algoritmo non possono essere inseriti cicli infiniti
- c) ogni algoritmo può essere espresso solo attraverso le strutture di sequenza, selezione e iterazione

**3** Stabilisci se le seguenti affermazioni sono vere o false:

- La codifica è la traduzione dell'algoritmo in un linguaggio di programmazione \_\_\_\_\_ ☐ V ☐ F
- Il programma è un algoritmo scritto in pseudocodifica \_\_\_\_\_ ☐ V ☐ F
- La realizzazione di un algoritmo è affidata al programmatore \_\_\_\_\_ ☐ V ☐ F
- L'analisi dei dati di un problema si svolge con l'ausilio della tabella delle variabili \_\_\_\_\_ ☐ V ☐ F
- Il test dell'algoritmo prende il nome di trace \_\_\_\_\_ ☐ V ☐ F
- Il controllo del trace avviene tramite la tabella delle variabili \_\_\_\_\_ ☐ V ☐ F
- La manutenzione del software è una fase di esclusiva manutenzione \_\_\_\_\_ ☐ V ☐ F

**4** Che cos'è l'indentazione?

- a) una tecnica che consente di scrivere le istruzioni in maniera incolonnata così da indicare la loro dipendenza
- b) una tecnica che consente di scrivere le istruzioni una sotto l'altra per indicare il flusso del controllo
- c) una tecnica che consente di scrivere le istruzioni tramite i diagrammi a blocchi
- d) una tecnica che consente di scrivere pseudocodice leggibile

**5** Qual è la differenza tra proposizione e enunciato?

**6** Qual è la differenza tra enunciato semplice ed enunciato composto?

**7** Che cosa sono i connettivi logici?

**8** Nella disgiunzione XOR:

- a) la proposizione risultante è vera se entrambe le proposizioni sono vere
- b) la proposizione risultante è vera se solo una delle due proposizioni è vera
- c) la proposizione risultante è vera se entrambe le proposizioni sono false

**9** Relativamente ai cicli, che cos'è un dato tappo?

- a) un valore prestabilito che ha il compito di introdurre la sequenza
- b) un valore prestabilito che ha il compito di chiudere la sequenza
- c) un valore prestabilito che ha il compito di segnalare un errore all'interno della sequenza

**10** Che cosa si intende con *guardia del ciclo*?

**11** Relativamente ai cicli MENTRE e RIPETI:

- a) entrambi si usano solo se è noto il numero di iterazioni
- b) entrambi i cicli eseguono almeno una volta il gruppo di azioni contenute al loro interno
- c) il valore di verità del ciclo RIPETI è opposto a quello del ciclo MENTRE equivalente

**12** Relativamente ai costrutti iterativi:

- a) qualsiasi tipo di ciclo può essere risolto mediante il ciclo MENTRE
- b) il ciclo RIPETI e il ciclo PER non possono mai essere eliminati per risolvere le iterazioni
- c) il ciclo PER può essere utilizzato anche come costrutto iterativo postcondizionale

**13** Come si fa a trasformare un costrutto iterazione postcondizionale in un equivalente precondizionale?

**14** Nel ciclo PER:

- a) non si deve incrementare il contatore di ciclo
- b) il contatore assume il nome di variabile indice
- c) l'incremento del contatore può essere solo unitario
- d) la condizione è posta in coda

# Training

PROVE APERTE PER LA VERIFICA DELLE ABILITÀ

## OSTRUTTO SEQUENZA

### 1 Tracing

Indica l'output prodotto dal seguente algoritmo:

ALGORITMO Uno

VARIABILI

A: INTERO

INIZIO

LEGGI(A)

$A \leftarrow A * 3$

$A \leftarrow (A + 2) - (A * 3)$

SCRIVI(A)

FINE

### 2 Tracing

Indica l'output prodotto dal seguente algoritmo:

ALGORITMO Due

VARIABILI

A, B, C, D: INTERO

INIZIO

LEGGI(A)

$B \leftarrow A + 2$

LEGGI(B)

$C \leftarrow A + B$

SCRIVI(C)

$D \leftarrow (A - B) + C$

SCRIVI(D)

FINE

### 3 Tracing

Trova il valore della variabile A visualizzato al termine del seguente algoritmo:

ALGORITMO Uno

VARIABILI

A: INTERO

INIZIO

LEGGI(A)

$A \leftarrow A * 3$

$A \leftarrow (A + 2) - (A * 3)$

LEGGI(A)

SCRIVI(A)

FINE

### 4 Gradi, primi e secondi

Scrivi un algoritmo che, data in input la misura di un angolo in gradi (G), primi (P) e secondi (S), determini la sua ampiezza espressa in secondi.

### 5 Media di tre numeri

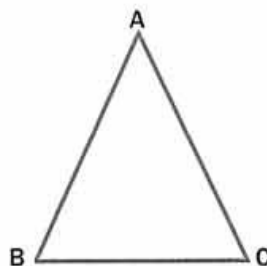
Scrivi un algoritmo che, dati in input tre numeri, ne determini la media.

### 6 Area del cerchio

Scrivi un algoritmo che determini l'area del cerchio circoscritto a un quadrato.

### 7 Perimetro e area triangolo

Scrivi un algoritmo che, date in input le dimensioni AB e BC del seguente triangolo isoscele, ne determini perimetro e area.



### 8 Età di una persona

Scrivi un algoritmo che, dati in input il cognome, il nome e l'anno di nascita di una persona, fornisca in output la sua età.

### 9 Equazione di primo grado

Scrivi un algoritmo che calcoli la radice dell'equazione di primo grado  $ax=b$ .

### 10 Calcolo secondi

Dati un numero di giorni, uno di ore, uno di minuti e uno di secondi, calcola il numero di secondi corrispondente.

Esempio:

Giorni = 2, Ore = 3, Minuti = 23, Secondi = 7

→ N = 184987

### 11 Calcolo litri di vino

Supponiamo di essere andati da un contadino a comprare il vino. Il contadino ci vende il vino a 1,70 euro il litro. Costruisci un algoritmo che, ricevuto in ingresso il numero di litri acquistati, comunichi l'importo da pagare.

### 12 Vernici e additivi

Per produrre una vernice sono necessari 10 grammi di additivo ogni chilo di prodotto fino a 10 chili e 5 grammi al chilo per i chili eccedenti. Stabilisci la quantità di additivo necessaria in base al quantitativo di vernice richiesto.

# Training

## PROVE APERTE PER LA VERIFICA DELLE ABILITÀ

### 13 Telefonate e scatti

□□□

Scrivi un algoritmo che determini il numero di scatti effettuati da un utente telefonico e l'ammontare della sua bolletta. Vengono forniti in input i seguenti dati:

- il nome dell'utente;
- il numero di scatti emersi dalla lettura della bolletta precedente;
- il numero di scatti emersi dalla lettura della bolletta attuale;
- il costo dello scatto.

Ricorda, inoltre, che per determinare il valore della bolletta occorre aggiungere un canone fisso il cui importo viene anch'esso fornito in input.

### 14 Spese condominiali

□□□

In un condominio si decide di calcolare una tassa *tantum* rispetto alle dimensioni dell'appartamento, espresse in metri quadri, in ragione di euro *K* per ogni metro quadro. All'importo così calcolato viene aggiunta una quota fissa di euro *X* e una percentuale del *T*%. Scrivi un algoritmo che, dati in input i valori di *K*, *X* e *T*, determini l'ammontare della tassa.

## COSTRUTTO SELEZIONE

### 15 Due numeri in ordine crescente

□□□

Scrivi un algoritmo che, dati in input due numeri, li scriva in ordine crescente.

### 16 Pari o dispari

□□□

Scrivi un algoritmo che, dato in input un numero, stabilisca se è pari o dispari.

### 17 Tipo di triangolo

□□□

Scrivi un algoritmo che, date in input le misure dei lati di un triangolo, determini se il triangolo è scaleno, equilatero o isoscele.

### 18 Calcolo espressione

□□□

Scrivi un algoritmo che, dati in input due numeri interi *X* e *Y*, visualizzi il valore di  $X + N * B$  dove *N* è un numero compreso tra 2 e 50.

### 19 Conversione euro/lire/dollaro

□□□

Progetta un algoritmo che data una quantità di soldi in lire letta da tastiera restituisca la quantità corrispondente in euro (/2000) o in dollari (/2500), a seconda del valore assunto da una variabile moneta di tipo carattere.

### 20 Valore assoluto

□□□

Scrivi un algoritmo che, dati in input due numeri *A* e *B*, calcoli la somma di quello che ha valore assoluto maggiore e del doppio di quello che ha valore assoluto minore.

### 21 Equazione di secondo grado

□□□

Scrivi un algoritmo che calcoli le radici di un'equazione di secondo grado.

### 22 Somma spesa da un cliente

□□□

Calcola la somma spesa da un cliente in un negozio di abbigliamento tenendo conto delle seguenti condizioni:

- per spese inferiori a euro 50, sconto 10%;
- per spese inferiori a euro 100, sconto 20%;
- per spese superiori a euro 100, sconto 30%.

### 23 Acquisto di merce

□□□

Calcola l'imponibile relativo all'acquisto di una certa merce, noti la quantità acquistata, il prezzo unitario e lo sconto, tenendo presente che lo sconto viene applicato soltanto per quantità superiori a un valore dato.

### 24 Maggiore e minore

□□□

Scrivi un algoritmo che, dati in input tre numeri, determini:

- il maggiore;
- il minore;
- la differenza tra il maggiore e il minore.

### 25 Metodo di Cramer

□□□

Sia dato il sistema lineare a due incognite. Scrivi un programma che, acquisiti da tastiera i coefficienti e il termine noto delle equazioni, risolva il sistema con il metodo di Cramer.

Metodo di Cramer:

$$\text{siano } D_s = ab' - a'b$$

$$D_x = cb' - c'b$$

$$D_y = ac' - a'c$$

Si possono verificare le seguenti situazioni:

- se  $D_s = 0$  e  $D_x = 0$  il sistema è indeterminato
- se  $D_s = 0$  e  $D_x \neq 0$  il sistema è impossibile
- se  $D_s \neq 0$  le soluzioni del sistema sono:

$$x = \frac{D_x}{D_s} \quad y = \frac{D_y}{D_s}$$

# Training

PROVE APERTE PER LA VERIFICA DELLE ABILITÀ

## COSTRUTTO SELEZIONE MULTIPLA

### 26 Mesi dell'anno

Scrivi un algoritmo che, dato in input il numero corrispondente al mese dell'anno, visualizzi il numero di giorni di cui l'anno è composto.

### 27 Le quattro operazioni

Scrivi un algoritmo che, dati in input due numeri interi, visualizzi, a scelta dell'utente, il risultato di una delle quattro operazioni fondamentali (addizione, sottrazione, moltiplicazione, divisione).

### 28 Stipendi e trattenute

Sullo stipendio dei dipendenti di una ditta viene applicata una trattenuta fiscale in base alla seguente tabella:

Scaglione 1	Trattenuta 5 %
Scaglione 2	Trattenuta 10 %
Scaglione 3	Trattenuta 15 %
Scaglione 4	Trattenuta 25 %
Scaglione 5	Trattenuta 35 %

Scrivi un algoritmo che, dato in input lo scaglione di appartenenza di un dipendente, calcoli la trattenuta da versare.

### 29 Rettangoli e calcoli

Date le misure dei lati di un rettangolo a, b fornite da tastiera, scrivi un programma che calcoli il perimetro, l'area o la diagonale del rettangolo secondo la richiesta dell'utente. (Supponi che l'utente possa inserire come scelta: 1 = perimetro, 2 = area o 3 = diagonale).

### 30 Sconti e fatture

Un negoziante, per incrementare le sue vendite, prevede di applicare uno sconto progressivo sull'importo della fattura, in base al numero di pezzi acquistati, secondo la seguente tabella:

Numero pezzi acquistati	Sconto
1	10 %
3	20 %
5	30 %
10	35 %
> 10	40 %

Calcola lo sconto praticato e la somma che il cliente dovrà pagare.

### 31 Visitatori di una mostra

Per aumentare il numero di visitatori di una mostra, si decide di far pagare il biglietto d'ingresso differenziato in base all'età. Precisamente:

Età	Prezzo del biglietto
Inferiore a 5 anni	Gratuito
Fino a 10 anni	euro 2
Da 11 a 17 anni	euro 4
Da 18 a 26 anni	euro 5
Oltre 26 anni	euro 7

Scrivi un algoritmo che, data in input l'età, visualizzi l'importo del biglietto da pagare.

### 32 Agenzia immobiliare 1

Un'agenzia immobiliare, per incrementare le sue vendite, decide di abbassare i prezzi degli appartamenti e affigge la seguente tabella:

Distanza dal centro	Prezzo al mq.
Centro	euro 1.500
Zona 1	euro 1.200
Zona 2	euro 1.400
Zona 3	euro 1.300
Periferia	euro 1.000

### 33 Agenzia immobiliare 2

Scrivi un algoritmo che, date in input le dimensioni dell'appartamento in mq. e la zona di appartenenza, determini il prezzo dell'appartamento. Infine, per determinare il prezzo complessivo di vendita, deve essere aggiunta la percentuale dell'X% relativa alla provvigione spettante all'agenzia.

## COSTRUTTO ITERAZIONE

### 34 Valore assoluto di N numeri

Scrivi un algoritmo che, dati in input N numeri, determini, per ognuno di loro, il loro valore assoluto.

### 35 Visualizza i numeri pari 1

Scrivi un algoritmo che visualizzi i primi 100 numeri pari.

# Training

## PROVE APERTE PER LA VERIFICA DELLE ABILITÀ

### 36 Visualizza i numeri pari 2

□□□

Generalizza l'esercizio precedente facendolo operare su N numeri.

### 37 Visualizza i numeri dispari

□□□

Scrivi un algoritmo che visualizzi i primi N numeri dispari.

### 38 Somma dei numeri dispari

□□□

Scrivi un algoritmo che legga N numeri interi (con  $N < 100$ ) e calcoli la somma dei soli numeri dispari.

### 39 Maggiori e minori di un numero

□□□

Scrivi un algoritmo che, dati in input N numeri interi e un numero X, determini:

- quanti numeri sono maggiori di X
- quanti sono minori di X
- quanti sono uguali a X

### 40 Massimo e minimo

□□□

Scrivi un algoritmo che, dati in input N numeri interi, determini il massimo e il minimo.

### 41 Massimo, minimo e posizione

□□□

Scrivi un algoritmo che, data una sequenza di numeri chiusa dallo zero, determini il valore massimo, il valore minimo e la posizione dei due valori nell'ambito della sequenza (lo zero non deve essere considerato).

### 42 Temperatura massima e minima

□□□

Ogni giorno vengono registrate la temperatura massima e minima di ogni città. Scrivi un algoritmo che, date in input la temperatura massima e quella minima registrata e il nome della città corrispondente, visualizzi il nome della città più calda e di quella più fredda.

### 43 Prodotti con il prezzo più alto

□□□

Data una serie di coppie nome prodotto e prezzo, stabilisci qual è il prodotto che ha il prezzo più alto.

### 44 Sufficienze e insufficienze

□□□

Dati i voti riportati da alcuni studenti in una prova, stabilisci quanti sono insufficienti e quanti sufficienti

### 45 Media aritmetica e media geometrica

□□□

Scrivi un algoritmo che, dati in input N valori interi positivi, calcoli la media aritmetica e la media geometrica.

### 46 Multipli di un numero 1

□□□

Scrivi un algoritmo che, dato in input un numero intero, determini se è multiplo di un numero X anch'esso intero e richiesto in input.

### 47 Multipli di un numero 2

□□□

Stampa i multipli di un numero intero A compreso tra due numeri interi X e Y.

### 48 Altezze di alcune persone

□□□

Data una serie di altezze, conta le persone che hanno altezza compresa tra due valori limite inseriti in input.

### 49 Conta positivi e negativi

□□□

Scrivi un algoritmo che, dopo l'immissione di N numeri interi, consenta di:

- contare quanti sono i numeri positivi effettuandone la somma;
- contare quanti sono i numeri negativi effettuandone la somma;
- visualizzare i conteggi effettuati.

### 50 Multipli di 3

□□□

Scrivi un programma che visualizzi tutti i numeri da 10 a 100 multipli di 3.

### 51 Multipli di 3 e di 5

□□□

Scrivi un programma che visualizzi tutti i numeri da 10 a 100 multipli di 3 e di 5.

### 52 Numeri di Fibonacci

□□□

Scrivi un algoritmo che visualizzi i primi N elementi della successione di Fibonacci

### 53 Numeri primi 1

□□□

Scrivere un algoritmo che verifichi se un numero è primo.

### 54 Numeri primi 2

□□□

Scrivi un algoritmo che visualizzi i primi N numeri primi

### 55 Somma di numeri primi

□□□

Scrivi un programma che visualizza la somma dei primi N numeri primi (con n letto da tastiera).

# Training

## PROVE APERTE PER LA VERIFICA DELLE ABILITÀ

### 1 Minimo, massimo e media

Scrivi un programma che legga da tastiera N numeri reali (N richiesto da tastiera) ed effettui i seguenti calcoli visualizzandone il risultato:

- minimo,
- massimo,
- media dei valori.

### 2 Centinaia, decine e unità

Dato un numero intero inferiore a mille, determina il numero di centinaia, decine e unità. Per esempio, 123 è composto da un centinaio, due decine e tre unità.

### 3 Somma di pari e prodotto di dispari

Data una sequenza di N numeri interi, calcola la somma dei pari e il prodotto dei dispari.

### 4 Media aritmetica e scostamenti

Dati N numeri reali calcola la media aritmetica e indica qual è il valore per il quale si registra il massimo scostamento.

### 5 Moltiplicazioni e divisioni

Scrivi un algoritmo che, servendosi esclusivamente delle operazioni di addizione e sottrazione, calcoli rispettivamente il prodotto e il quoziente di due numeri interi positivi X e Y. Per quanto riguarda il quoziente, accertati di non incorrere in forme indeterminate.

### 6 Quattro operazioni

Supponendo che l'esecutore conosca le quattro operazioni, scrivi un algoritmo per il calcolo della potenza  $x^y$  di due interi non negativi x e y ricevuti in ingresso. Riscrivi poi l'algoritmo nel caso in cui l'esecutore conosca solo le operazioni di somma e sottrazione.

### 7 Somma dei quadrati

Dati tre numeri determina la somma dei quadrati dei due più piccoli.

### 8 Vocali e consonanti

Date N lettere maiuscole conta le vocali e le consonanti.

### 64 Minimo comune multiplo

Calcola il minimo comune multiplo tra due numeri interi positivi A e B procedendo nel seguente modo: confronta i due valori e somma al valore più piccolo se stesso, confronta la somma ottenuta con l'altro valore, somma un altro valore alla somma più piccola finché le due somme diventano uguali.

Esempio: A=3 B=5.

3+3,5 6,5+5 9+3,10 12,10+5

12+3,15 15,15

### 65 Successioni

Data la successione 1, 2, 4, 8, 16, 32..., in cui ogni elemento è il doppio del precedente, stabilisci qual è il primo termine maggiore di un valore intero N introdotto.

### 66 Potenze

Dato un numero reale N e uno intero M, calcola la potenza di N alla M con  $M \geq 0$ .

Variante: supponi che M possa assumere valori negativi.

### 67 Tracing

Osserva il seguente algoritmo espresso in pseudocodifica:

```
► INIZIO
► LEGGI(Numero)
► I ← 1
► MENTRE (I < Numero) ESEGUI
►   SCRIVI(I)
►   I ← I + 1
► FINEMENTRE
► FINE
```

Ora:

- enuncia il problema che risolve;
- trasforma il ciclo iterativo MENTRE in un ciclo iterativo RIPETI;
- trasforma il ciclo iterativo MENTRE in un ciclo iterativo determinato;
- se alla variabile Numero viene assegnato il valore zero, quante volte viene ripetuto il ciclo? (Serviti della tavola di traccia)

# Training

## PROVE APERTE PER LA VERIFICA DELLE ABILITÀ

### 68 Somma di quadrati

□□□

Scrivi un algoritmo che calcoli la somma dei quadrati dei primi  $K$  numeri naturali successivi a un numero naturale  $N$ , noti  $K$  e  $N$ .

### 69 Divisori interi

□□□

Scrivi un algoritmo che, dati in input  $N$  numeri, determini, per ognuno di essi, tutti i divisori interi.

### 70 Fattori primi

□□□

Scrivi un algoritmo che, dati in input  $N$  numeri, determini, per ognuno di essi, tutti i fattori primi.

### 71 Fattoriale

□□□

Scrivi un algoritmo che, dati in input  $N$  numeri, determini il fattoriale di ognuno.

### 72 Percentuali

□□□

Scrivere un algoritmo che, dati in input  $N$  numeri interi, determini la percentuale dei positivi, dei negativi, dei pari e dei dispari.

### 73 Quadrato di numeri

□□□

Scrivi un algoritmo che calcoli il quadrato dei primi  $N$  numeri naturali.

(Per risolvere il problema serviti della seguente regola: il quadrato di un numero  $x$  diverso da zero è uguale alla somma dei primi  $x$  numeri dispari. Per esempio: il quadrato di 5 è dato da  $1 + 3 + 5 + 7 + 9 = 25$ ).

### 74 Precedente e successivo

□□□

Un calcolatore riesce a eseguire soltanto il precedente e il successivo di un numero. Alla luce di questa informazione, scrivi un algoritmo che realizzi la somma di due numeri  $X$  e  $Y$ .

(Nota che, in questo caso, la somma può essere realizzata aggiungendo a  $X$  tante unità quante possiamo toglierne a  $Y$ ).

### 75 Successioni

□□□

Scrivi un algoritmo che, dato in input un numero intero positivo  $A$ , determini quanti termini della successione definita da:

$$x_0 = a + 1$$

$$x_i = x_{i-1} + 1 \text{ per } i = 1, 2, \dots$$

occorre sommare successivamente (partendo dal primo), per superare strettamente un limite dato in input.

### 76 Polinomi

□□□

Realizza un algoritmo che per un dato valore di  $X$  calcoli il valore del polinomio:

$$Pol = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$$

### 77 Cartellini e ore lavorate

□□□

Si vuole automatizzare il calcolo delle ore lavorative settimanali da retribuire a ciascun dipendente di una ditta. Scrivi un algoritmo che, date in input l'ora di entrata e l'ora di uscita riportate nel cartellino personale, calcoli, approssimativamente, il totale delle ore da retribuire.

### 78 Espressione

□□□

Data una espressione composta da addizioni e sottrazioni (in cui si alternano un numero e un operatore e che termina con il simbolo  $=$ ) calcola il risultato.

### 79 Elenco studenti 1

□□□

L'elenco degli studenti di una scuola riporta la classe frequentata dallo studente e il suo nome. L'elenco è ordinato per classi. Conta gli studenti di ogni classe. Per classe si intende l'anno di corso frequentato (1, 2, 3, 4, 5) e non la sezione. Non è noto a priori quante e quali siano le classi presenti nella scuola.

### 80 Elenco studenti 2

□□□

L'elenco degli studenti di una scuola riporta la classe, la sezione e il nome dello studente. L'elenco è ordinato per classe e per sezione. Calcola quanti studenti sono presenti nella classe di ciascuna sezione e quanti frequentano lo stesso anno.

### 81 Calcolo radice quadrata

□□□

Scrivi una funzione radice che calcoli la radice quadrata (intera) di un naturale  $N$ .

#### Suggerimento

Considera un intero  $X$  dopo l'altro a partire da 1, e calcolane il quadrato  $X * X$ : fermati appena tale quadrato supera  $N$ . Il precedente numero considerato ( $X - 1$ ) è il risultato.

### 82 Successioni 1

□□□

Scrivi un algoritmo che letto un numero  $A$  in input, generi e visualizzi i primi 100 valori della seguente successione:

$$5 + A, 10 + A, 15 + A, 20 + A, \dots$$

### 83 Successioni 2

□□□

Per ognuna delle seguenti successioni di numeri, costruisci l'algoritmo che le genera:

- 1, 3, 5, 7, 9, ..., 99
- 1, 4, 9, 16, 25, ..., 2500
- 1, -2, 3, -4, 5, -6, ..., -100
- $1/2, 2/3, 3/4, 4/5, \dots, 99/100$

# Training

## PROVE APERTE PER LA VERIFICA DELLE ABILITÀ

### 4 Successioni 3

Scrivi un algoritmo che legga da input una sequenza arbitraria di numeri  $a_1, a_2, a_3, a_4, \dots$  e restituisca nella variabile SOMMA la seguente espressione:  
 $SOMMA = a_1 + a_2^2 + a_3 + a_4^2 + \dots$  (dove  $a_2$  rappresenta l'elevamento di  $a$  al quadrato).

### 5 Successioni 4

Scrivi un algoritmo che legga da input una sequenza arbitraria di numeri  $a_1, a_2, a_3, a_4, \dots$  e restituisca nella variabile SOMMA la seguente espressione:  
 $SOMMA = (a_1 * a_2) + (a_3 * a_4) + \dots$

### 6 Successioni 5

Scrivi un algoritmo che legga da input una sequenza arbitraria di numeri  $a_1, a_2, a_3, a_4, \dots$  e restituisca nella variabile SOMMA la seguente espressione:  
 $SOMMA = (a_1 * a_2 * a_3) + (a_4 * a_5 * a_6) + \dots$

### 7 Successioni 6

Scrivi un algoritmo che legga da input una sequenza arbitraria di numeri  $a_1, a_2, a_3, a_4, \dots$  e restituisca nella variabile SOMMA la seguente espressione:  
 $SOMMA = (a_1 * a_2 * a_3) + (a_2 * a_3 * a_4) + \dots$

### 8 Successioni 7

Scrivi un algoritmo che legga da input una sequenza arbitraria di numeri  $a_1, a_2, a_3, a_4, \dots$  e restituisca nella variabile SOMMA la seguente espressione:  
 $SOMMA = (a_1 * a_2 * a_3) + (a_3 * a_4 * a_5) + \dots$

### 9 Conto corrente

Vogliamo gestire un conto corrente bancario. Realizza un algoritmo che, una volta inserito il saldo iniziale, richieda le varie operazioni (versamento, prelevamento) e la somma. L'algoritmo dovrà:

- segnalare eventuali scoperti e, in caso siano presenti, non permettere l'operazione;
- visualizzare il numero di versamenti e la somma complessivamente versata;
- visualizzare il numero di prelievi e la somma complessivamente prelevata;
- visualizzare il saldo finale.

### 0 Azienda di produzione

Una ditta produce pacchetti di sale. Tali pacchetti devono contenere 1 Kg di sale con una tolleranza in più o in meno del 2%. Costruisci un algoritmo che, data in ingresso la sequenza dei pesi dei pacchetti (che termina con 0), fornisca:

- il numero di pacchetti di peso corretto, sottopeso e sovrappeso;
- il peso totale della merce nei pacchetti a norma ed il peso totale nei pacchetti fuori norma.

### 91 Conversione binario-ottale

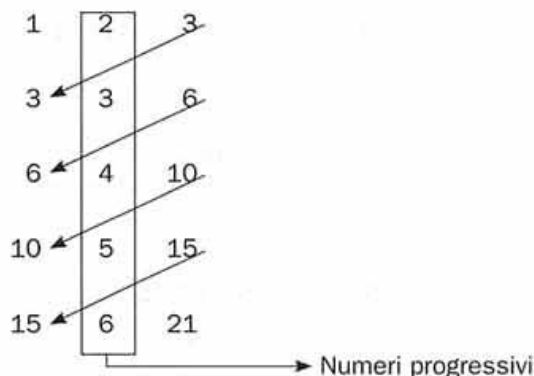
Scrivi un algoritmo che converta un numero intero  $N$  nel suo equivalente binario o ottale in funzione della scelta fatta dall'utente.

### 92 Distanza tra due date

Scrivere un algoritmo che, date in input due date, visualizzi la distanza intercorrente espressa in giorni.

### 93 Terne di numeri triangolari

Scrivere un algoritmo per il calcolo delle prime  $N$  terne di numeri triangolari. Ricordiamo che le terne di numeri triangolari sono legate dalla seguente relazione:



### 94 Triangolo di Tartaglia

Scrivere un algoritmo che stampi le prime  $N$  righe del triangolo di Tartaglia. Esempio:

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
```

### 95 Triangolo di Floyd

Scrivi un algoritmo per la rappresentazione delle prime  $N$  righe del triangolo di Floyd. Per esempio, se  $N=5$  il triangolo stampato sarà il seguente:

```
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```

# Training

PROVE APERTE PER LA VERIFICA DELLE ABILITÀ

## 96 Data della Pasqua

□□□

Un interessante algoritmo: stabilisci la data della Pasqua per qualsiasi anno. Traduci in pseudocodice i seguenti passi:

- assegna alla variabile  $y$  l'anno preso in considerazione;
- assegna alla variabile  $n$  la differenza tra 1900 e  $y$ ;
- assegna alla variabile  $a$  il resto della divisione tra la variabile  $n$  e 19;
- assegna alla variabile  $b$  il quoziente della divisione tra  $(7a+1)$  e 19;
- assegna alla variabile  $m$  il resto della divisione tra  $(11a+4b)$  e 29;
- assegna alla variabile  $q$  il quoziente della divisione tra  $n$  e 4;
- assegna alla variabile  $w$  il resto della divisione tra  $(n+q+31-m)$  e 7;
- la data della Pasqua è data da  $25-m-w$ . Se il risultato ottenuto è positivo, il mese in cui cadrà la Pasqua è aprile. Se il risultato è negativo, il mese sarà marzo e il giorno potrà essere stabilito secondo la seguente tabella:

Risultato	Data
0	31 marzo
-1	30 marzo
-2	29 marzo
.....	.....
-9	22 marzo

## PROPOSTE OPERATIVE SULLA LOGICA BOOLEANA

### 97 Enunciati

□□□

Riconosci tra le seguenti espressioni linguistiche quali sono enunciati e quali no:

- a il numero 5 è divisore del numero 24
- b 10 è un numero composto
- c attenti alla punteggiatura
- d il numero 7 è un numero dispari
- e zitto, arriva il preside
- f Roma è la capitale d'Italia
- g Gigi è un bel ragazzo
- h Spegni la radio
- i Il rettangolo ha due diagonali

### 98 Valore di verità 1

□□□

Attribuisci un valore di verità agli enunciati individuati al punto precedente.

### 99 Valore di verità 2

□□□

Attribuisci un valore di verità ai seguenti enunciati:

- a  $9+4=12$
- b 5 è un numero primo
- c non è vero che 11 sia un numero primo
- d il Tevere bagna Roma
- e il rombo è un particolare parallelogramma

### 100 Negazione

□□□

Scrivi la negazione dei seguenti enunciati:

- a il signor Rossi è sposato
- b ieri ero a Roma
- c i Beatles si esibiscono a Milano
- d oggi ho studiato
- e non è vero che ieri non sono andato a scuola
- f tutti gli alunni sono presenti



# Training

## PROVE APERTE PER LA VERIFICA DELLE ABILITÀ

### 101 Enunciati derivati 1

■□□

Dati i seguenti enunciati:

$p$  = Gigi è partito per Milano,  $q$  = Gigi è studente

Scrivi gli enunciati:

not  $p$ , not  $q$ ,  $p$  AND  $q$ ,  $p$  OR  $q$ ,  $p$  AND NOT  $q$ ,  
NOT  $p$  AND  $q$ , NOT  $p$  OR NOT  $q$

### 102 Enunciati derivati 2

■□□

Con riferimento agli enunciati  $p$  e  $q$  dell'esercizio precedente, nella ipotesi che sia  $p$  vera,  $q$  falsa, stabilisci il valore di verità dei seguenti enunciati:

$p$  AND  $q$ ,  $p$  OR  $q$ ,  $p$  AND NOT  $q$ , NOT  $p$  AND  $q$ ,  
NOT  $p$  OR NOT  $q$

### 103 Enunciati derivati 3

■□□

Ripeti l'esercizio precedente nelle seguenti ipotesi:

- (a)  $p$  vera,  $q$  vera
- (b)  $p$  falsa,  $q$  vera
- (c)  $p$  falsa,  $q$  falsa

### 104 Enunciati semplici e composti 1

■□□

Individua gli enunciati semplici e i connettivi dei seguenti enunciati composti:

- (a) piove e non fa freddo
- (b) piove oppure la nebbia è molto fitta
- (c) Gigi e Daniele sono interrogati in matematica
- (d) Il numero 7 non è primo e non è pari

### 105 Enunciati semplici e composti 2

■□□

Scrivi sotto forma simbolica sia gli enunciati semplici che quelli composti dell'esercizio precedente e di ciascuno indica il valore di verità.

### 106 Enunciati semplici e composti 3

■□□

Considera i seguenti enunciati:

$p$  = Anna è amica di Lisa

$q$  = Lisa è amica di Anna

E scrivi in forma simbolica i seguenti enunciati composti:

- (a) Anna non è amica di Lisa
- (b) Lisa e Anna sono amiche
- (c) Lisa e Anna non sono amiche
- (d) Anna è amica di Lisa o Lisa è amica di Anna
- (e) Anna è amica di Lisa o Lisa non è amica di Anna
- (f) Anna non è amica di Lisa e Lisa è amica di Anna

### 107 Tavole di verità 1

■□□

Costruisci le tabelle di verità delle seguenti espressioni logiche:

- (a)  $d = \text{NOT} ((a \text{ OR } c) \text{ OR } b) \text{ OR } (a \text{ AND } c)$
- (b)  $d = \text{NOT} (a \text{ OR NOT } b) \text{ AND NOT } c$
- (c)  $c = (a \text{ OR } b) \text{ OR NOT } (a \text{ AND } b)$
- (d)  $d = (a \text{ AND } (b \text{ OR } c)) \text{ AND NOT } c$

### 108 Tavole di verità 2

■□□

Calcola le tabelle di verità delle seguenti espressioni booleane:

- (a)  $(a \text{ AND } b) \text{ XOR } c$
- (b)  $a \text{ AND } b \text{ OR } c$
- (c)  $a \text{ AND } (b \text{ OR } c)$
- (d)  $a \text{ OR } b \text{ AND } c$

### 109 Espressioni booleane

■□□

Supponendo che l'enunciato A sia falso, l'enunciato B sia vero e l'enunciato C sia vero, indica il risultato delle seguenti espressioni booleane:

- (a)  $(a \text{ XOR } b) \text{ AND } c$
- (b)  $a \text{ OR } b \text{ OR } c$
- (c)  $a \text{ AND } (b \text{ OR } c)$
- (d)  $a \text{ AND } b \text{ AND } c$



Con questa scheda puoi autovalutare il tuo livello di acquisizione delle conoscenze e delle abilità insegnate nell'Unità formativa. Attribuisce un punto a ogni risposta esatta. Se totalizzi un punteggio:

<4	Tra 4 e 6	Tra 6 e 8	>8
Rifletti un po' sulle tue "disgrazie"	Rivedi l'unità formativa nelle sue linee generali	Rivedi l'unità formativa nelle sue linee generali	Tutto OK
Rivedi con attenzione tutta l'unità formativa	Ripeti il questionario		
Ripeti il questionario			

1. La traduzione dell'algoritmo nel linguaggio adottato prende il nome di:

a) codifica  
b) programmazione  
c) traslazione  
d) implementazione

3. Effettua l'analisi dei dati del seguente problema completando l'apposita tabella. Devono essere ben definite le specifiche funzionali, le variabili e le eventuali costanti:

Realizzare un algoritmo che chieda l'età di uno studente. Sapendo che a 14 anni si può guidare il motorino da 50 cc, a 16 la moto fino a 125 cc, a 18 l'auto fino a 130 Km/h e a 21 le auto di tutti i tipi, visualizzare l'elenco dei mezzi che può guidare.

4. Risolvi il seguente esercizio:

Il comune di Lecce ha deciso di effettuare una serie di rilevazioni della concentrazione di polveri sottili in N giorni successivi. Durante l'analisi, ogniqualvolta il valore di concentrazione è maggiore di 50 si deve visualizzare un messaggio di allarme. Alla fine delle rilevazioni si deve visualizzare il valore massimo e il numero che identifica il giorno in cui è stato ottenuto, il valore minimo e il numero che identifica il giorno in cui è stato ottenuto e la variazione percentuale, cioè (massimo - minimo) / minimo.

5. Trasforma il seguente ciclo iterativo postcondizionale in uno precondizionale. Correda, inoltre, il frammento con le parti mancanti al fine di renderlo un algoritmo completo.

► RIPETI  
► SCRIVI("Inserisci tre numeri")  
► LEGGI(A, B, C)  
► SCRIVI(A\*4, B\*3, C\*2)  
► SCRIVI("Confermi?")  
► LEGGI(Risposta)  
► FINCHÉ(Risposta = 'S')

6. Quale output produce il seguente algoritmo?

```

► ALGORITMO Otto
► VARIABILI
► Somma, K : INTERO
► INIZIO
► Somma ← 0
► K ← 10
► MENTRE(K - (K DIV 2) > 0) ESEGUI
►   Somma ← Somma + K
►   K ← K - 2
► FINEMENTRE
► SCRIVI(Somma)
► FINE
  
```

a) 45    b) 40    c) 35    d) 30

7. Stabilisci se le seguenti affermazioni sono vere o false.

• Un ciclo precondizionale può sempre essere trasformato in un ciclo postcondizionale ☐ V ☐ F  
• Quando è noto il numero di ripetizioni si deve sempre usare un ciclo determinato ☐ V ☐ F

8. Stabilisci se le seguenti affermazioni sono vere o false.

• Un ciclo determinato può sempre essere sostituito da un ciclo con precondizionale ☐ V ☐ F  
• Un ciclo determinato può sempre essere sostituito da un ciclo postcondizionale ☐ V ☐ F

9. Stabilisci se le seguenti affermazioni sono vere o false.

• Un ciclo precondizionale non può mai essere sostituito da un ciclo determinato ☐ V ☐ F  
• Un ciclo postcondizionale non può mai essere sostituito da un ciclo determinato ☐ V ☐ F

10. Stabilisci se le seguenti affermazioni sono vere o false.

• Un ciclo postcondizionale termina sempre quando la condizione di uscita è vera ☐ V ☐ F  
• In determinati casi il ciclo può non riuscire a terminare causando un loop infinito ☐ V ☐ F