



Descrizioni rigorose

LEZIONE

7



Facciamo un esempio di strategia risolutiva. Possiamo prendere in considerazione qualsiasi problema, di tipo matematico o preso dalla vita di tutti i giorni, poiché le considerazioni che faremo hanno validità generale. Consideriamo, allora, il seguente problema: **Cucinare la pasta asciutta per la propria famiglia.**

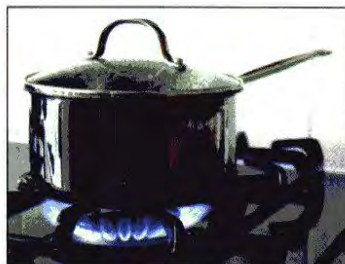
Come puoi immaginare, è ben diverso cucinare pasta asciutta per 2 persone o per 8. È evidente che manca un dato iniziale importantissimo: il numero di persone di cui è composta la nostra famiglia. Indichiamo genericamente con N questo dato iniziale. Quindi:

Dati iniziali

N: numero di persone di cui è composta la propria famiglia

Una prima strategia risolutiva potrebbe essere composta dai seguenti passi:

1. Inizio strategia risolutiva
2. Prendere una pentola adeguata
3. Aggiungere dell'acqua nella pentola
4. Accendere il fuoco e mettere la pentola sul fornello
5. Attendere che l'acqua bolla
6. Aggiungere il sale
7. Versare la pasta nella pentola
8. Attendere che sia cotta
9. Scolare la pasta togliendo l'acqua.
10. Fine strategia risolutiva



Questa strategia risolutiva a prima vista sembrerebbe sufficiente a descrivere la soluzione del nostro problema, ma osserviamola meglio!

Nel passo 2 si parla di "pentola adeguata". Ma che cosa significa adeguata? Ciò che può essere adeguato per una persona, potrebbe non esserlo per un'altra. Questo aggettivo risulta, pertanto, **ambiguo** e, quindi, non eseguibile dall'esecutore "stupido" da noi indicato. Ambigue sarebbero anche le espressioni: "una pentola sufficiente" oppure "una pentola giusta". È ambigua anche l'espressione: "una pentola per N persone". Come fa l'esecutore a stabilire quanto deve essere grande una pentola per N persone? Possiamo spiegarglielo trasformando il passo 2 nel modo seguente:

2. Prendere una pentola da $N * L$ litri dove L è il numero di litri a persona (per esempio $L = 0,5$, cioè mezzo litro a persona) e N è il numero delle persone.

L è, ora, un nuovo dato iniziale, che andiamo ad aggiungere all'insieme dei dati iniziali del nostro problema.

Dati iniziali

N: numero di persone di cui è composta la propria famiglia

L: litri di acqua per persona

Non sempre è immediatamente chiaro quali siano i dati iniziali di un problema, anzi, spesso si arriva a determinarli durante la stesura della strategia risolutiva. Riscriviamo, allora, anche il passo 3, che ora diventa:

3. Aggiungere $N * L$ litri di acqua nella pentola

I passi 2 e 3 così riscritti ci aiutano a formalizzare il problema e la strategia risolutiva e a dare rigore alle nostre descrizioni.

Osserviamo, ora, il passo 5. Tale passo non è sufficientemente **dettagliato**. Che cosa vuol dire per un esecutore (stupido) attendere che l'acqua bolla? Occorre essere più precisi e scrivere:

5. Attendere finché la temperatura dell'acqua (misurata con un termometro) raggiunge i 100 gradi centigradi.



Anche gli altri passi devono essere riscritti per **evitare ambiguità**: occorre essere precisi nella descrizione del da farsi. Una possibile riscrittura della strategia risolutiva, pertanto, può essere la seguente:

Strategia risolutiva

1. Inizio strategia risolutiva
2. Prendere una pentola da $N * L$ litri dove L è il numero di litri a persona (per esempio $L = 0,5$ mezzo litro a persona) e N è il numero delle persone
3. Aggiungere $N * L$ litri di acqua nella pentola
4. Accendere il fuoco e mettere la pentola sul fornello acceso
5. Attendere finché la temperatura dell'acqua (misurata con un termometro) raggiunge i 100 gradi centigradi
6. Aggiungere $N * S$ grammi di sale
7. Versare $N * G$ grammi di pasta nella pentola
8. Attendere T minuti
9. Scolare la pasta togliendo l'acqua.
10. Fine strategia risolutiva

dove:

S indica la quantità in grammi di sale per ogni persona;

T indica il tempo in minuti di cottura;

G indica i grammi di pasta per persona.

I dati iniziali sono ora i seguenti.

Dati iniziali

N: numero di persone

L: litri di acqua per persona

G: quantità di pasta per persona

S: grammi di sale per persona

T: tempo di cottura

La strategia risolutiva (descritta da un risolutore) è ora **rigorosa**, nel senso che non è ambigua, è più completa ed è sufficientemente dettagliata. Solo in questo modo l'**esecutore** può compiere le azioni necessarie a risolvere il problema. Il problema appena analizzato è ben formulato; infatti, per quanto detto:

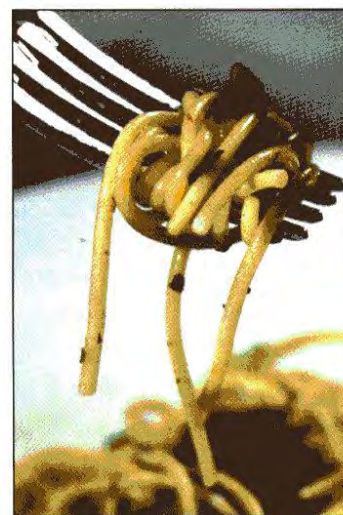
1. l'obiettivo finale è "cucinare la pasta asciutta" ed è, pertanto, esplicito;
2. il criterio di verifica consiste nel controllare che il risultato sia: "pasta asciutta correttamente cucinata: cioè al dente, non scotta, giustamente salata ecc.";
3. i dati iniziali indispensabili per proseguire nel processo risolutivo sono tutti presenti:

Dato iniziale	Significato
N	numero di persone
L	litri di acqua per persona
G	grammi di pasta per persona
S	grammi di sale per persona
T	tempo di cottura

L'assenza di uno solo di tali dati non consentirebbe all'esecutore di cucinare la pasta. Notiamo, infine, che la strategia risolutiva descrive come cucinare un qualsiasi tipo di pasta e non solo spaghetti o farfalline. Si dice che tale soluzione è *generale*.

Una strategia risolutiva è **generale** quando risolve problemi simili, cioè, problemi che fanno parte di una stessa famiglia di problemi detta **classe di problemi**.

Nel nostro esempio la classe di problemi è: "cucinare pasta asciutta". Se, per esempio, dobbiamo risolvere un'equazione di secondo grado, descriveremo come risolvere qualunque equazione di secondo grado e non solo una equazione specifica. Questo a condizione che ci forniscano, ogni volta, i coefficienti dell'equazione di cui vogliamo trovare le radici.





LEZIONE

**Trasporto della capra, del lupo e del cavolo da una sponda a un'altra del fiume**

Un contadino doveva trasportare al di là di un fiume il suo lupo, la sua capra e una cesta di cavoli, avendo a disposizione una barca poco capiente che avrebbe potuto trasportare solo lui in compagnia di una delle due bestie o lui insieme alla sola cesta di cavoli. Ma se avesse lasciato su una delle due rive del fiume il lupo insieme alla capra, questi l'avrebbe uccisa per mangiarsela; allo stesso modo non avrebbe potuto lasciare insieme capra e cavoli, perché la bestia li avrebbe sicuramente mangiati. La sua presenza era importante perché il lupo non nuocesse alla capra e la capra non toccasse i cavoli.

Strategia risolutiva

1. Inizio strategia risolutiva
2. Porta la capra sull'altra sponda
3. Torna indietro
4. Porta il cavolo sull'altra sponda
5. Porta la capra indietro
6. Porta il lupo sull'altra sponda
7. Torna indietro
8. Porta la capra sull'altra sponda
9. Fine strategia risolutiva

Calcolo del massimo comun divisore secondo il procedimento euclideo

Abbiamo imparato che il massimo comun divisore (MCD) di due numeri è il più grande divisore comune a entrambi (per esempio $\text{MCD}(18, 12) = 6$; $\text{MCD}(5, 10) = 5$; $\text{MCD}(4, 3) = 1$) e ci è stato insegnato anche a calcolarlo scomponendo in fattori i due numeri e prendendo i fattori comuni con esponente minore (per esempio, per calcolare $\text{MCD}(18, 12)$: $18 = 3^2 \cdot 2$ e $12 = 3 \cdot 2^2$ quindi $\text{MCD} = 2 \cdot 3 = 6$).

Il **metodo euclideo delle divisioni successive** serve per determinare il massimo comun divisore di due numeri naturali non nulli a e b .

Dati iniziali

a, b: numeri naturali non nulli di cui calcolare il massimo comun divisore

Dati finali

MCD: Massimo Comun Divisore dei numeri a e b

Illustriamo il funzionamento di questo metodo mediante due esempi.

Supponiamo di voler determinare il MCD di 2079 e 987.

Dividendo 2079 per 987 otteniamo come quoziente 2 e come resto 105.

Abbiamo, quindi: $2079 = 987 \cdot 2 + 105$.

Dividiamo ora 987 per 105. Si ha: $987 = 105 \cdot 9 + 42$.

Procediamo dividendo 105 per 42: $105 = 42 \cdot 2 + 21$.

Dividiamo ora 42 per 21: $42 = 21 \cdot 2 + 0$.

Siamo pervenuti a un resto nullo. L'ultimo divisore, in questo caso 21, è il MCD cercato. Quindi $\text{MCD}(2079, 987) = 21$.



Calcoliamo ora $\text{MCD}(2835, 1540)$:

dividiamo 2835 per 1540: $2835 = 1540 \cdot 1 + 1295$,

dividiamo 1540 per 1295: $1540 = 1295 \cdot 1 + 245$,

dividiamo 1295 per 245: $1295 = 245 \cdot 5 + 70$,

dividiamo 245 per 70: $245 = 70 \cdot 3 + 35$,

dividiamo 70 per 35: $70 = 35 \cdot 2 + 0$.

Il resto è 0, per cui $\text{MCD}(2835, 1540) = 35$.

Strategia risolutiva

1. Inizio strategia risolutiva
2. Siano x e y due numeri.
3. Calcola il resto della divisione di x e y .
4. Se il resto è diverso da zero, ricomincia dal passo 3 utilizzando come x il valore attuale di y e come y il valore del resto, altrimenti prosegui al passo successivo.
5. Il massimo comun divisore è uguale al valore attuale di y .
6. Fine strategia risolutiva

Calcolo della soluzione dell'equazione $ax + b = 0$

Riprendiamo il metodo già conosciuto per risolvere un'equazione di primo grado, o meglio un'equazione algebrica razionale intera di primo grado (a una incognita). Consideriamo l'equazione nella sua forma generale (ricorrendo, cioè, a dei coefficienti letterali):

$$ax + b = 0$$

Aggiungendo a entrambi i membri dell'uguaglianza la quantità $-b$ (regola del trasporto) otteniamo:

$$ax = -b$$

Infine, dividendo ambo i membri per a , otteniamo la soluzione:

$$x = -b/a$$

Naturalmente il coefficiente a dovrà essere diverso da zero, perché l'equazione sia determinata (in questo esercizio daremo per scontato che il coefficiente a sia diverso da zero poiché non conosciamo ancora uno strumento che ci consenta di effettuare dei controlli). Per esempio:

$$2x - 8 = 0$$

$$2x = 8 \text{ (regola del trasporto)}$$

$$x = 8/2 \text{ (dividendo ambo i membri per 2)}$$

La soluzione dell'equazione data è:

$$x = 8/2 = 4.$$

Strategia risolutiva

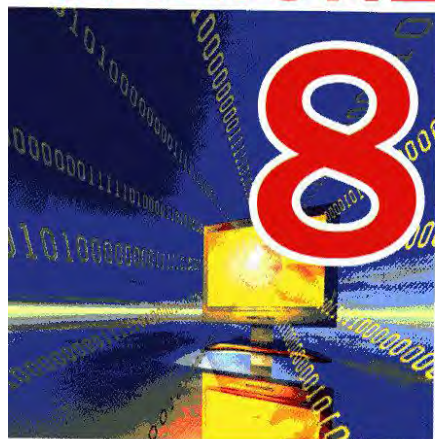
1. Inizio strategia risolutiva
2. Acquisisci i valori di a e b
3. Calcola $-b$
4. Dividi $-b$ per a e assegna il risultato a x
5. Visualizza il valore di x
6. Fine strategia risolutiva



L'algoritmo

LEZIONE

8



Sappiamo che una strategia risolutiva è caratterizzata da una serie di passi. Per essere più precisi, al termine *passo* dobbiamo sostituire quello di *azione*.

Si definisce **azione** un evento di cui sono noti il soggetto, detto *esecutore*, l'oggetto, o gli oggetti, su cui l'esecutore deve agire e la trasformazione prodotta su di essi in un'unità finita di tempo.

La frase *Daniela mangia una mela* puntualizza un'azione, in quanto, in un'unità finita di tempo, Daniela (esecutore) opera una trasformazione sulla mela (oggetto). Se poniamo l'accento sull'azione svolta e su come *Daniela mangia una mela*, risulta evidente che tale azione è composta da azioni più semplici. Nel dettaglio, infatti, l'azione in questione si può scomporre nel seguente modo:

- Daniela prende una mela;
- Daniela sbuccia la mela;
- Daniela taglia a fette la mela;
- Daniela mastica la mela.

Le azioni devono avvenire in **sequenza**, una dopo l'altra (non si ritiene quindi possibile che Daniela tagli a fette la mela e, contemporaneamente, la mastichi).

Un'azione si dice **elementare** quando non può essere scomposta in altre azioni più semplici. L'azione elementare, detta anche **istruzione**, è interpretabile in modo univoco dall'esecutore e direttamente eseguibile. Ciò significa che l'esecutore comprende in modo univoco che cosa deve fare e sa come farlo.

Prendiamo in esame il problema: *Prelevare una somma di denaro dal bancomat*.

Specifiche funzionali	
Dati iniziali	Dati finali
C: codice segreto I: importo da prelevare	Banconote

Per descrivere la strategia risolutiva possiamo utilizzare la seguente sequenza ordinata di azioni elementari:

- introdurre la carta nel lettore
- digitare il codice segreto: C
- digitare l'importo da prelevare: I
- ritirare la carta dal bancomat
- prelevare le banconote

Abbiamo ora tutti gli elementi per comprendere la seguente definizione.

Un procedimento risolutivo è un **algoritmo** quando, fissato l'insieme **finito** delle **azioni elementari univocamente interpretabili e definite**, è possibile descrivere passo per passo il procedimento che risolve un problema costruendo una **successione ordinata e finita** di istruzioni la cui esecuzione si arresta per fornire i **risultati** di un problema a partire da ogni valore assunto dai **dati iniziali**.

Potrà sembrare strano, ma esistono algoritmi che non si arrestano e sono pur sempre algoritmi. In questo corso ci occuperemo esclusivamente di algoritmi che si arrestano per fornire il risultato desiderato.

Affinché si possa parlare di algoritmo è necessario che vengano rispettate le caratteristiche fondamentali viste in precedenza. Un algoritmo deve essere:

- **finito**: la strategia risolutiva descritta dall'algoritmo deve essere composta da un numero finito di azioni elementari. L'algoritmo, inoltre, deve prevedere un solo inizio e una sola fine;
- **univoco** o **non ambiguo** o **preciso**: ogni azione deve essere definita nei suoi effetti rigorosamente e senza ambiguità per l'esecutore;
- **generale**: deve essere valido non solo per un particolare problema, ma per tutti i problemi di una stessa classe;
- **completo**: deve considerare tutti i casi possibili che si possono verificare durante l'esecuzione e, per ogni caso, indicare la soluzione da seguire;



- **osservabile nei risultati**: deve esserci riscontro oggettivo del risultato. Nell'esempio del problema "cucinare la pasta" ottengo della pasta da mangiare;
- **deterministico**: a ogni unità finita di tempo l'esecutore deve scegliere e compiere una e una sola azione. Si dice anche che, partendo dagli stessi dati iniziali, l'esecuzione dell'algoritmo deve fornire sempre gli stessi risultati finali.

Come abbiamo già visto, per la risoluzione di un problema è possibile trovare diverse strategie risolutive alle quali corrispondono altrettanti algoritmi. Il parametro che consente di preferire una strategia, e quindi un algoritmo, rispetto a un'altra è l'**efficienza**.

Un algoritmo si dice **efficiente** quando:

- è **corretto**, cioè produce il risultato atteso;
- è **veloce** in termini di tempo impiegato per produrre il risultato;
- è **parsimonioso** in termini di risorse allocate per produrre il risultato.

Il termine **algoritmo** deriva dal nome di un matematico arabo del IX secolo: **Abu Ja'far Mohammed ibn Musa al-Kowarizm** (in arabo il nome significa "padre di Ja'far, Maometto, figlio di Mosè, nativo di Kowarizm", e Kowarizm è il nome dell'odierna città russa Khiva). Nell'825 questo personaggio pubblicò due opere: una di aritmetica e una intitolata *Kitab Al-jabr Wal Muqabala*. Dalla parte centrale di quest'ultima ebbe origine la parola inglese **algebra**, mentre il termine **algoritmo** deriva da modifiche e adattamenti del nome del matematico.

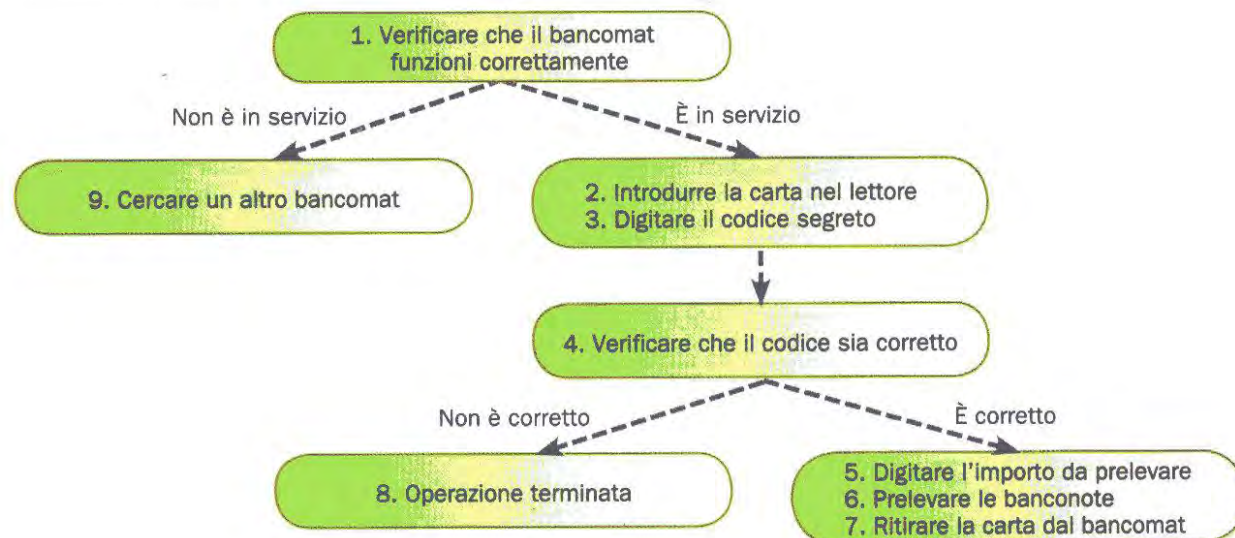
Leonardo da Pisa, un altro grandissimo matematico vissuto intorno agli inizi del XIII secolo e noto con il soprannome di Fibonacci, riferendosi alle scoperte del matematico arabo era solito semplificare il suo nome in: "Algoritmo diceva che...". Considerato che Abu Ja'far Mohammed ibn Musa al-Kowarizm si era sempre interessato allo studio dei procedimenti aritmetici di calcolo necessari per risolvere i problemi, il termine **algoritmo** venne utilizzato per indicare questo tipo di procedimento.



Ora riconsideriamo il problema (e l'algoritmo) del "prelievo dal bancomat". Le azioni descritte in quel semplice algoritmo sono perfettamente applicabili se:

- il bancomat funziona correttamente;
- il codice segreto viene inserito correttamente.

In altre parole, l'algoritmo che descrive la soluzione del problema non è completo! Trasformiamolo, allora, nel modo riportato di seguito. Per semplicità supponiamo che, in caso di inserimento errato del codice segreto, il bancomat non consenta di digitarlo di nuovo.



Questa versione ha una nuova veste grafica. Per facilitare la lettura abbiamo utilizzato le frecce per visualizzare i percorsi logici presenti nell'algoritmo e abbiamo raggruppato le azioni da eseguire al verificarsi di determinate situazioni.



Rappresentazione degli algoritmi

9

Per rappresentare un algoritmo possiamo ricorrere:

- al formalismo dei **diagrammi a blocchi** (detti anche *DaB* e *flow-chart*);
- al formalismo dello **pseudolinguaggio** (detto anche *linguaggio di progetto*).

I diagrammi a blocchi

Un diagramma a blocchi è una **descrizione grafica** dell'algoritmo, realizzata mediante appositi simboli, che mette in evidenza il flusso di esecuzione delle istruzioni. I simboli utilizzabili nei diagrammi a blocchi sono quelli riportati nella seguente tabella.

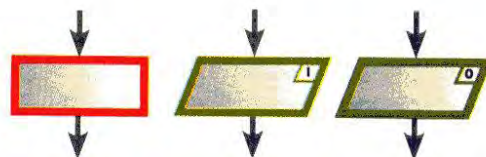
Il simbolo	indica nei diagrammi a blocchi
	l'inizio dell'algoritmo
	la fine dell'algoritmo
	un'azione o un blocco di azione
	l'ordine in cui eseguire i vari passi
	una condizione che può essere vera o falsa
	la possibilità di fornire informazioni (input) al computer, mediante la tastiera, durante l'esecuzione
	la possibilità di ricevere informazioni (output) dal computer, sullo schermo, durante l'esecuzione

Per usare correttamente questi simboli occorre tenere presente che:

- all'inizio si può seguire una sola direzione e alla fine si può giungere da un'unica strada;



- un blocco di azioni, di input o di output ha una sola freccia che vi arriva e una sola che parte da esso;



- solo il simbolo del rombo prevede due frecce in uscita, ma ne ha sempre solo una in entrata;



Lo pseudolinguaggio

Per indicare le azioni, negli algoritmi analizzati finora abbiamo utilizzato la lingua italiana, che è un **linguaggio naturale**. Questo tipo di linguaggio, però, pur essendo molto utile per chiarire le idee sul problema, non si presta bene a una precisa descrizione dell'algoritmo, poiché contiene sinonimi, ambiguità e, talvolta, anche eccezioni, metafore, allegorie.



Se dobbiamo affermare che uno studente è molto volenteroso, possiamo anche utilizzare gli aggettivi: zelante, operoso, attivo, alacre, sollecito, dinamico, solerte e così via. Inoltre, se proviamo a leggere la frase:

"Ieri sera ho visto Riccardo con un conoscente"

vediamo che può essere interpretata in due modi diversi. Vale a dire:

"Ieri sera ho visto Riccardo che era in compagnia di un conoscente"

oppure:

"Ieri sera ho visto Riccardo mentre ero in compagnia di un conoscente".

Un altro esempio di ambiguità sono le istruzioni:

prendere una pentola adeguata

aggiungere dell'acqua nella pentola

della strategia risolutiva del problema: *Cucinare la pasta asciutta per la propria famiglia.*



Lo pseudolinguaggio è un **linguaggio formale**, ossia un linguaggio che utilizza simboli ai quali corrisponde uno e un solo significato in qualsiasi contesto. La descrizione formale dell'algoritmo in pseudolinguaggio si dice **pseudocodice**. L'attività di scrittura dello pseudocodice prende il nome di **pseudocodifica**. Scopo principale della pseudocodifica è portare il risolutore a esprimere le proprie istruzioni in una forma naturale, utilizzando frasi ed espressioni elementari della lingua italiana. Ciò permette di concentrarsi sulla risoluzione logica del problema, invece che sulla forma e sui vincoli da rispettare nella sua enunciazione. Non esiste uno pseudolinguaggio standard e convenzionalmente usato: gli autori definiscono spesso un proprio pseudolinguaggio che utilizzano nelle loro pubblicazioni, inoltre ciascun programmatore può essere portato a utilizzare una propria variante. Ogni pseudolinguaggio ha un proprio lessico, una propria sintassi e una propria semantica, ma la progettazione di questo tipo di formalismo è volta alla comprensibilità e alla leggibilità del codice; la sintassi sarà quindi meno rigorosa rispetto a un vero linguaggio e le parole chiave saranno evocative, in modo da rendere più intuitiva la sua interpretazione.

Per usare correttamente lo pseudolinguaggio occorre rispettare le seguenti regole.

1. Le **parole chiave**, o parole riservate, sono scritte in maiuscolo (non possono essere usate come identificatori). Sono utilizzati verbi all'imperativo proprio per evidenziare l'ordine che il programmatore impone all'esecutore.
2. Le altre risorse usate nell'algoritmo sono scritte in maiuscolo se composte da una sola lettera (per esempio A, K), con l'iniziale maiuscola se composte da più caratteri (per esempio Somma, Totale, Contatore) e rigorosamente senza spazi.
3. Nell'illustrazione della sintassi di un'istruzione:
 - le parole racchiuse tra parentesi angolari < > rappresentano le **categorie sintattiche**, ossia elementi generali del linguaggio che devono essere ulteriormente specificati. Per esempio:
 - le parentesi quadre [] indicano l'**opzionalità**, ossia i blocchi in esse racchiusi possono anche non essere presenti; per esempio, la pseudoistruzione riportata qui a destra indica che dopo la parola chiave **SCRIVI** occorre aprire la parentesi tonda e inserire un messaggio; se lo si desidera si può anche inserire una variabile, purché sia separata dal messaggio con una virgola;
 - i blocchi separati dal simbolo | possono essere usati in **alternativa**. Per esempio, la pseudoistruzione riportata qui a destra indica che alla variabile posso assegnare un numero oppure una lettera;
 - le parentesi graffe { } indicano la **ripetizione**, ossia i blocchi in esse racchiusi possono essere ripetuti più volte; per esempio, la pseudoistruzione riportata qui a destra indica che dopo la parola chiave **LEGGI** occorre aprire la parentesi tonda e inserire il nome di una variabile; se dobbiamo inserirne delle altre occorrerà separarle con una virgola;
 - il simbolo // è utilizzato per inserire i commenti per le varie istruzioni, che saranno ignorati dall'esecutore. Un esempio è riportato qui a destra.

► **<Variabile> ← <Espressione>**

può essere: $\begin{cases} A \leftarrow B \\ A \leftarrow C + 2 * D \\ C \leftarrow 0 \end{cases}$

► **SCRIVI(<Messaggio>[, <Variabile>])**

► **<Variabile> ← <Numero> | <Lettera>**

► **LEGGI(<Variabile> {, <Variabile>})**

► **LEGGI(A) // Questo è un commento**

Lo pseudolinguaggio, quindi, viene usato per esprimere con chiarezza e semplicità la soluzione logica di un problema di elaborazione dati. Di per sé non può essere immesso direttamente in un calcolatore per essere eseguito, dovrà essere tradotto in un codice scritto in un **linguaggio di programmazione** che possa essere interpretato dal computer (per esempio il linguaggio C, il linguaggio Java e così via).



Variabili e costanti

LEZIONE

10

Consideriamo il seguente problema di tipo matematico:

Calcolare e visualizzare l'area della superficie di un triangolo

Procediamo alla sua risoluzione servendoci della descrizione della strategia risolutiva in italiano, corredandola dell'analisi e delle specifiche funzionali.

Analisi

L'area di un triangolo si ottiene moltiplicando la misura della base per quella dell'altezza e dividendo il prodotto ottenuto per due. Per risolvere il problema è necessario conoscere, pertanto, le due misure.

Specifiche funzionali

Dati iniziali	Dati finali
<i>Base</i> : misura della base del triangolo	<i>Area</i> : misura dell'area del triangolo
<i>Altezza</i> : misura dell'altezza del triangolo	

Strategia risolutiva

1. Inizio strategia risolutiva
2. Acquisisci in *Base* il valore della base del triangolo
3. Acquisisci in *Altezza* il valore dell'altezza del triangolo
4. Moltiplica il valore di *Base* per il valore di *Altezza* e dividi il risultato per due. Memorizza il risultato in *Area*
5. Comunica il valore di *Area*
6. Fine strategia risolutiva

Base, *Altezza* e *Area* individuano tre variabili.

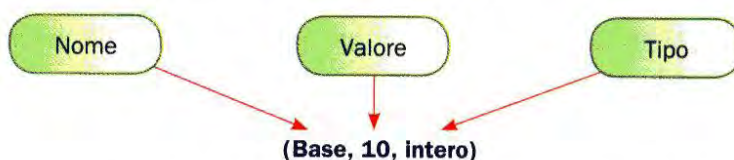
Una **variabile** è un'area di memoria RAM riservata per contenere un particolare dato che può essere modificato durante l'esecuzione del procedimento risolutivo.

La variabile costituisce, quindi, l'astrazione di una cella di memoria: tale cella contiene in ogni istante il valore della variabile.

La variabile è caratterizzata da:

- un **nome** che identifica univocamente la cella di memoria e, quindi, il dato in essa contenuto (per esempio: *Base*); è buona norma utilizzare nomi che consentano di richiamare immediatamente il significato della variabile;
- un **valore** che il dato può assumere (per esempio: 10);
- un **tipo**, cioè l'insieme di valori che il dato può assumere, per esempio *valori interi* (l'operazione che definisce il tipo di dato di una variabile prende il nome di **tipizzazione** o **dichiarazione**).

Nel nostro esempio la variabile *Base* è individuata dalla terna (*Base*, 10, intero), dove: *Base* è il nome, 10 è il valore, intero è il tipo.



Il concetto di variabile è molto importante in informatica. Possiamo pensare a una variabile come a un contenitore aperto, all'interno del quale è possibile inserire dei valori e sostituirli con altri dello stesso tipo.

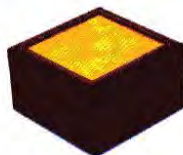
Penseremo alla **costante**, invece, come a un "oggetto" a cui è associato un identificatore e un valore che non può essere modificato, nel senso che potrà essere utilizzato ma non modificato, rimanendo così fisso per tutta l'esecuzione del procedimento risolutivo. Una costante ha, quindi, un valore statico predefinito. Un esempio di costante è il valore π .



Variabili e costanti sono normalmente rappresentate con scatole che simboleggiano le celle di memoria, infatti altro non sono che aree di memoria adibite a contenere temporaneamente (cioè per tutto il corso dell'elaborazione) determinati valori.



Variabile



Costante

Le variabili *Base*, *Altezza* e *Area* contengono i dati del nostro algoritmo.

Classificazioni dei dati

I dati possono essere classificati in diversi modi e, in base a come interagiscono con il computer, si distinguono come segue.

- **Dati di input**; sono quelli che vengono forniti dall'esterno e servono per la risoluzione del problema (per esempio il valore della base e dell'altezza).
- **Dati di output**; sono quelli che vengono comunicati all'esterno come soluzione del problema (per esempio il valore dell'area).
- **Dati di lavoro** (o **dati intermedi**); sono quelli che vengono utilizzati durante l'esecuzione del processo risolutivo.

In base agli oggetti che rappresentano e, quindi, al loro tipo, i dati si classificano come segue.

- **Numerici**; sono quelli che contengono numeri (la misura dell'altezza, dell'area, la costante 2, la costante 127.78) e che possono essere utilizzati per effettuare le operazioni matematiche. Sono, quindi, astrazioni dei corrispondenti insiemi di numeri della matematica. Questi dati possono essere ulteriormente distinti in **interi** e **reali**. Questa classificazione risulta utile per rappresentare in modo più dettagliato le informazioni alle quali si riferiscono.
- **Alfanumerici** (detti anche **stringhe**); sono dati che contengono caratteri (cifre, lettere dell'alfabeto, segni di interpunzione, caratteri speciali). Possono anche essere formati da sole cifre, ma su di essi non è possibile effettuare calcoli matematici (per esempio, che senso avrebbe sommare due numeri telefonici?). Esempi di dati alfanumerici sono, pertanto, il numero di codice fiscale, il nome di un cliente, il CAP, la parola "Appennino", il carattere 'A' e così via.

Lo pseudolinguaggio che utilizzeremo in questo corso prevede che le risorse utilizzate nell'algoritmo debbano essere scritte in maiuscolo se composte da una sola lettera (per esempio **A**, **K**), con l'iniziale maiuscola se composte da più caratteri (per esempio **Somma**, **Totale**, **Contatore**) e rigorosamente senza spazi. Questa regola si applica sia alle variabili, sia alle costanti. Per garantire una maggiore leggibilità e consentire una loro netta distinzione, scriveremo gli identificatori delle costanti interamente in maiuscolo e senza spazi. Pertanto:

- **PIGRECO**, **MASSIMO**, **MINIMO** sono tutti identificatori di costanti;
- **Somma**, **Media**, **NumeroMax** sono identificatori di variabili.

Rimane il dubbio sugli identificatori composti da una sola lettera che, secondo le regole stabilite, deve essere maiuscola. Al fine di evitare confusione, utilizzeremo nomi di una sola lettera solo per indicare variabili.





LEZIONE

11

Tipi di dati e astrazione: il tipo intero

I tipi di dati

Nella progettazione di un algoritmo devono essere affrontati i problemi relativi alla rappresentazione delle informazioni, che deve essere **efficiente** (senza sprechi inutili) ed **efficace** (non si deve perdere traccia di dati importanti). È importante, quindi, assegnare un corretto tipo di dato. Formalmente:

un **tipo di dato** è un'entità caratterizzata dai seguenti elementi:

- un insieme X di valori che rappresenta il **dominio** del tipo di dato;
- un insieme di **operazioni** su X .

Come è facile intuire, un algoritmo lavora sia con dati non numerici, cioè stringhe (ovvero sequenze di caratteri) sia numerici; questi ultimi, poi, si dividono ulteriormente in numeri interi, decimali, valute e così via. Questa distinzione è molto importante, perché ogni tipo di dato ha una dimensione (cioè un'occupazione in memoria) diversa: per esempio, un numero intero occupa meno memoria di un numero decimale a precisione doppia. Tali particolari possono sembrare sottigliezze, però quando si sviluppano applicazioni di una certa complessità assumono un'importanza rilevante.

I tipi di dati possono essere classificati in due grandi categorie:

- **tipi elementari** (o **tipi semplici**), sono i dati atomici visti sinora (interi, reali, carattere, booleani) e, come tali, non sono costituiti da altri dati;
- **tipi strutturati**, sono aggregazioni di altri dati (semplici o strutturati). Queste aggregazioni hanno una loro struttura, dalla quale è possibile estrarre i dati tramite appropriate operazioni.

Ogni linguaggio di programmazione consente di usare, in modo più o meno esplicito, un certo numero di tipi di dati predefiniti e fornisce di solito un certo insieme di strumenti per definire nuovi tipi di dati sulla base delle esigenze specifiche di un programma; questi ultimi sono conosciuti come **tipi di dati astratti** o **ADT** (*Abstract Data Type*). In questa e nella successiva lezione ci occuperemo di quelli più usati.

Il tipo intero

È un tipo base che intuitivamente potremmo assimilare all'insieme dei numeri interi relativi. Tuttavia non è proprio così, in quanto un calcolatore può rappresentare solo un sottoinsieme finito di questo insieme. Il dominio del tipo **intero** è pertanto un sottoinsieme finito dell'insieme Z dei numeri relativi. I suoi elementi (le **costanti**) sono compresi nell'intervallo:

$- \text{MassimoIntero}, + \text{MassimoIntero} - 1$

Il valore della costante *MassimoIntero* dipende dal numero di bit che vengono utilizzati per rappresentare il numero. Nella maggioranza dei linguaggi, un dato di tipo intero occupa due o quattro byte in memoria, cioè 16 o 32 bit. Nelle più comuni rappresentazioni, il valore del bit più significativo indica il segno (0 positivo, 1 negativo). Il massimo numero rappresentabile con 16 bit è:

Segno

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

che, convertito in decimale, rappresenta il valore 65536.

Generalmente l'intervallo di definizione degli interi è $(-32768, +32767)$. Come mai l'estremo inferiore ha un'unità in più? Abbiamo detto che, dei 16 bit occupati da un dato di tipo intero, 15 sono riservati al valore e uno al segno. Con 15 bit possiamo rappresentare tutti i numeri compresi tra 1 e 2^{15} estremi inclusi, oppure tra 0 e $2^{15}-1$, sempre con gli estremi inclusi. Considerando il bit del segno, possiamo affermare che con 15 bit è possibile rappresentare **2^{15} numeri positivi** e **2^{15} numeri negativi** a due a due uguali ma con segno opposto. Così facendo, anche il numero zero avrà due rappresentazioni: $+0$ e -0 .



Questo tipo di rappresentazione, denominato **rappresentazione in modulo e segno**, ha, quindi, lo svantaggio di fornire due differenti rappresentazioni per il valore zero e inoltre non rende vera l'uguaglianza $X + (-X) = 0$. Per questo motivo, per rappresentare i numeri negativi si ricorre al criterio del **complemento a 2** e quindi i numeri negativi rappresentabili in due byte vanno da -1 a -2^{15} .

In conclusione, se N è il numero di bit a disposizione per la rappresentazione di un tipo intero, utilizzando la rappresentazione in modulo e segno per i numeri positivi e quella del complemento a 2 per i numeri negativi è possibile rappresentare il seguente insieme di valori:

$$[-2^{(N-1)} \dots 2^{(N-1)} - 1]$$

Esaminiamo ora gli **operatori aritmetici associati al tipo intero**, ossia gli operatori che agiscono su operandi interi e forniscono risultati interi:

Operatore	Esempi	
$+$ per l'addizione	$30 + 14 = 44$	$-5 + 2 = -3$
$-$ per la sottrazione	$28 - 18 = 10$	$-4 - 30 = -34$
$*$ per la moltiplicazione	$2 * 8 = 16$	$-6 * 3 = -18$
DIV per la divisione intera. Se viene applicato a numeri interi dello stesso segno fornisce un risultato positivo, altrimenti negativo. Se il secondo operando è 0 viene segnalato un errore.	$(-20) \text{ DIV } (-3) = -6$ $4 \text{ DIV } (-3) = -1$ $5 \text{ DIV } 0 = \text{ERRORE}$	$(-5) \text{ DIV } 2 = -1$ $8 \text{ DIV } 5 = 1$
MOD per il <i>modulo</i> (resto della divisione tra gli operandi indicati). Si applica solo con il secondo operando positivo e fornisce sempre risultati positivi, altrimenti segnala un errore.	$24 \text{ MOD } 4 = 0$ $20 \text{ MOD } -2 = \text{ERRORE}$	$15 \text{ MOD } 6 = 3$ $-15 \text{ MOD } 2 = 1$

Su tutti gli operandi numerici è possibile applicare anche gli **operatori relazionali** ($<$, $>$, \leq , \geq , \neq). Se si lavora con espressioni aritmetiche, queste devono essere composte esclusivamente da operandi e operatori interi. I risultati devono sempre appartenere allo stesso insieme. Durante il calcolo il computer seguirà le regole dell'**algebra**, rispettando priorità e precedenze di operazioni e operandi. È molto importante definire le **espressioni intere**, in quanto talvolta può accadere che alcuni risultati parziali non risultino compresi nell'intervallo stabilito. In questi casi il computer può interrompere l'esecuzione, oppure assegnare a questo risultato parziale un valore non controllabile, falsando così il risultato finale.

Se il computer deve eseguire $32767 + 3$, può anche non interrompere l'esecuzione (segnalando un messaggio di errore), ma eseguire l'addizione fornendo come risultato -32766 . Com'è possibile? Per comprendere questo metodo, immaginiamo un orologio i cui numeri fanno parte di un insieme numerico finito.



Analizziamo l'**aritmetica dell'orologio**:

$2 + 5 = 7$
 $8 + 4 = 12 = 0$
 $10 + 5 = 15$ che sull'orologio ha valore 3
 $8 + 9 = 17$ che sull'orologio ha valore 5

Risultati di questo tipo sono possibili solo in insiemi finiti e l'aritmetica di questo tipo è detta **aritmetica modulare**: pertanto l'aritmetica dell'orologio è modulo 12. Analogo ragionamento vale per il nostro tipo intero: l'aritmetica dell'insieme intero è modulo 65536 (65536 è il numero massimo rappresentabile in 16 bit, cioè la somma di $32768 + 32768$) e presenta una struttura ciclica identica a quella dell'orologio, solo un po' più estesa.

L'errore dovuto al superamento degli estremi dell'intervallo prende il nome di:

- **errore di overflow** (quando il risultato è troppo grande, ossia quando si supera l'estremo superiore);
- **errore di underflow** (quando il risultato è troppo piccolo, ossia si va oltre l'estremo inferiore).

Tali errori si verificano per un superamento della capacità della memoria di ricevere un valore che supera i limiti previsti. Gli **errori di overflow** e **di underflow**, insieme a quelli di **divisione per 0** e **modulo negativo**, sono segnalati in fase di esecuzione del programma, provocandone l'arresto; per questo motivo vengono detti errori a **run-time** (o errori a **tempo di esecuzione**).



LEZIONE

12

Tipi di dati e astrazione: reale, carattere, stringa, booleano

Il tipo reale

È un tipo base che potremmo intuitivamente associare ai numeri reali. Sappiamo però, da quanto già visto per il tipo intero, che il calcolatore non può rappresentare un insieme infinito di numeri, e quindi ciò vale anche per i numeri reali. Inoltre, in questo caso c'è un'ulteriore limitazione in quanto il numero reale della matematica richiede in taluni casi un numero infinito di cifre per la sua rappresentazione, quindi anche nella rappresentazione del singolo numero reale abbiamo una restrizione, che si traduce in un'approssimazione del numero reale. I valori di tipo reale sono anche detti numeri in virgola mobile o con rappresentazione a mantissa e esponente. Ciò corrisponde al fatto che il calcolatore, per ogni numero reale, rappresenta esattamente una coppia di numeri (m,e) dove il primo è la mantissa e il secondo l'esponente.

Il dominio del tipo reale è pertanto un sottoinsieme finito dell'insieme \mathbb{R} dei numeri reali. I suoi elementi sono quei numeri decimali relativi compresi tra:

$$\pm \text{MinimoReale} \times 10^{-\text{EsponenteNegativo}} \dots \pm \text{MassimoReale} \times 10^{+\text{EsponentePositivo}}$$

in cui sia i valori assunti dalle costanti *MinimoReale*, *MassimoReale*, *EsponentePositivo* ed *EsponenteNegativo*, sia il numero di cifre significative (generalmente compreso tra 7 e 16) dipendono dal numero di bit impiegati per la rappresentazione della mantissa e dell'esponente.

Gli operatori aritmetici associati a questo tipo sono i seguenti:

Operatore	Esempi
+ per l'addizione	$4,3 + 3,1 = 7,4$ $8,45 + (-6,8) = 1,65$
- per la sottrazione	$-18,6 - 3,4 = -22,0$ $124,72 - 0,039 = 124,681$
* per la moltiplicazione	$9,6 * 4,2 = 40,32$ $5,12 * (-6,4) = -32,768$
/ per la divisione	$15,3 / 2,1 = 7,285714286$ $18 / 4 = 4,5$

Il tipo carattere e il tipo stringa

Sul computer è possibile rappresentare anche caratteri non numerici (per esempio una lettera dell'alfabeto, un segno di operazione, un segno di interpunzione e così via). Il dominio del tipo *carattere* è l'insieme di tutti i caratteri disponibili e riproducibili sul computer dove il linguaggio è implementato, e cioè:

- le 26 lettere maiuscole;
- le 26 lettere minuscole;
- le 10 cifre numeriche;
- lo spazio o *blank* (b);
- i segni d'interpunzione;
- molti altri simboli non presenti come tasti sulla tastiera del computer, ma ugualmente rappresentabili. Questi simboli possono essere ottenuti tenendo premuto il tasto Alt e digitando, sulla tastierina numerica, il numero di **codice ASCII** corrispondente. Ciò, ovviamente, è valido per i sistemi che utilizzano il codice ASCII. Tale codice contiene la codifica dei caratteri rappresentabili, ordinati in modo da consentire di effettuare dei confronti anche tra caratteri.

Risulta vero che il dato di tipo carattere 'A' è minore del dato di tipo carattere 'a', oppure che 'a' < 'b' in quanto il numero di codice ASCII di 'A' (65) è minore del numero di codice ASCII di 'a' (97) e, analogamente, il numero di codice ASCII di 'a' è minore di quello di 'b' (98).



Byte	Cod	Car	Byte	Cod	Car	Byte	Cod	Car	Byte	Cod	Car	Byte	Cod	Car
00000000	0	NUL	00011010	26	Substitution	00110100	52	4	01001110	78	N	01101000	104	h
00000001	1	Start of heading	00011011	27	Escape	00110101	53	5	01001111	79	O	01101001	105	i
00000010	2	Start of text	00011100	28	File separator	00110110	54	6	01010000	80	P	01101010	106	j
00000011	3	End of text	00011101	29	Group separator	00110111	55	7	01010001	81	Q	01101011	107	k
00000100	4	End of transmit	00011110	30	Record separator	00111000	56	8	01010010	82	R	01101100	108	l
00000101	5	Enquiry	00011111	31	Unit separator	00111001	57	9	01010011	83	S	01101101	109	m
00000110	6	Acknowledge	00100000	32	Spc	00111010	58	:	01010100	84	T	01101110	110	n
00000111	7	Audible bell	00100001	33	!	00111011	59	:	01010101	85	U	01101111	111	o
00001000	8	Backspace	00100010	34	"	00111100	60	<	01010110	86	V	01100000	112	p
00001001	9	Horizontal tab	00100011	35	#	00111101	61	=	01010111	87	W	01100001	113	q
00001010	10	Line feed	00100100	36	\$	00111110	62	>	01011000	88	X	01100010	114	r
00001011	11	Vertical tab	00100101	37	%	00111111	63	?	01011001	89	Y	01100011	115	s
00001100	12	Form feed	00100110	38	&	01000000	64	@	01011010	90	Z	01101000	116	t
00001101	13	Carriage return	00100111	39	'	01000001	65	A	01011011	91	[01101001	117	u
00001110	14	Shift out	00101000	40	(01000010	66	B	01011100	92	\	01101010	118	v
00001111	15	Shift in	00101001	41)	01000011	67	C	01011101	93]	01101011	119	w
00010000	16	Data link escape	00101010	42	*	01000100	68	D	01011110	94	^	01111000	120	x
00010001	17	Device control 1	00101011	43	+	01000101	69	E	01011111	95	_	01111001	121	y
00010010	18	Device control 2	00101100	44	,	01000110	70	F	01100000	96	`	01111010	122	z
00010011	19	Device control 3	00101101	45	-	01000111	71	G	01100001	97	a	01111011	123	{
00010100	20	Device control 4	00101110	46	.	01001000	72	H	01100010	98	b	01111100	124	
00010101	21	Neg. acknowledge	00101111	47	/	01001001	73	I	01100011	99	c	01111101	125	}
00010110	22	Synchronous idle	00110000	48	0	01001010	74	J	01100100	100	d	01111110	126	~
00010111	23	End trans. block	00110001	49	1	01001011	75	K	01100101	101	e	01111111	127	DEL
00011000	24	Cancel	00110010	50	2	01001100	76	L	01100110	102	f			
00011001	25	End of medium	00110011	51	3	01001101	77	M	01100111	103	g			

In generale, sul tipo carattere è imposto il seguente ordine:

a < b < c < ... < z
A < B < C < ... < Z
0 < 1 < 2 < ... < 9

Racchiuderemo i dati di tipo carattere tra apici. Questo è indispensabile per tenere distinte le cifre dal loro eventuale valore: nel tipo carattere, per esempio, le cifre da '0' a '9' non hanno alcun legame con il rispettivo valore numerico. Pertanto, 2 è diverso da '2', 9 è diverso da '9'.

Gli operatori definiti sul tipo carattere sono i classici operatori relazionali e forniscono, ovviamente, risultati di tipo booleano.

Una **sequenza di caratteri** prende il nome di **stringa** e, a differenza dei dati di tipo carattere, la racchiuderemo tra virgolette. Pertanto "Il mio nome è Piero" oppure "Come ti chiami?" o, ancora, "Ciao" sono tutti esempi di stringhe.

Il tipo booleano

Il dato di tipo **booleano** riveste un ruolo fondamentale nelle situazioni in cui si operano delle scelte in base al verificarsi di determinate condizioni.

Tale tipo di dato rappresenta il dominio composto da due soli valori o, più precisamente, da due costanti logiche (**valori di verità**) dette **Vero** e **Falso**. Questi due valori sono ordinati all'interno dell'insieme in modo che Falso < Vero.

Gli operatori che possono essere applicati a questi dati sono quelli booleani:

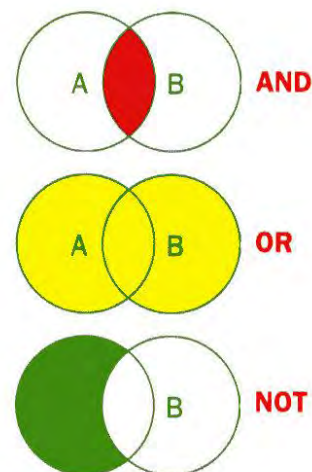
AND congiunzione logica
OR disgiunzione logica
NOT negazione logica

e forniscono risultati booleani. In tale contesto si adotta la classica convenzione che attribuisce all'operatore NOT la precedenza sull'operatore AND e all'operatore AND la precedenza sull'operatore OR.

Tipi ordinati

In base a quanto è stato detto in questa e nella precedente lezione, i tipi intero, reale, carattere sono tipi ordinati. Su un tipo ordinato sono definiti i seguenti operatori di relazione:

= uguale
≠ diverso
< minore
<= minore o uguale
> maggiore
>= maggiore o uguale





LEZIONE

13

Espressioni e loro valutazione

La valutazione delle espressioni

Esaminiamo il passo:

Moltiplica il valore dell'area della base (*AreaBase*) per il valore di *Altezza* e dividi il risultato per tre

per calcolare il volume di una piramide, e consideriamo l'equivalente espressione aritmetica:

$$\text{AreaBase} * \text{Altezza} / 3$$

In questa espressione:

- *AreaBase*, *Altezza* e 3 sono **operandi**;
- * e / sono **operatori**.

Quando un esecutore considera un'espressione che contiene più operatori, per calcolarne il valore, cioè per poter stabilire, per esempio, che l'espressione vale 10, deve stabilire l'ordine di esecuzione dei vari operatori e poi eseguire i calcoli veri e propri. Questo processo prende il nome di *processo di valutazione* e l'esecutore prende il nome di *valutatore*.

Nel **processo di valutazione** si stabilisce dapprima l'ordine di esecuzione dei vari operatori, cioè in quale ordine gli operatori presenti devono essere applicati agli operandi dell'espressione, e poi si eseguono le operazioni.

Si parla di **precedenza degli operatori** intendendo che ogni operatore ha la sua **priorità** e, quindi, può avere o meno precedenza rispetto a un altro operatore. Per calcolare il valore della nostra espressione potremmo prima moltiplicare il valore di *AreaBase* per quello di *Altezza* (cioè associare * ad *AreaBase* e ad *Altezza*) e poi dividere il prodotto per 3. È come se utilizzassimo le parentesi tonde nel seguente modo:

$$(\text{AreaBase} * \text{Altezza}) / 3$$

In alternativa, potremmo dividere il valore di *Altezza* per 3 (cioè associare / ad *Altezza* e al 3) e poi moltiplicare il tutto per il valore di *AreaBase*. È come se utilizzassimo le parentesi tonde nel seguente modo:

$$\text{AreaBase} * (\text{Altezza} / 3)$$

Il risultato in matematica non cambierebbe, perché gli operatori * e / hanno la **stessa priorità** e quindi è possibile associare gli operandi indifferente a destra o a sinistra. Questo invece non è vero in informatica, per i motivi precedentemente esposti di finitezza delle rappresentazioni. Per esempio, se *Altezza* = 1 e *AreaBase* = 6 avremo che *AreaBase* * (*Altezza* / 3) = 6 * 0,333 = 1,999998, mentre (*AreaBase* * *Altezza*) / 3 = 6 / 3 = 2. Quindi:

$$(\text{AreaBase} * \text{Altezza}) / 3 \neq \text{AreaBase} * (\text{Altezza} / 3)$$

Consideriamo, ora, l'espressione:

$$\text{Base} + \text{Altezza} / 2$$

Poiché gli operatori + e / hanno **differenti priorità** (/ ha priorità maggiore, cioè viene associato per primo), vi è differenza se associamo prima l'operatore * alla *Base* e all'*Altezza* e poi dividiamo per 2, oppure se prima associamo l'operatore / all'*Altezza* e al 2 e poi aggiungiamo la *Base*. Pertanto:

$$(\text{Base} + \text{Altezza}) / 2 \neq \text{Base} + (\text{Altezza} / 2)$$

In generale, quindi, nella valutazione di un'espressione, nota che sia la *precedenza* degli operatori e utilizzando la proprietà associativa, un esecutore elimina l'*ambiguità di esecuzione* e può ottenere il risultato dell'espressione.

Inserendo opportunamente le parentesi, forziamo l'esecutore a valutare l'espressione esattamente come desideriamo.

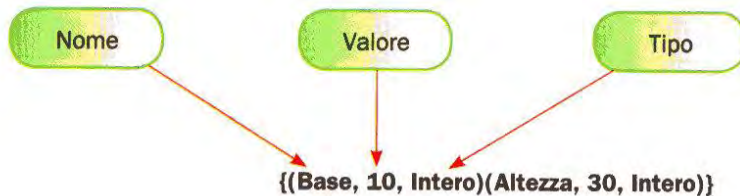


L'ambiente di valutazione delle espressioni

Prendiamo in esame l'espressione:

$$(Base * Altezza) / 2$$

Sappiamo che l'ordine di valutazione è stabilito dalla priorità degli operatori $*$ e $/$ che, in questo caso, è la stessa. Per calcolare il risultato, occorre attribuire i valori alle variabili in gioco. Supponiamo che la variabile *Base* abbia un valore *intero* pari a 10 e la variabile *Altezza* un valore *intero* pari a 30. Formalmente, potremmo definire un insieme così strutturato:



Il valore dell'espressione è, pertanto, 150 (quindi di tipo intero). Alla luce di queste considerazioni possiamo affermare che:

L'insieme delle terne:

$\{(NomeVariabile1, Valore1, Tipo1), (NomeVariabile2, Valore2, Tipo2), \dots, (NomeVariabileN, ValoreN, TipoN)\}$

utilizzato per valutare un'espressione costituisce il suo **ambiente di valutazione**.

Nella valutazione di una generica espressione, occorre, quindi, tenere conto del suo ambiente di valutazione.

Consideriamo l'espressione:

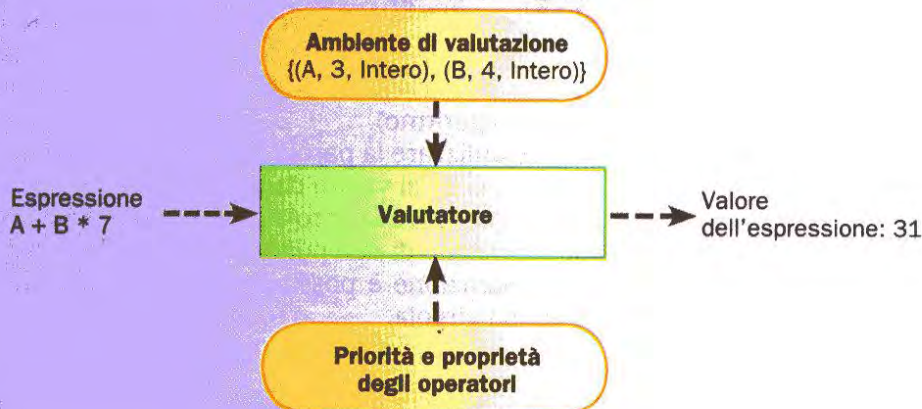
$$A + B * 7$$

In base all'ordine stabilito dalla priorità degli operatori viene valutata come se fosse:

$$A + (B * 7)$$

e vale:

- 31 se valutata nell'ambiente: $\{(A, 3, Intero), (B, 4, Intero)\}$
- 19 se valutata nell'ambiente: $\{(A, 5, Intero), (B, 2, Intero)\}$
- 8 se valutata nell'ambiente: $\{(A, 1, Intero), (B, 1, Intero)\}$





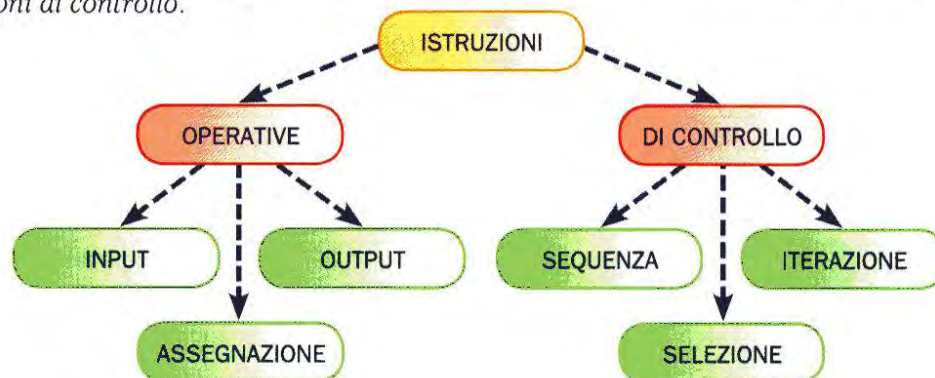
LEZIONE

14

Le istruzioni operative

Tipi di istruzioni

Le istruzioni presenti in un algoritmo possono essere classificate in base al loro comportamento. In generale possiamo suddividerle in *istruzioni operative* e *istruzioni di controllo*.



Le **istruzioni operative** sono quelle che corrispondono ad azioni direttamente eseguibili dall'esecutore e servono per acquisire i dati iniziali, effettuare le elaborazioni e comunicare i risultati finali. Si classificano in istruzioni di **assegnazione**, di **input** e di **output**.

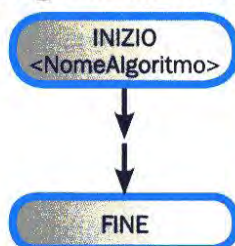
Le **istruzioni di controllo** consentono di scegliere percorsi differenti durante l'esecuzione, in funzione del verificarsi o meno di determinate condizioni. Si suddividono in istruzioni di **sequenza**, **selezione** e **iterazione**.

Iniziamo a occuparci delle istruzioni operative; per ognuna di esse, oltre alla specifica dell'attività eseguita, riportiamo la formalizzazione secondo il simbolismo dei diagrammi a blocchi e dello pseudolinguaggio.

Le istruzioni di inizio e fine

Abbiamo visto che un algoritmo deve prevedere un solo inizio e una sola fine per indicare, rispettivamente, quale istruzione dell'algoritmo debba essere eseguita inizialmente e quale determini la fine dell'esecuzione.

Diagramma a blocchi



Pseudolinguaggio

```

ALGORITMO <NomeAlgoritmo>
INIZIO

FINE
  
```

La dichiarazione degli identificatori

Tutti gli identificatori di costanti e di variabili presenti in un algoritmo devono essere dichiarati prima di essere utilizzati, specificandone il tipo. Questa istruzione crea la costante o la variabile allocando un'apposita area di memoria. L'istruzione di dichiarazione viene utilizzata solo quando ci si avvale del formalismo dello pseudolinguaggio, mentre viene intenzionalmente esclusa nella formalizzazione con diagrammi a blocchi (perché la dichiarazione degli identificatori avviene in una fase che precede la stesura dell'algoritmo).

Per dichiarare una costante occorre utilizzare la parola chiave **COSTANTI** seguita dal nome della costante, dal simbolo \leftarrow e dal valore da assegnare.

Per dichiarare una variabile occorre utilizzare la parola chiave **VARIABILI** seguita dall'identificatore della variabile, dal segno di due punti e dal tipo di appartenenza. In una stessa istruzione di dichiarazione è possibile dichiarare più variabili dello stesso tipo separandole con una virgola.

Costante

COSTANTI <Costante> \leftarrow <Valore>

Variabile

VARIABILI <Variabile> {,<Variabile>} : <Tipo>



L'istruzione di assegnazione

Per attribuire un valore a una variabile utilizziamo l'**istruzione di assegnazione** (o assegnamento), caratterizzata dall'operatore binario identificato dal simbolo \leftarrow . L'istruzione di assegnazione ha due termini, uno a sinistra e uno a destra del simbolo \leftarrow . Il termine di sinistra (*LValue*) è sempre il nome di una variabile, mentre quello di destra (*RValue*) può essere una costante, una variabile o, più in generale, un'espressione.

Diagramma a blocchi



Pseudolinguaggio

`<Variabile> ← <Espressione>`

L'assegnazione viene eseguita se il risultato della valutazione di `< Espressione >`, posta a destra dell'operatore \leftarrow , appartiene all'insieme di definizione della variabile `<Variabile>` posta a sinistra di \leftarrow .

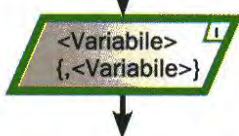
Quando si dichiara una variabile viene riservato dello spazio in memoria per potervi memorizzare un valore (numerico, carattere, stringa e così via). Di norma questo spazio in memoria viene fornito così com'è, senza "pulizia". Quindi, se abbiamo dichiarato una variabile intera, non è detto che questa contenga per forza il valore zero, anche se può capitare. È utile, perciò, dopo aver dichiarato una variabile, assegnarle un valore iniziale; tale operazione prende il nome di **inizializzazione**. Inizializzare significa proprio "assegnare il primo valore" a una variabile non ancora usata. Dopo l'inizializzazione, in qualunque punto dell'algoritmo potremo dire con certezza quanto vale la variabile.



L'istruzione operativa di input

L'istruzione di input è un particolare tipo di istruzione di assegnazione: consente di assegnare a una variabile un valore fornito dall'esterno, modificando, di conseguenza, l'**ambiente di valutazione** della variabile.

Diagramma a blocchi



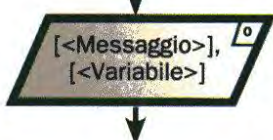
Pseudolinguaggio

`LEGGI(<Variabile> {,<Variabile>})`

L'istruzione operativa di output

L'istruzione di output consente di visualizzare il valore di una variabile, o di un'espressione, sul video o sulla stampante. È altresì utilizzata per visualizzare dei messaggi a video. L'istruzione di output non modifica in nessun modo l'ambiente di valutazione delle variabili.

Diagramma a blocchi



Pseudolinguaggio

`SCRIVI([<Messaggio>], [<Variabile>])`



LEZIONE

14



Riprendiamo l'analogia scatola-variabile e riportiamo alcuni esempi utili per comprendere l'istruzione di assegnazione.

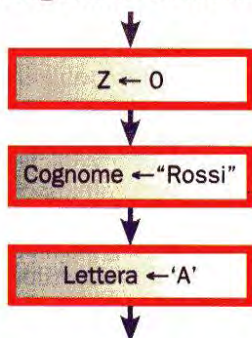
<Variabile> ← <Costante>

Assegniamo alla variabile *Z* il valore 0, alla variabile *Cognome* il valore *Rossi* e alla variabile *Lettera* il valore *A*.



La rappresentazione dell'assegnazione secondo il formalismo del diagramma a blocchi e dello pseudolinguaggio è la seguente:

Diagramma a blocchi



Pseudolinguaggio

VARIABILI
Z: **INTERO**
Cognome: **STRINGA[20]**
Lettera: **CARATTERE**
 ...
Z ← 0
Cognome ← "Rossi"
Lettera ← 'A'

Da notare che il valore stringa (cioè *Rossi*) deve essere racchiuso tra virgolette, mentre il valore carattere (cioè *A*) va racchiuso tra apici singoli. Inoltre, quando si dichiara una variabile di tipo *stringa*, è necessario specificare tra parentesi quadre qual è la dimensione massima consentita (nel nostro caso 20 caratteri).

<Variabile> ← <Variabile>

Assegniamo alla variabile *K* il valore contenuto nella variabile *A* nell'ambiente $\{(K, 5, \text{intero}), (A, 8, \text{intero})\}$

Vediamo graficamente le condizioni delle variabili prima e dopo l'assegnazione.

Prima



Dopo

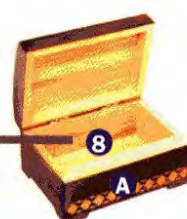
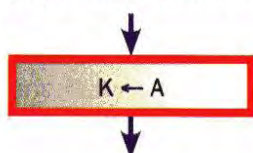


Diagramma a blocchi



Pseudolinguaggio

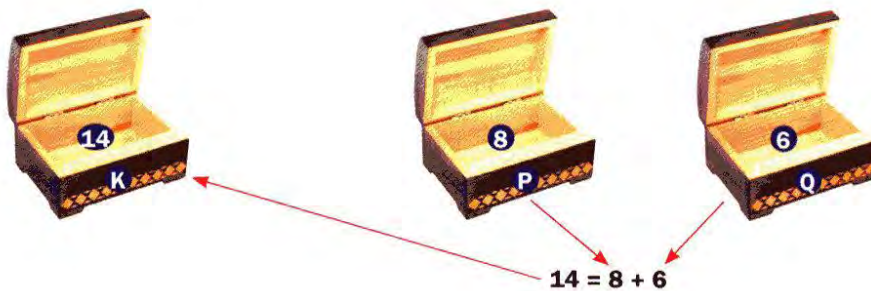
VARIABILI
K, A: **INTERO**
 ...
K ← *A*

La "scrittura" di una variabile, cioè l'assegnazione di un valore alla variabile stessa, rappresenta un'operazione distruttiva: nel nostro caso, il vecchio valore della variabile *K*, cioè 5, viene irrimediabilmente perso e sostituito con il nuovo valore, cioè 8, ceduto dalla variabile *A*. La "lettura" di una variabile, cioè il processo di valutazione, non è distruttiva: la variabile, nel nostro caso *A*, continua a mantenere lo stesso valore, cioè 8.

<Variabile> ← <Espressione>

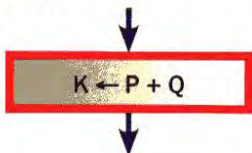


Assegniamo alla variabile K il risultato della somma dei valori contenuti nelle variabili P e Q nell'ambiente $\{(K, 0, \text{intero}), (P, 8, \text{intero}), (Q, 6, \text{intero})\}$. Il processo di assegnazione è illustrato nella seguente figura.



Formalizzando, abbiamo:

Diagramma a blocchi



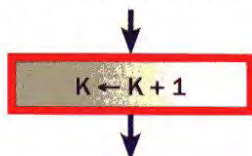
Pseudolinguaggio

VARIABILI
 $K, P, Q: \text{INTERO}$
 ...
 $K \leftarrow P + Q$

Per eseguire l'istruzione si valuta l'espressione $P + Q$, recuperando i valori presenti in P e Q e facendone la somma. Si pone il risultato nella variabile K dopo avere eliminato il valore precedentemente presente.

Dato che la valutazione di un'istruzione di assegnazione avviene "da destra verso sinistra", è possibile utilizzare **istruzioni che assegnano una variabile a se stessa**. Diamo uno sguardo alle seguenti rappresentazioni:

Diagramma a blocchi

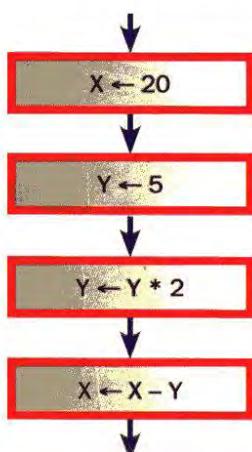


Pseudolinguaggio

VARIABILI
 $K: \text{INTERO}$
 ...
 $K \leftarrow K + 1$

Supponiamo che il valore iniziale della variabile K sia 3. Si valuta l'espressione $K + 1$, recuperando il valore di K (cioè 3) e sommandogli 1. Si pone il risultato nella variabile K dopo avere eliminato il valore precedentemente presente. Pertanto, il valore finale della variabile K è 4. Vediamo un esempio più completo:

Diagramma a blocchi



Pseudolinguaggio

VARIABILI
 $X, Y: \text{INTERO}$
 ...
 $X \leftarrow 20$ //inizializzazione della variabile X
 $Y \leftarrow 5$ //inizializzazione della variabile Y
 $Y \leftarrow Y * 2$ //in Y viene memorizzato 10 (cioè $5 * 2$)
 $X \leftarrow X - Y$ //in Y viene memorizzato 10 (cioè $20 - 10$)



LEZIONE

14



Esaminiamo alcuni esempi di istruzioni operative di input. Supponiamo di voler acquisire in input (ossia dalla tastiera) il valore della variabile *Lato*. L'istruzione:

Diagramma a blocchi



Pseudolinguaggio

LEGGI(Lato)

nell'ambiente $\{(Lato, ?, intero)\}$ rappresenta l'ordine impartito all'esecutore di introdurre un valore inserito dalla tastiera all'interno della variabile *Lato*.

LEGGI(Lato)

Quando il computer incontra l'istruzione **LEGGI** attende che l'utente inserisca un dato da tastiera



5

A digitazione avvenuta, confermata dalla pressione del tasto Invio, il dato viene trasferito nella variabile *Lato*



A input effettuato, il nuovo ambiente di valutazione sarà $\{(Lato, 5, intero)\}$. Con una istruzione operativa di input è possibile acquisire il valore di più variabili. Per esempio, l'istruzione:

Diagramma a blocchi



Pseudolinguaggio

LEGGI(Base, Altezza)

nell'ambiente $\{(Base, ?, intero), (Altezza, ?, intero)\}$ rappresenta l'ordine impartito all'esecutore di introdurre un valore inserito dalla tastiera all'interno della variabile *Base* e, dopo aver premuto il tasto Invio, di introdurre un valore nella variabile *Altezza*.

LEGGI(Base, Altezza)

Quando il computer incontra l'istruzione **LEGGI** attende che l'utente inserisca un dato da tastiera



8

9

A digitazione avvenuta, confermata dalla pressione del tasto Invio, il dato viene trasferito nella variabile *Base* e il computer si pone nuovamente in attesa per permettere all'utente di inserire il valore della variabile *Altezza*





Vediamo ora alcuni esempi di istruzioni di output. Supponiamo di voler "scrivere" un messaggio sul monitor del computer. La sola cosa che occorre ricordare è che il messaggio in questione deve essere racchiuso tra virgolette.

Diagramma a blocchi**Pseudolinguaggio**

SCRIVI("Ciao, come stai?")

La precedente istruzione scrive sul monitor la frase *Ciao, come stai?* Vediamo come viene eseguita.

SCRIVI("Ciao!! Come stai?")

Quando il computer incontra l'istruzione **SCRIVI**("Ciao!! Come stai?") visualizza sullo schermo la dicitura racchiusa tra virgolette



Per scrivere sul monitor il contenuto di una variabile è sufficiente scrivere tra parentesi il suo nome.

Diagramma a blocchi**Pseudolinguaggio**

SCRIVI(Lato)

La precedente istruzione scrive sul monitor il contenuto della variabile *Lato*. Vediamo come viene eseguita.

SCRIVI(Lato)



Quando il computer incontra l'istruzione **SCRIVI**(Lato) accede alla locazione di memoria riservata alla variabile *Lato* e ne visualizza il contenuto sul monitor



Supponiamo, ora, di voler scrivere sul monitor del computer un messaggio e il contenuto di una variabile. Occorre scrivere il messaggio racchiuso tra virgolette, seguito da una virgola e dal nome della variabile.

Diagramma a blocchi**Pseudolinguaggio**

SCRIVI("Il lato misura cm", Lato)

SCRIVI("Il lato misura cm", Lato)



Tutto in test...a

PROVE OGGETTIVE PER LA VERIFICA DELLE CONOSCENZE

- 1** Che cosa si intende con il termine azione?
- 2** Un'azione elementare:
- ☐ a può essere scomposta in azioni basilari
 - ☐ b non può essere scomposta in altre azioni più semplici
 - ☐ c è anche detta istruzione
 - ☐ d non è compresa in modo univoco dall'esecutore
- 3** Il problema: "Determinare il volume di una piramide quadrata di altezza 12,4 cm":
- ☐ a non può essere risolto, poiché non sono esplicitamente definiti i risultati richiesti
 - ☐ b non può essere risolto, poiché non è scindibile in azioni elementari
 - ☐ c non può essere risolto, poiché non è presente un dato significativo
 - ☐ d può essere risolto in un numero finito di azioni elementari
 - ☐ e nessuna delle precedenti affermazioni è corretta
- 4** Nel problema: "Un computer costa € 350,00; vengono pagati € 25,00 in contanti e per il saldo viene richiesto il 20%. Il resto viene rateizzato. Stabilisci l'ammontare della rata mensile, sapendo che il prestito dovrà essere rimborsato in un anno":
- ☐ a i dati sono eccessivi, pertanto il problema ammette più di una soluzione
 - ☐ b i dati sono insufficienti: non viene precisato il numero di rate
 - ☐ c il risultato richiesto dal problema è ambiguo: non è chiaro quando dovrà essere pagato l'interesse
 - ☐ d i dati sono precisi e i risultati possono essere determinati univocamente
 - ☐ e nessuna delle precedenti affermazioni è corretta
- 5** Quando un procedimento risolutivo può essere definito un algoritmo?
- 6** Quale tra le seguenti non è una caratteristica degli algoritmi?
- ☐ a finitezza
 - ☐ b generalità
 - ☐ c completezza
 - ☐ d razionalità
- 7** La caratteristica della non ambiguità di un algoritmo fa riferimento al fatto che:
- ☐ a un algoritmo deve essere valido per tutti i problemi di una classe
 - ☐ b ogni azione deve essere rigorosamente definita nei suoi effetti
 - ☐ c a ogni unità finita di tempo l'esecutore deve scegliere e compiere una e una sola azione
 - ☐ d la strategia risolutiva descritta dall'algoritmo deve essere composta da un numero finito di azioni
- 8** Una categoria sintattica è:
- ☐ a una parola chiave di un linguaggio
 - ☐ b un elemento del linguaggio che deve essere scritto così come è riportato
 - ☐ c un elemento del linguaggio che deve essere ulteriormente specificato
 - ☐ d un elemento di un linguaggio che deve essere scritto totalmente in maiuscolo
- 9** Una variabile è caratterizzata da:
- ☐ a un nome
 - ☐ b un valore
 - ☐ c un indirizzo fisico di memoria
 - ☐ d un tipo
- 10** L'operazione che definisce il tipo di dato di una variabile prende il nome di:
- ☐ a assegnazione
 - ☐ b inizializzazione
 - ☐ c tipizzazione
 - ☐ d dichiarazione
- 11** I dati alfanumerici sono anche detti:
- ☐ a costanti
 - ☐ b stringhe
 - ☐ c booleani
 - ☐ d reali
- 12** Che cosa si intende con "tipo di dato" e, in particolare, che cosa significa "assegnare un tipo di dato a una variabile"?

Tutto in test...a

PROVE OGGETTIVE PER LA VERIFICA DELLE CONOSCENZE

13 Completa la seguente frase:

Un tipo di dato è caratterizzato da un insieme X di valori che rappresenta il _____ del tipo di dato, un insieme non vuoto di costanti che caratterizzano gli _____ di X e un insieme di _____ su X.

14 I tipi di dati elementari:

- ☐ a sono anche detti atomici
- ☐ b sono anche detti semplici
- ☐ c non sono costituiti da altri dati
- ☐ d sono costituiti dai soli dati interi e reali

15 I tipi di dati strutturati

- ☐ a sono anche detti primitivi
- ☐ b non sono costituiti da altri dati
- ☐ c possono essere costituiti da altri dati elementari
- ☐ d possono essere costituiti da altri dati strutturati

16 Che cosa si intende con la sigla ADT?

17 L'intervallo di definizione dei dati di tipo intero è, generalmente:

- ☐ a -32768, +32767
- ☐ b -127, +127
- ☐ c 0, 255
- ☐ d -65536, +65536

18 Stabilisci se le seguenti affermazioni sono vere o false:

- ☐ a il tipo di dato viene definito esclusivamente in funzione del valore che il dato stesso può assumere ☐ V ☐ F
- ☐ b i dati numerici sono per forza senza decimali ☐ V ☐ F
- ☐ c il peso specifico dell'acqua è rappresentato con una variabile ☐ V ☐ F

19 Che cos'è l'ambiente di valutazione di un'espressione?

20 Che cos'è la precedenza di un operatore? Fai un esempio di come funziona

21 Che cosa si intende per aritmetica modulare?

22 A che cosa serve il codice ASCII?

23 Una sequenza di caratteri prende il nome di:

- ☐ a alfanumerica
- ☐ b stringa
- ☐ c variabile
- ☐ d costante

24 Il processo di valutazione di un'espressione stabilisce:

- ☐ a l'ordine di esecuzione dei vari operatori
- ☐ b le variabili che devono essere considerate per il calcolo del risultato
- ☐ c la differenziazione tra variabili e costanti
- ☐ d il solo modo secondo cui l'espressione può essere eseguita dall'esecutore

25 Che cosa si intende per ambiente di valutazione di un'espressione?

26 Qual è la differenza tra pseudolinguaggio e pseudocodifica?

27 Stabilisci se le seguenti affermazioni sono vere o false:

- ☐ a le istruzioni operative fanno parte delle istruzioni di controllo ☐ V ☐ F
- ☐ b il simbolo = rappresenta l'operatore di assegnazione ☐ V ☐ F
- ☐ c l'assegnazione modifica l'ambiente di valutazione ☐ V ☐ F

28 Le istruzioni operative sono quelle che:

- ☐ a corrispondono ad azioni direttamente eseguibili dall'esecutore
- ☐ b servono per acquisire i dati iniziali
- ☐ c non sono utilizzate per effettuare le elaborazioni
- ☐ d comunicano i risultati finali

29 Le istruzioni di controllo:

- ☐ a si suddividono in istruzioni di input e di output
- ☐ b si classificano in istruzioni operative e dichiarative
- ☐ c consentono di instradare percorsi differenti durante l'esecuzione
- ☐ d consentono di instradare il percorso definito dal programmatore

Tutto in test...a

PROVE OGGETTIVE PER LA VERIFICA DELLE CONOSCENZE

30 L'istruzione di assegnazione:

- ☐ a attribuisce un valore a un'espressione
- ☐ b attribuisce un valore a una variabile
- ☐ c è caratterizzata dall'operatore binario identificato dal simbolo \leftarrow
- ☐ d è caratterizzata dall'operatore binario identificato dal simbolo \rightarrow

31 Completa la seguente frase:

La "scrittura" di una variabile, cioè _____ di un valore alla variabile rappresenta un'operazione _____

32 Completa la seguente frase:

Secondo le regole dello pseudolinguaggio i blocchi racchiusi tra parentesi quadre ([]) indicano _____

33 Stabilisci se le seguenti affermazioni sono vere o false

- ☐ a il linguaggio dei diagrammi a blocchi viene utilizzato per descrivere un algoritmo ☐ V ☐ F
- ☐ b linguaggio di progetto e pseudolinguaggio sono sinonimi ☐ V ☐ F
- ☐ c linguaggio di progetto e pseudocodifica sono sinonimi ☐ V ☐ F
- ☐ d una categoria sintattica è un elemento generale del linguaggio che deve essere sostituita con un'opportuna occorrenza ☐ V ☐ F

34 Stabilisci se le seguenti affermazioni sono vere o false:

- ☐ a il risultato di una divisione è un dato di tipo intero ☐ V ☐ F
- ☐ b il numero civico di un'abitazione è di tipo reale ☐ V ☐ F
- ☐ c il numero dei mesi dell'anno può essere memorizzato all'interno di una costante ☐ V ☐ F
- ☐ d l'indirizzo di un'abitazione è di tipo stringa ☐ V ☐ F
- ☐ e il numero di telefono di un amico è di tipo stringa ☐ V ☐ F

35 Al termine dell'esecuzione della pseudoistruzione:

$A \leftarrow 5 * 8$

la variabile A contiene il valore:

- ☐ a 0
- ☐ b 40
- ☐ c 15
- ☐ d non si sa, poiché non è noto il valore che aveva la variabile A prima dell'esecuzione dell'istruzione

36 Al termine dell'esecuzione della seguente istruzione:

$A \leftarrow A * 8$ la variabile A contiene il valore:

- ☐ a 0
- ☐ b 40
- ☐ c 15
- ☐ d non si sa, poiché non è noto il valore che aveva la variabile A prima dell'esecuzione dell'istruzione

37 Al termine dell'esecuzione del seguente blocco

$A \leftarrow 0$

$A \leftarrow A * 8$

la variabile A contiene il valore:

- ☐ a 0
- ☐ b 40
- ☐ c 15
- ☐ d non si sa, poiché non è noto il valore che aveva la variabile A prima dell'esecuzione dell'istruzione

38 La variabile A vale 3 e la variabile B vale 2. Stabilisci quale valore assume la variabile C dopo l'esecuzione dell'istruzione:

$C \leftarrow A + B - 3$

- ☐ a -2
- ☐ b 0
- ☐ c 1
- ☐ d 2

39 La variabile A vale 3 e la variabile B vale 2. Stabilisci quale valore assume la variabile C dopo l'esecuzione del seguente blocco:

$C \leftarrow 2$

$C \leftarrow A + B - 3$

- ☐ a -2
- ☐ b 0
- ☐ c 1
- ☐ d 2

40 L'istruzione $A \leftarrow B + 1$:

- ☐ a assegna alla variabile A il valore della variabile B e, successivamente, aggiunge 1
- ☐ b somma 1 al contenuto della variabile B, quindi assegna alla variabile A il risultato dell'operazione
- ☐ c è errata in quanto l'operatore di assegnazione è =
- ☐ d nessuna delle precedenti affermazioni è corretta

Training

PROVE APERTE PER LA VERIFICA DELLE ABILITÀ

1 Di seguito è riportata la ricetta per realizzare una parmigiana di melanzane. Dettaglia il procedimento in modo rigoroso evidenziando le azioni elementari.

2 Dato il seguente ambiente di valutazione:

(a,10), (b, 20), (c, 15)

qual è il nuovo ambiente di valutazione nei seguenti casi?

- a) $a \leftarrow 10 * b$
- b) $b \leftarrow a * b$
- c) $c \leftarrow a + 10 * b$
- d) $a \leftarrow a + 10 * b$
- e) $b \leftarrow a$
- f) $a \leftarrow a$

3 Scrivi accanto a ogni identificatore se può essere dichiarato come costante o variabile:

Pigreco _____

AltezzaTriangolo _____

PesoPersona _____

PesoSpecificoAcqua _____

Età _____

EtàMaggiorenne _____

4 Per i seguenti esercizi:

- individua le costanti e le variabili
- individua i dati in input e in output
- descrivi l'evoluzione dell'ambiente di valutazione
- riporta la strategia risolutiva
- a) Calcola il doppio di un numero fornito in input
- b) Scambia il contenuto di due variabili
- c) Calcola la misura dell'ipotenusa di un triangolo rettangolo noti i cateti
- d) Ottieni una bibita da un distributore automatico
- e) Prepara un piatto di spaghetti al sugo
- f) Scatta una fotografia
- g) Travasa un contenitore d'acqua in 30 bottiglie da un litro
- h) Aggiorna l'orario di un orologio digitale
- i) Aggiorna la data di un orologio digitale
- j) Cambia un pneumatico (forato) di un'auto
- k) Sintonizzati sul programma televisivo di tuo interesse
- l) Registra dalla radio una canzone con il registratore

5 Disegna il simbolo che indica l'inizio di un diagramma a blocchi.

6 Disegna il simbolo che indica la fine di un diagramma a blocchi.

7 Disegna il simbolo per effettuare un calcolo.

8 Disegna il simbolo per assegnare alla variabile x il valore 25.

9 Disegna il simbolo per scrivere sullo schermo la scritta "Ok funziona!".

10 Disegna il simbolo per scrivere sullo schermo il risultato del calcolo $(5 - 3) * (5 + 3)$.

11 Disegna il simbolo per assegnare alla variabile N il risultato del calcolo $(12 - 8) * (1 + 2)$.

12 Disegna il simbolo per leggere dalla tastiera un valore da memorizzare nella variabile Ipotenusa.

13 Disegna il simbolo corretto intorno alla seguente istruzione:

$Guadagno = Ricavo - Costo$

e, di seguito, correggi l'errore presente:

14 Date le variabili A, B e C qual è il risultato delle seguenti assegnazioni?

► $C \leftarrow A$

► $A \leftarrow B$

► $B \leftarrow C$

15 Associa alla tre pseudoistruzioni i corrispondenti simboli della diagrammazione a blocchi:

► LEGGI(Ricavo)

► $A \leftarrow 5$

► SCRIVI(A)

16 Associa alle tre pseudoistruzioni i corrispondenti simboli della diagrammazione a blocchi scrivendoli uno di seguito all'altro come a formare un algoritmo.

► LEGGI(Base)

► LEGGI(Altezza)

► $Area \leftarrow (Base * Altezza) / 2$

► SCRIVI("L'area è ")

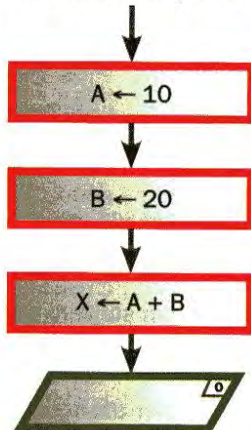
► SCRIVI(Area)

Training

PROVE APERTE PER LA VERIFICA DELLE ABILITÀ

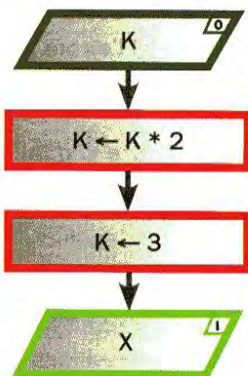
- 17** Modifica l'esercizio 16 introducendo nel diagramma i blocchi che determinano l'inizio e la fine del procedimento. Poi, accorpa le due ultime pseudoistruzioni in modo da ottenere una sola istruzione di output.

- 18** Traduci il seguente blocco in pseudoistruzioni:



Qual è il valore della variabile X visualizzato?

- 19** Il seguente blocco di istruzioni deve richiedere da tastiera il valore della variabile K, moltiplicarla per due e dopo il calcolo visualizzare sul video il valore ottenuto. Sono presenti, però, tre errori. Sapresti correggerli?



- 20** Le seguenti pseudoistruzioni presentano degli errori. Riscrivile correttamente.

- **SCRIVI**('Area =')
- **LEGGI** ("Lato")
- **A** = 2
- **X**: = **INTERO**
- **Nome** : = **STRINGA**(25)

- 21** Dopo aver trovato gli errori presenti nel seguente blocco di pseudoistruzioni, traducilo secondo il formalismo dei diagrammi a blocchi.

- **VARIABILI**
- **X, Y: INTERO**
-
- **X** = 120
- **Y** ← '5'
- **K** ← **X** * **Y**
- **X** ← **X** - **Y**
- **SCRIVI**('Il risultato è' K)

- 22** Commenta il seguente algoritmo:



- 23** Scrivi gli algoritmi risolutivi dei seguenti problemi:

- a) calcola il doppio di un numero fornito in input
- b) scambia il contenuto di due variabili
- c) calcola la misura dell'ipotenusa di un triangolo rettangolo noti i cateti
- d) calcola l'area di un rettangolo note le misure della base e dell'altezza

- 24** Scrivi gli algoritmi risolutivi dei seguenti problemi:

- a) risolvi l'equazione di primo grado $ax=b$
- b) calcola l'area e la misura di una circonferenza noto il raggio

Training

PROVE APERTE PER LA VERIFICA DELLE ABILITÀ

25 Dopo aver trovato gli errori presenti nel seguente blocco di pseudoistruzioni, traducilo secondo il formalismo dei diagrammi a blocchi

```
► VARIABILE  
► A, B: CARATTERI  
► C, D: INTERI  
►  
► INIZIO  
►  $A \leftarrow C + D$   
►  $A = A^2$   
► Nuovo  $\leftarrow C + D$ ;  
►  $C \leftarrow C + 2$ ;  
► LEGGI(Variab)  
►  $C \leftarrow C + \text{Variab}$   
► SCRIVI('Il risultato è', Variab)  
► FINE
```

26 Trasforma le seguenti descrizioni rigorose in pseudocodice e in diagramma a blocchi.

- Acquisisci in input (leggi) il valore del lato ponendolo in L
- Esegui l'operazione $L * 4$
- Poni il risultato nella variabile P
- Stampa il valore di P

27 Trasforma le seguenti descrizioni rigorose in pseudocodice e in diagrammi a blocchi.

- Leggi il numero dei lati e la misura del lato ponendoli in N e L
- Esegui l'operazione $L * N$
- Poni il risultato nella variabile P
- Stampa il valore di P

28 Trasforma il seguente codice in diagramma a blocchi.

```
► intero L,N,P;  
► leggi (N,L);  
►  $P = N * L$ ;  
► stampa(P);
```

29 Trasforma il seguente codice in diagramma a blocchi.

```
► intero A,B,C;  
► leggi (A, B);  
►  $C = A$ ;  
►  $A = B$ ;  
►  $B = C$ ;  
► stampa(A, B);
```

30 Il seguente diagramma a blocchi presenta simboli non chiari e al loro interno sono presenti istruzioni non scritte correttamente. Correggilo e riporta anche lo pseudocodice.

