

Algoritmi

Algoritmi

- Risolvere un problema significa individuare un procedimento che permetta di arrivare al risultato partendo dai dati
- Il procedimento (chiamato *algoritmo*) è composto da passi elementari
- Il modo di esprimere la sequenza dei passi elementari deve essere standardizzato

Esempi di algoritmi

Ricetta per cucinare gli spaghetti

- Metti l'acqua nella pentola
- Fai bollire l'acqua
- Metti la pasta nell'acqua
- Aggiungi un po' di sale
- Attendi 6 minuti
- Scola la pasta

Esempi di algoritmi

Verifica se un numero è pari o dispari

- Prendi il numero
- Calcola il resto della divisione intera del numero per 2
- Se il resto è zero
 - Allora il numero è pari
 - Altrimenti il numero è dispari

Esempi di algoritmi

Ricerca utente in un elenco telefonico:

Dati: un insieme ordinato di coppie (*nome, numero*)
un nome **X** da ricercare

- 1) Leggi la prima coppia (*nome, numero*)
- 2) **Se** non hai oltrepassato l'ultima coppia **allora**:
 - **Se** *nome* = **X** **allora** il numero di telefono di **x** è quello letto;
 - **Altrimenti** leggi la prossima coppia e vai al passo 2).
- **Altrimenti**
 - Il nome **X** cercato non è nell'elenco

Esempi di algoritmi

Ricerca di un utente X in un elenco telefonico:

(alternativa)

- 1) **Leggi** la prima coppia (*nome, numero*)
 - 2) **Se** non hai oltrepassato l'ultima coppia **e** *nome* precede X (secondo l'ordine alfabetico) **allora** osserva la prossima coppia e ripeti il passo 2)
 - 3) **Se** hai oltrepassato l'ultima coppia **oppure** *nome* segue X **allora** X non è presente nell'elenco
- **Altrimenti** il num. tel. di X è l'ultimo letto;

Proprietà degli algoritmi

- L'algoritmo è caratterizzato da:
 - **finitezza**: composto da un numero finito di passi elementari; le operazioni sono eseguite un numero finito di volte
 - **non ambiguità**: i risultati non variano in funzione della macchina/persona che esegue l'algoritmo (deterministico)
 - **realizzabilità**: deve essere eseguibile con le risorse a disposizione

Proprietà degli algoritmi

- *... ma gli esempi precedenti possiedono queste proprietà?*
- Problemi presenti:
 - Ambiguità
 - Ipotesi implicite sulle capacità dell'esecutore
 - Uso del linguaggio naturale

Algoritmo

- Per definire un algoritmo è necessario:
 - condurre un'attenta analisi del problema
 - individuare i possibili ingressi
 - precisare le uscite
 - definire completamente e dettagliatamente la sequenza dei passi che portano alla soluzione
 - è conveniente suddividere il problema in piccoli sottoproblemi*

Rappresentazione degli algoritmi

- E' necessario far riferimento a dei formalismi che:
 - non introducano ambiguità
 - siano universalmente riconosciuti ed interpretati allo stesso modo da un generico esecutore
 - permettano di rappresentare in modo efficace un algoritmo
 - Costituiscono un utile strumento per poi poter passare alla fase di codifica in un linguaggio di programmazione

Rappresentazione degli algoritmi

Rappresentazione grafica

Diagrammi a blocchi / Flow Chart

Rappresentazione testuale

Notazione Lineare Strutturata / PseudoCodice

Algoritmi: operazioni base

- Le operazioni primarie sono:
 - Trasferimento di informazioni (istruzioni di I/O)
lettura dati, scrittura risultati, visualizzazione dati intermedi
 - Esecuzione di calcoli (valutazione espressioni)
 - Istruzioni di assegnamento
 - Strutture di controllo (che modificano il flusso sequenziale di esecuzione delle operazioni)

Simboli convenzionali



ingresso/uscita



inizializzazione



documento



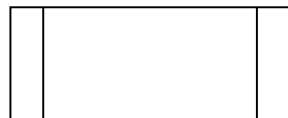
Elaborazione



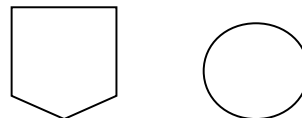
inizio/fine



input manuale



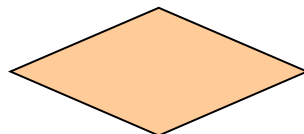
elab. predefinita



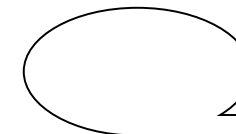
connessioni



disco



decisione

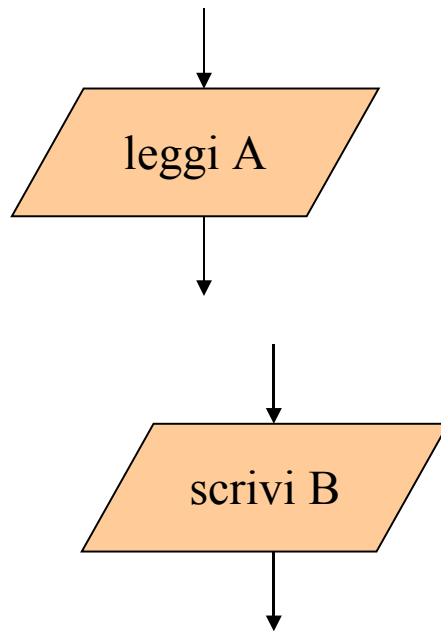


mem. sequenziale

Istruzioni di I/O

- lettura di dati in input
- scrittura dei risultati in output

Diagramma a blocchi



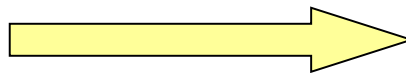
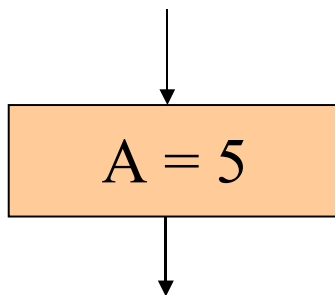
Pseudo-codice

leggi A

scrivi B

Istruzione di assegnamento

- Concetto di **variabile**
 - Identificata da un'**etichetta** / **identificatore simbolico**
 - Può essere comodo pensare alla variabile come ad un contenitore in cui possiamo memorizzare o reperire dei dati utilizzati durante il calcolo
 - L'istruzione di assegnamento permette di **assegnare** un valore ad una variabile

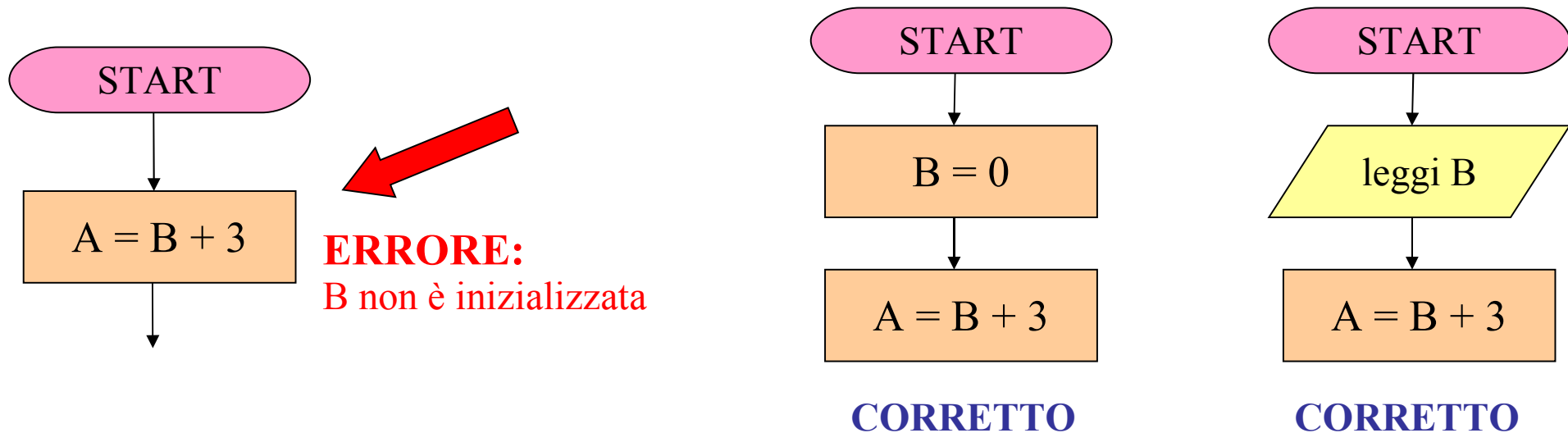


*Alla variabile **A**
viene assegnato il
valore **5***

Istruzione di assegnamento

Inizializzazione

- All'inizio di un algoritmo una variabile non ha alcun valore
- Non ha quindi senso utilizzarla in una espressione (**è un errore!**)
- Essa deve essere inizializzata:
 - O esplicitamente mediante un assegnamento
 - Oppure mediante una operazione di lettura



Istruzione di assegnamento

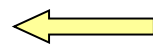
- E' stato usato il simbolo $=$ per indicare l'istruzione di assegnamento (*sarà lo stesso usato nel linguaggio C*)
- Alcuni testi/autori indicano invece il simbolo \leftarrow (*per evitare confusione con l'operatore di uguaglianza*)

Dato il seguente frammento di codice:

...

$A = 5$

$A = A + 1$



Cosa fa quell'istruzione?

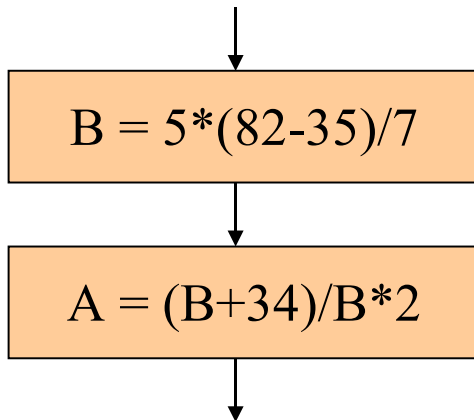
L'esecutore esegue i seguenti passi:

- prima valuta l'espressione a destra, cioè calcola il valore di $A+1$ che vale 6
- dopo assegna tale valore alla variabile a sinistra, cioè ad A
- alla fine dell'esecuzione di quella istruzione quindi, A vale 6

Valutazione espressioni

- L'esecutore è in grado di valutare espressioni aritmetiche

Diagramma a blocchi



Pseudo-codice

$B = 5*(82-35)/7$

$A = (B+34)/B*2$

Rappresentazione degli algoritmi

I diagrammi di flusso possono presentarsi:

- Ad un **livello generale**
- Ad un **livello particolare**

a seconda del livello di dettaglio con cui specificano le operazioni da compiere.

E' opportuno procedere per fasi successive:

- Un diagramma globale (di massima) per focalizzare le operazioni essenziali da compiere
- Un diagramma di flusso più particolareggiato, che tenga conto di operazioni più semplici e più elementari

Tecniche di programmazione

- Programmazione *top-down*:
 - scomposizione iterativa del problema in sottoproblemi
 - i sottoproblemi devono essere indipendenti ed avere interfacce ben definite
 - visibilità dei dettagli di ogni sottoproblema
- Programmazione strutturata