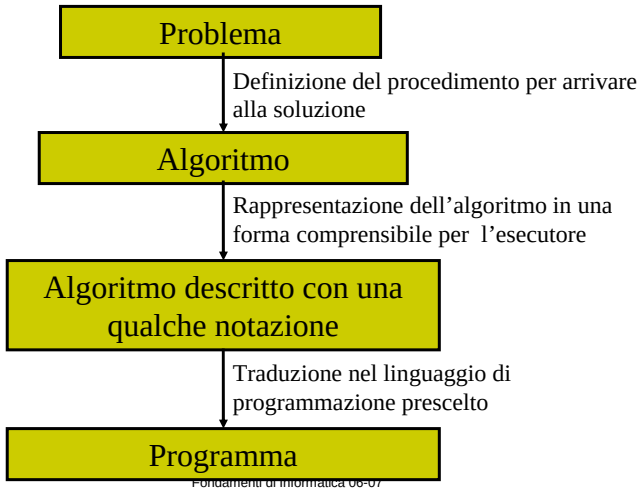
RAPPRESENTAZIONE GLI ALGORITMI

NOTAZIONE PER LA RAPPRESENTAZIONE DI UN ALGORITMO

Rappresentazione degli algoritmi



Linguaggio naturale – notazione lineare

- Nella rappresentazione di un algoritmo si possono riconoscere tre classi di istruzioni:
- istruzioni di ingresso e uscita (dati e risultati del problema)
- istruzioni di assegnamento (P = A con il significato assegna il nome logico P al valore A)
- istruzioni di controllo: sono quelle istruzioni che modificano la sequenza dell'esecuzione
  - alterazione incondizionata
  - alterazione condizionata

Esempio

- Calcolare il prodotto di due numeri interi positivi A e B supponendo che l'esecutore conosca solo le operazioni di somma, sottrazione e confronto fra numeri.

N.istruzione Istruzione

1 leggi A e B
2 P = A
3 se B è uguale 1 andare all'istruzione 7
4 P = P+ A
5 B = B - 1
6 andare alla 3
7 scrivi P
8 fine

Tabella di Traccia

- L'algoritmo può essere testato attraverso l'utilizzo di una tabella di Traccia.
- Esempio per l'algoritmo precedente eseguito per i valori di ingresso 7 e 4

istruzione	A	B	P
inizio			
Leggi A e B	7	4	
P = A			7
B è uguale a 1 ?			
P = P + A			14
B = B - 1		3	
B è uguale a 1 ?			
P = P + A			21
B = B - 1		2	
B è uguale a 1 ?			
P = P + A			28
B = B - 1		1	
B è uguale a 1 ?			
Scrivi P			

Rappresentazione degli algoritmi

Rappresentazione grafica

Diagrammi a blocchi / Flow Chart

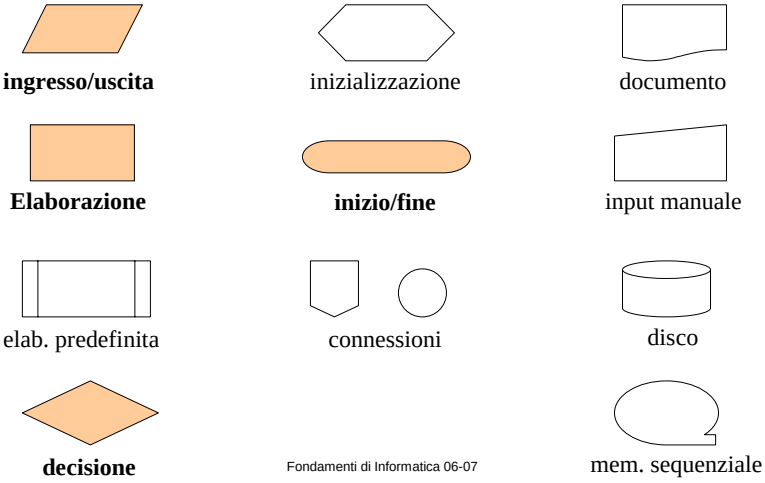
Rappresentazione testuale

Notazione Lineare Strutturata / PseudoCodice

Algoritmi: operazioni base

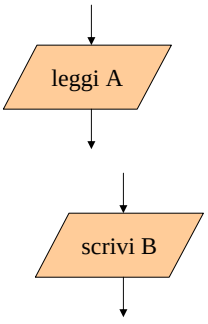
- Le operazioni primarie sono:
  - Trasferimento di informazioni (istruzioni di I/O)
    - lettura dati, scrittura risultati, visualizzazione dati intermedi
  - Esecuzione di calcoli (valutazione espressioni)
  - Istruzioni di assegnamento
  - Strutture di controllo (che modificano il flusso sequenziale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

Simboli convenzionali



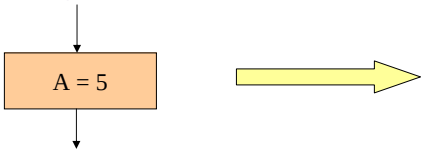
Istruzioni di I/O

- 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

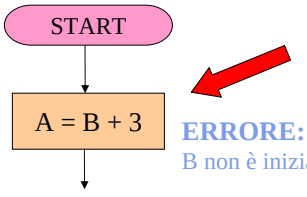


*Alla variabile **A** viene assegnato il valore **5***

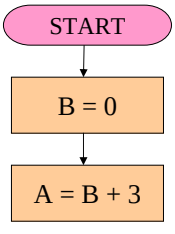
Istruzione di assegnamento

Inizializzazione

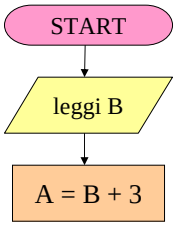
- All'inizio di un algoritmo una variabile non ha alcun valore
- Non ha quindi senso utilizzarla in una espressione (**è un errore!**)
- Essa deve essere inizializzata:
  - O esplicitamente mediante un assegnamento
  - Oppure mediante una operazione di lettura



ERRORE:
B non è inizializzata



CORRETTO



CORRETTO

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 ← (*per evitare confusione con l'operatore di uguaglianza*)

Dato il seguente frammento di codice:

```
...
A = 5
A = A + 1
```

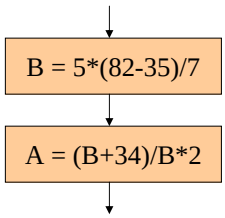
Cosa fa quell'istruzione?

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.

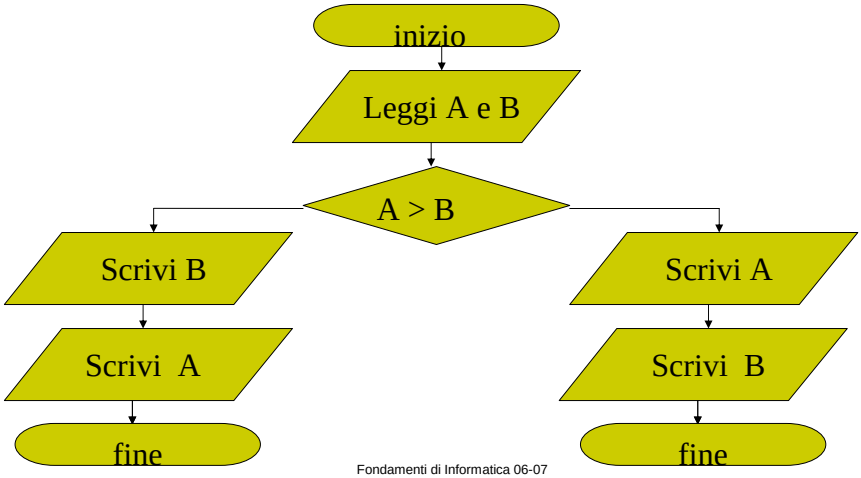
E' opportuno procedere per fasi successive:

- 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

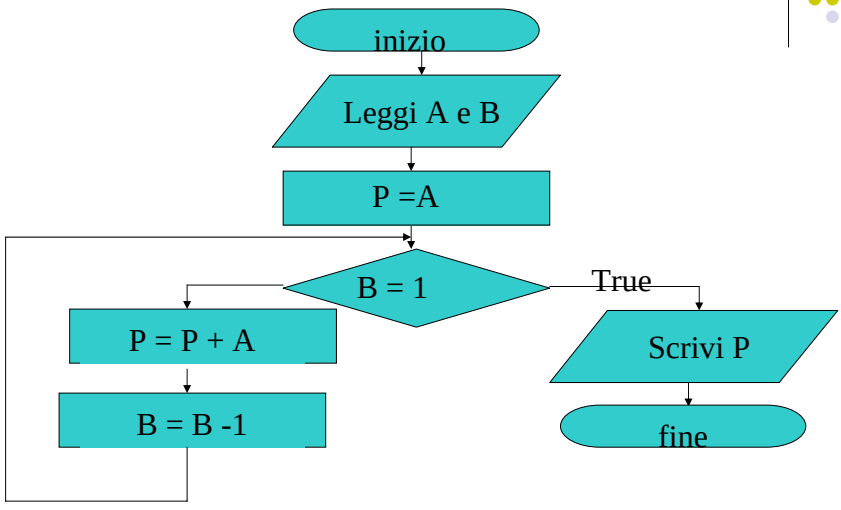
Tecniche di programmazione

- 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

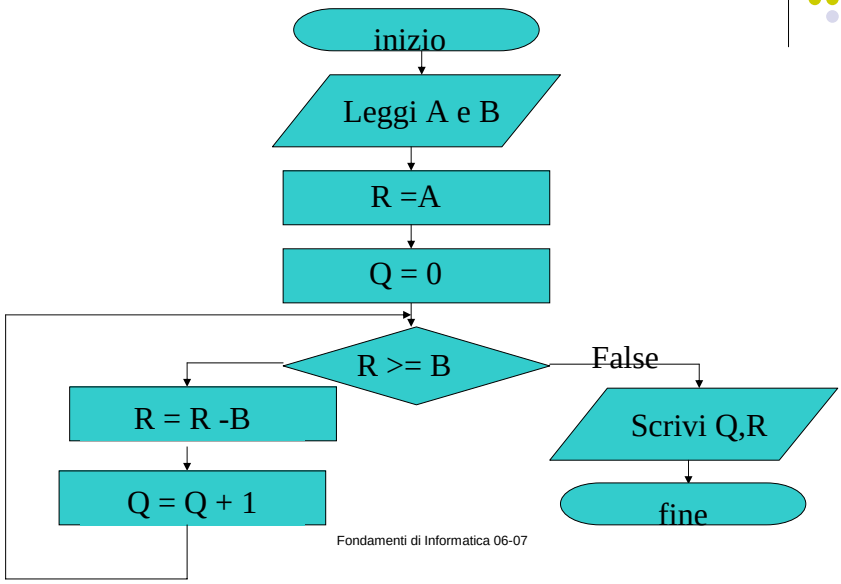
Dato due numeri interi stampare prima il più grande e poi il più piccolo



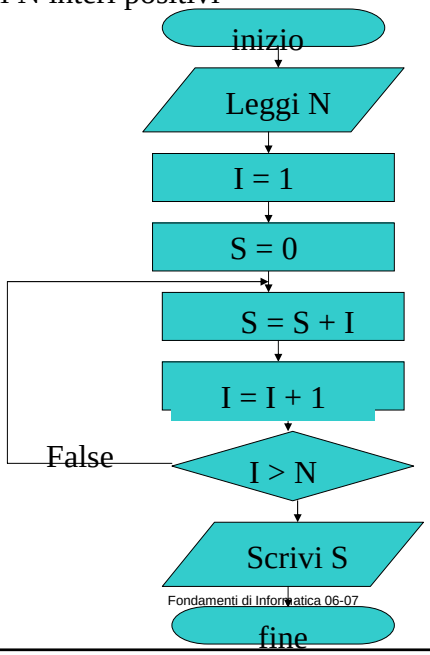
Dato due numeri interi positivi effettuare il loro prodotto con il metodo delle addizioni successive



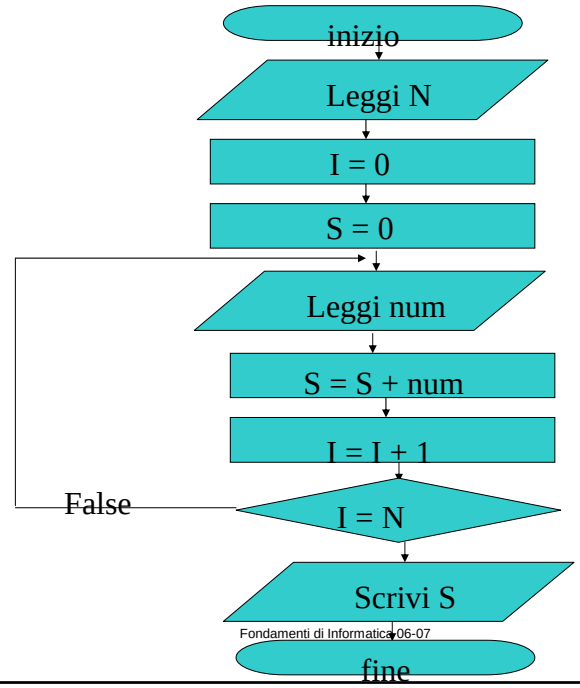
Dato due numeri interi positivi diversi da zero calcolare quoziente e resto con il metodo delle sottrazioni successive



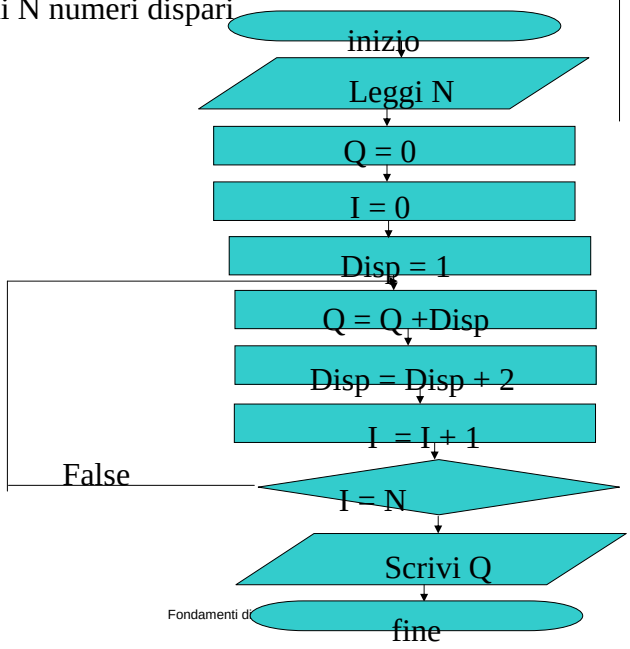
Sommare i primi N interi positivi



Leggere N numeri e scrivere la loro somma.



Calcolare il quadrato di un numero N intero positivo utilizzando la somma dei primi N numeri dispari



Limiti dei diagrammi a blocchi

- I DaB sono generalmente illeggibili, non si riesce a seguire l'algoritmo soprattutto quando superano le dimensioni di un semplice esercizio didattico. La lettura andrebbe fatta un po' dall'alto un po' dal basso senza un ordine preciso
- I DaB sono facilmente esposti ad errori logici, bastano pochi accavallamenti di cicli per perdere il filo del controllo. Ciò nasce dalle correzioni consentite da un indisciplinato uso delle frecce che causa un proliferare di errori logici che si accentua con la complessità del problema e l'inesperienza del risolutore
- Scarsa praticità dovuta alla natura grafica bidimensionale
- difficoltà di riconoscimento della struttura di controllo

Soluzione

- La soluzione a questi problemi consiste nell'imporre una ***disciplina di composizione*** che eviti cattive strutturazioni degli algoritmi
- L'idea base è:
- **un algoritmo deve essere letto dall'alto al basso secondo un ordine sequenziale di esecuzione**
- Cio non significa che non possono esistere dei cicli o dei test, ma che questi siano strutturati in modo da poter essere considerati come un unico blocco operativo con un unico ingresso e una sola uscita.
- Si possono distringuere
  - blocchi semplici
  - blocchi composti

Tutti con un unico punto di ingresso e un unico punto di uscita