1.1 Concetti base dell'Informatica: Algoritmi

Insegnamento di Informatica

Elisabetta Ronchieri

Corso di Laurea di Economia, Universitá di Ferrara

I semestre, anno 2014-2015

Argomenti

Rappresentazione

- Diagramma di flusso
- Notazione lineare
- Pseudocodice

Algoritmi notevoli

- Categorie
- Algoritmi di ricerca
- Algoritmi di ordinamento

Argomenti

Diagramma di flusso

- É una rappresentazione grafica di un algoritmo.
- Descrive le azioni da eseguire e l'ordine di esecuzione.
- É un insieme di blocchi collegati da frecce direzionali.
  - I blocchi rappresentano le operazioni (azioni) dell'algoritmo.
  - Le frecce direzionali indicano il flusso di esecuzione della sequenza di operazioni.
- I blocchi sono espressi da simboli.

Significato dei simboli

Inizio algoritmo

Fine algoritmo

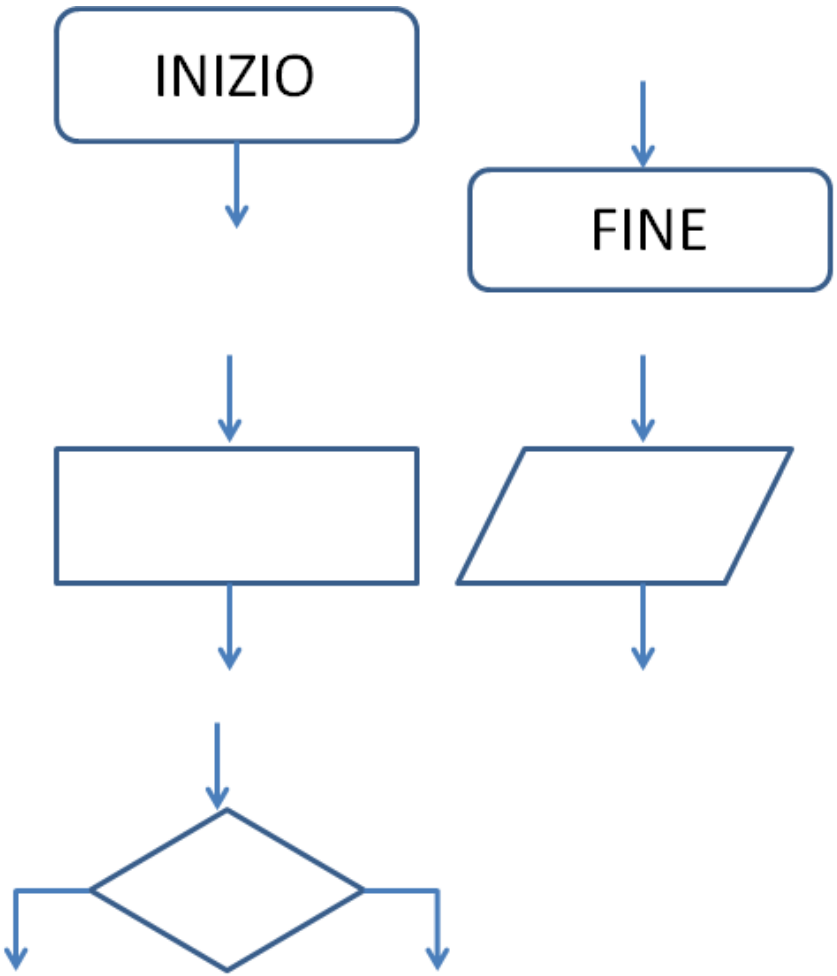
1 azione o
blocco di azione

Ordine di esecuzione
dei vari passi

Condizione: Vera o Falsa

Specifica informazioni in Ingresso e Uscita

Regole di utilizzo



All'inizio è ammesso seguire una sola direzione.

Alla fine si giunge da una sola strada.

Un blocco di azioni, di ingresso o di uscita ha:
- 1 sola freccia di arrivo
- 1 sola di partenza.

Il blocco di condizioni prevede:
- due frecce di uscita
- 1 freccia di arrivo

Caratteristiche: Vantaggi

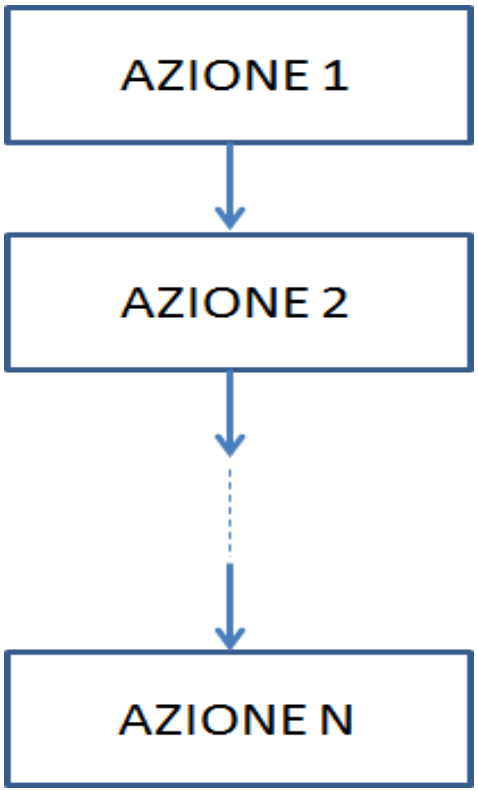
- Grafici
- Adatti a comunicare un algoritmo agli esseri umani
- Adatti a rappresentare processi sequenziali
- Immediatamente visualizzabili
- Poco ambigui
- Convertibile in piú

Caratteristiche: Svantaggi

- Spesso non entrano in una pagina
- Difficili da seguire e modificare
- Non strutturati
- Lontani dai linguaggi dei calcolatori
- Possono rivelarsi errati in fase di programmazione

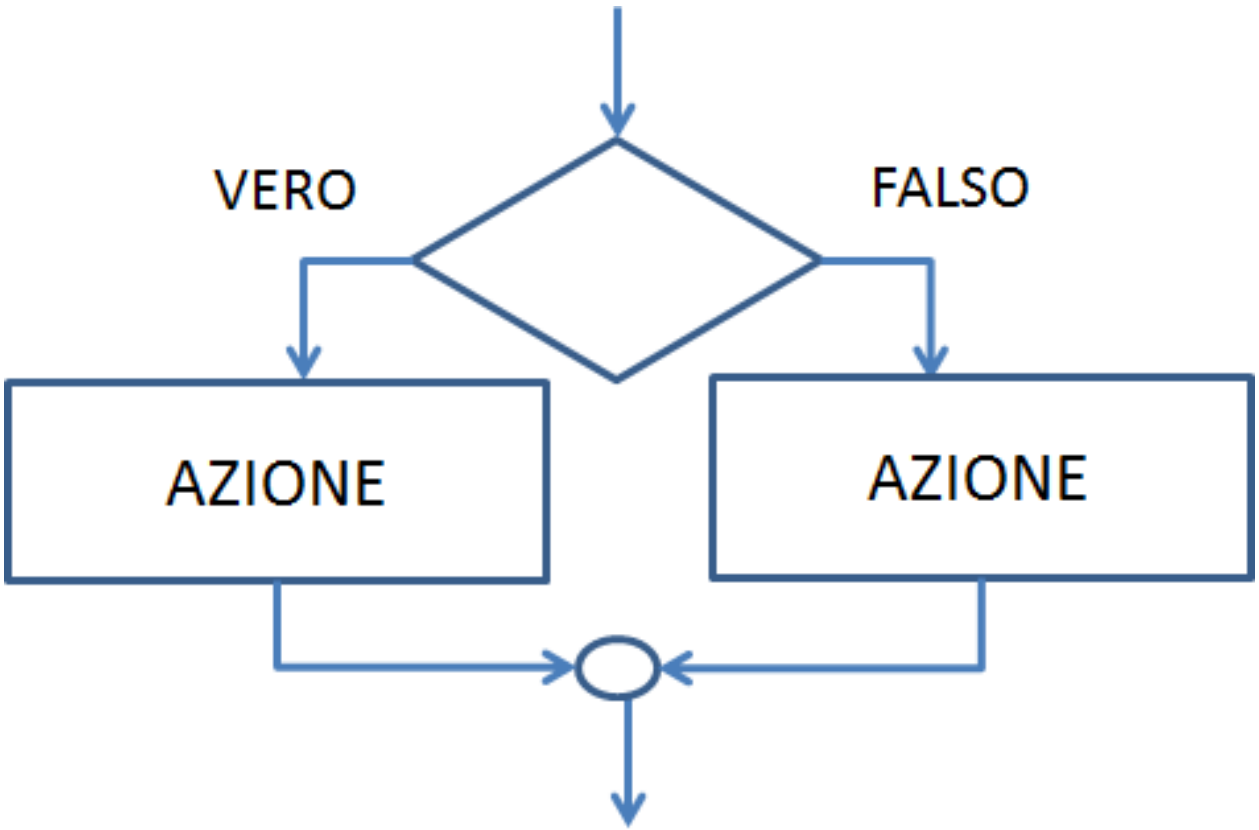
Sequenza di azioni

Ogni azione deve essere eseguita una di seguito all'altra.



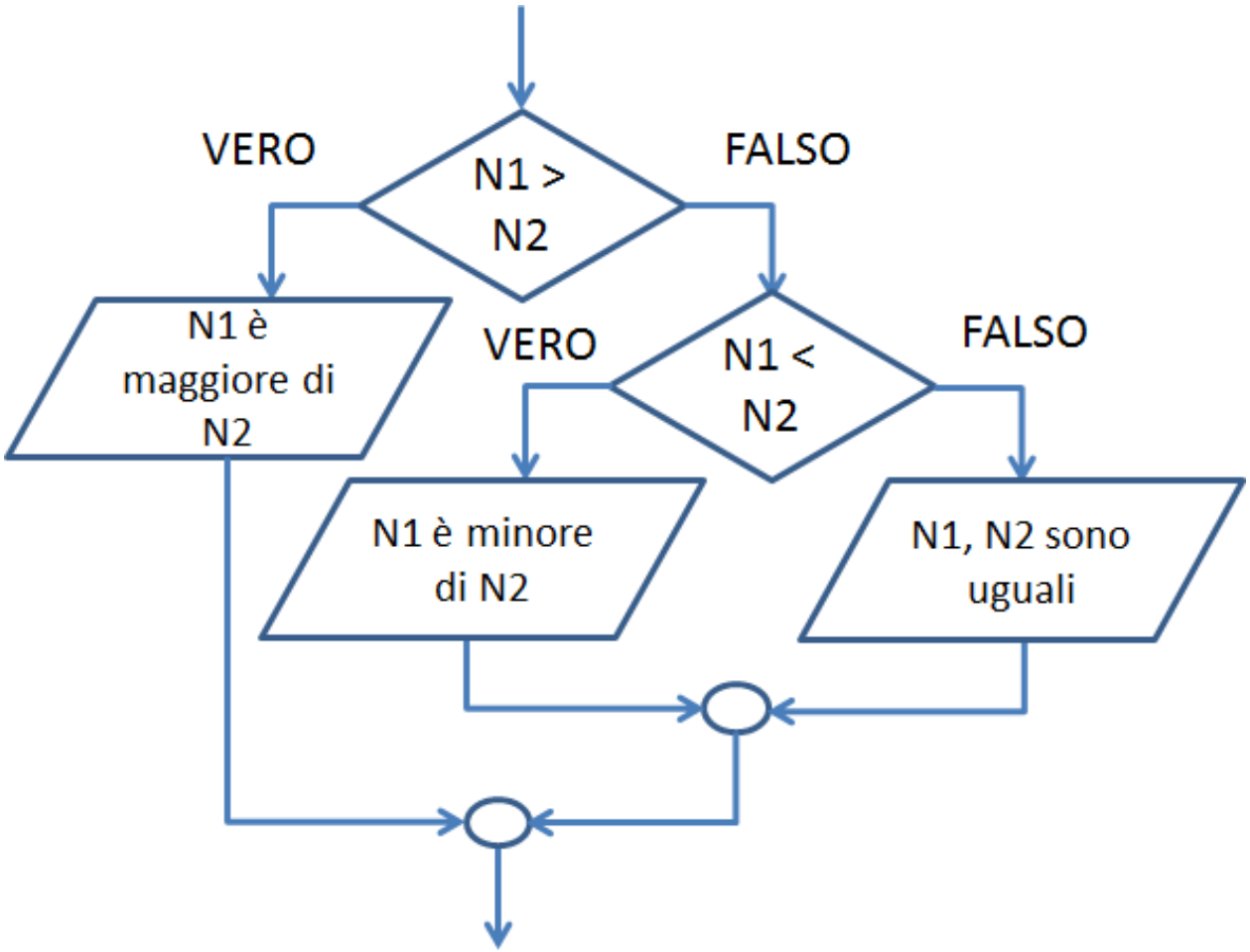
Selezione di percorsi diversi

Se l'istruzione é vera esegui le azioni di un ramo, altrimenti esegui quelle dell'altro ramo.



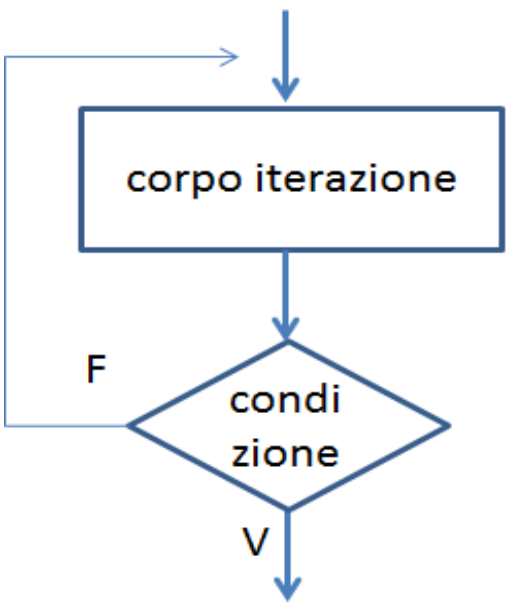
Esempio di diagramma di flusso

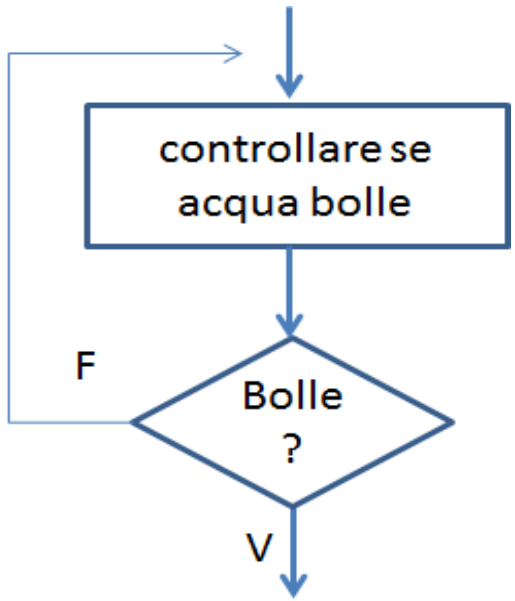
Problema: dati due numeri $N1$ e $N2$ stabilire il maggiore.



Ciclo d'attesa

Un'azione viene ripetuta o iterata.

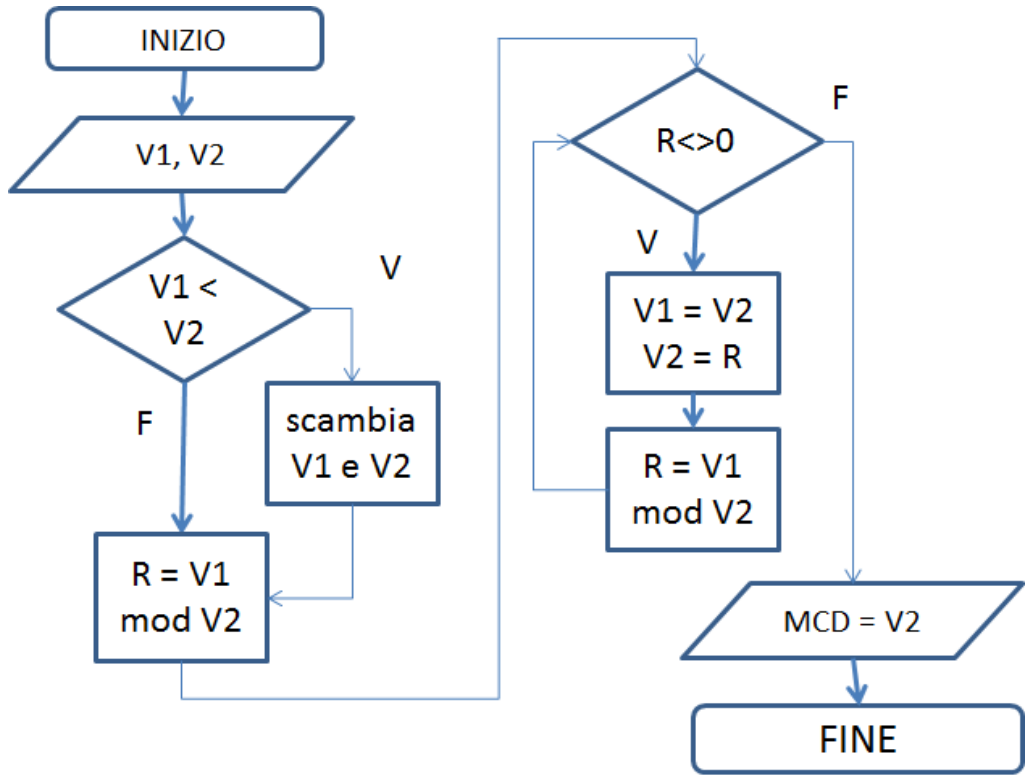




Esempio di diagramma di flusso con ciclo d'attesa

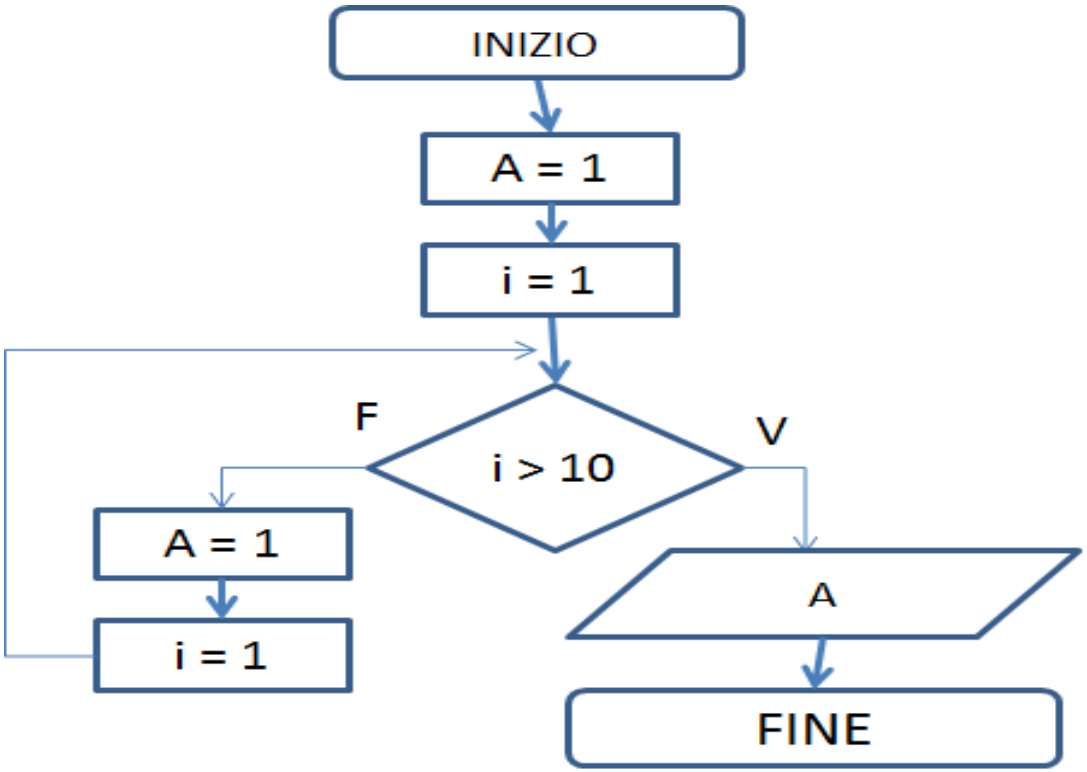
Problema: calcolare il Massimo Comun Divisore (MCD) tra i numeri $V1$ e $V2$.

1. Prendi i valori in ingresso $V1$, $V2$.
2. Se $V1 < V2$ allora scambiali.
3. Metti in R il resto tra $V1$ e $V2$.
4. Ripeti finché $R <> 0$:
    5. Metti in $V1$ il contenuto di $V2$.
    6. Metti in $V2$ il contenuto di R.
    7. Metti in R il resto tra $V1$ e $V2$.
8. Fine ripeti.
9. Mostra MCD pari a $V2$.



Esempio con ciclo infinito

Problema: determinare il numero del contatore *i*.



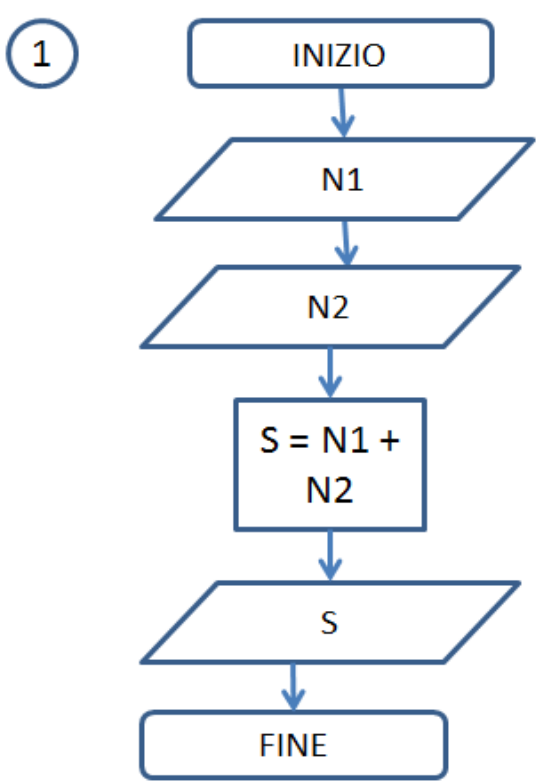
NOTA: In questa soluzione sono presenti almeno due errori.

Come si verifica un algoritmo

- Tramite una prova logica della correttezza della linea operativa tracciata:
  - utilizzando dati campione;
  - calcolando i risultati;
  - controllando i risultati.

Esempio di verifica di un algoritmo

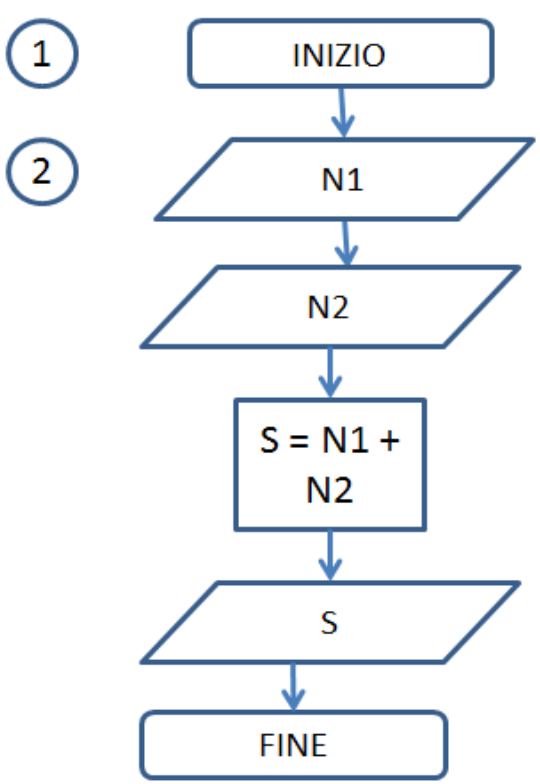
Problema: somma di due numeri $N1$ e $N2$.
Dati iniziali: $N1$ é 12, $N2$ é 35.

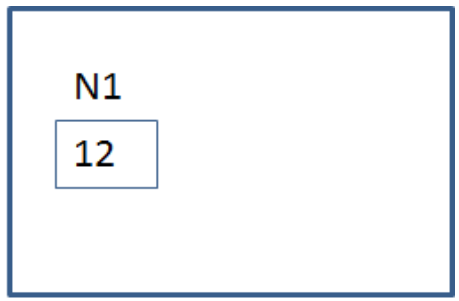


Elaboratore

Esempio di verifica di un algoritmo

Problema: somma di due numeri $N1$ e $N2$.
Dati iniziali: $N1$ é 12, $N2$ é 35.

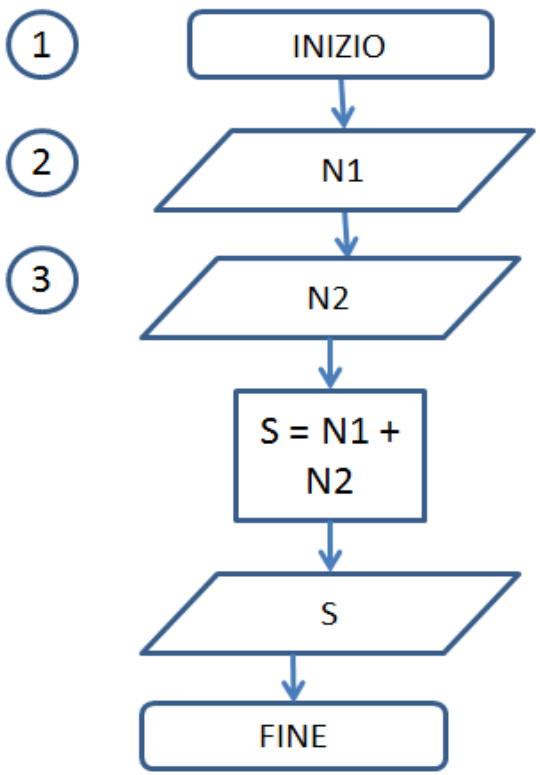


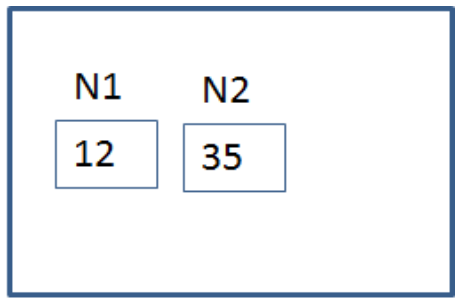


Elaboratore

Esempio di verifica di un algoritmo

Problema: somma di due numeri $N1$ e $N2$.
Dati iniziali: $N1$ é 12, $N2$ é 35.

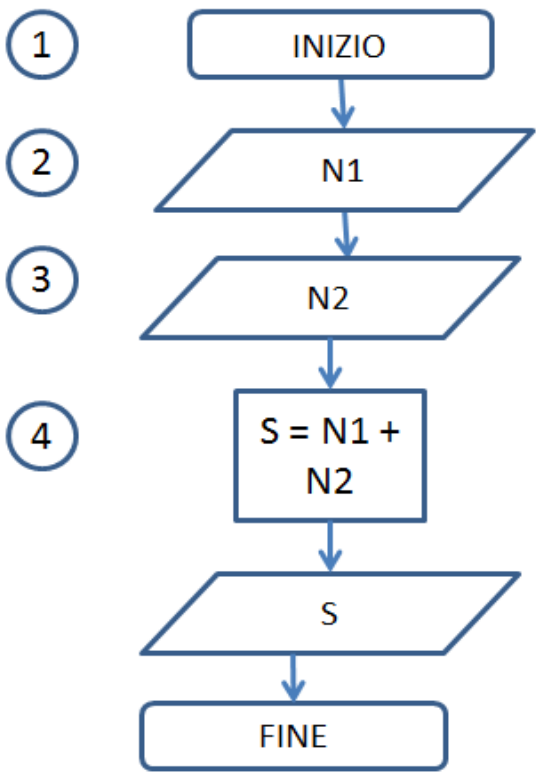


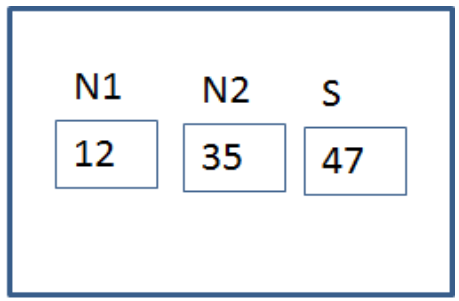


Elaboratore

Esempio di verifica di un algoritmo

Problema: somma di due numeri $N1$ e $N2$.
Dati iniziali: $N1$ é 12, $N2$ é 35.

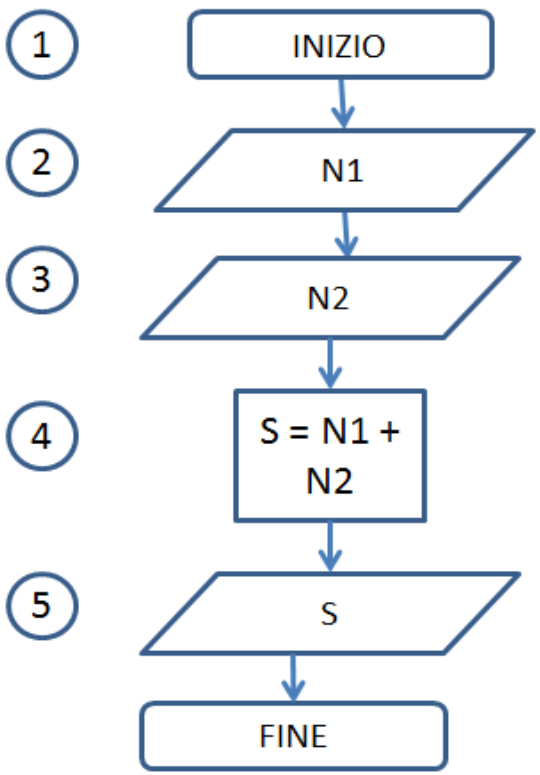


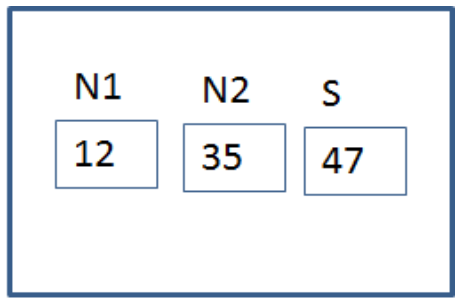


Elaboratore

Esempio di verifica di un algoritmo

Problema: somma di due numeri $N1$ e $N2$.
Dati iniziali: $N1$ é 12, $N2$ é 35.

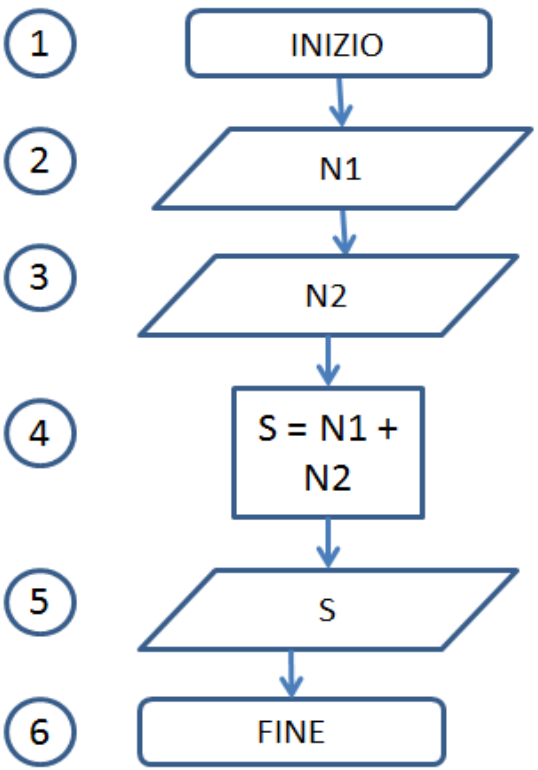




Elaboratore

Esempio di verifica di un algoritmo

Problema: somma di due numeri $N1$ e $N2$.
Dati iniziali: $N1$ é 12, $N2$ é 35.



Elaboratore

Come si progetta un algoritmo

- Dato un problema: determinare il massimo tra due numeri $N1$ e $N2$.
- Suddividere il problema in azioni:
  1. Leggi il primo numero e mettilo nella cella $N1$.
  2. Leggi il secondo numero e mettilo nella cella $N2$.
  3. Se il numero contenuto nella cella $N1 >$ o uguale al numero contenuto nella cella $N2$:
  4. Metti nella cella *max* il contenuto di $N1$.
  5. Altrimenti metti nella cella *max* il contenuto di $N2$.
  6. Restituisci il valore di *max*.
- Realizzare il diagramma di flusso.
- Verificare l'algoritmo.

Progettare un algoritmo

Problema: dati capra, lupo e cavolo gestire le seguenti azioni:

- Porta la capra sull'altra sponda.
- Torna indietro.
- Porta il cavolo sull'altra sponda.
- Porta indietro la capra.
- Porta il lupo sull'altra sponda.
- Torna indietro.
- Porta la capra sull'altra sponda.

Progettare un algoritmo

Problema: risolvere l'equazione lineare $ax + b = 0$ nell'incognita x.

- Leggi i valori a e b.
- Calcola $-b$.
- Dividi quello che hai ottenuto per a e chiama x il risultato.
- Stampa x.

NOTA: Introdurre nella sequenza di azioni il controllo sul valore a.

Progettare un algoritmo

Problema: stabilire se una parola P viene alfabeticamente prima della parola Q.
Dati iniziali: P uguale a *pulcino*, Q uguale a *quadrifoglio*.

1. Leggi i valori P e Q.
2. Ripeti quanto segue fino a quando hai trovato le prime lettere diverse:
3. Se la prima lettera di $P <$ della prima lettera Q:
4. Allora scrivi vero.
5. Altrimenti se la prima lettera di $P >$ della prima lettera di Q:
6. Allora scrivi falso.
7. Altrimenti le lettere sono uguali.
8. Togli da P e Q la prima lettera.

Progettare un algoritmo

Problema: calcolare il massimo di un insieme di numeri naturali di nome A.
Dati iniziali: A contiene i seguenti numeri 2, 10, 4, 25, 1.

1. Scegli un elemento come massimo provvisorio detto *max*
2. Per ogni elemento i-esimo dell'insieme A:
3. se $i > max$ eleggi i come nuovo massimo provvisorio
4. il risultato sará *max*

Notazione lineare

- É una rappresentazione letterale di un algoritmo.
- Sfrutta tre tipi di operazioni:
  1. ingresso e uscita (dati e risultati del problema);
  2. assegnamento ($P = A$ con il significato assegna il nome logico P al valore A);
  3. controllo (modificano la sequenza dell'esecuzione).

Caratteristiche

Vantaggi:

- Intuitivitá.
- Facilitá di scrittura.

Svantaggi:

- Ambiguitá.
- Ridondanza.
- Scarso rigore.

Esempio di notazione lineare

Problema: calcolare il prodotto di due numeri interi positivi A e B.
NOTA: l'esecutore conosce solo le operazioni di somma, sottrazione e confronto tra numeri.

1. Leggi A e B.
2. $P = A$.
3. se B é uguale a 1 allora vai all'istruzione 7.
4. $P = P + A$.
5. $B = B - 1$.
6. vai all'istruzione 3
7. scrivi P.
8. fine

Pseudocodice

- É una rappresentazione descrittiva delle azioni dell'algoritmo.
- Usa frasi rigorose anziché simboli grafici.
- Usa parole chiave scritte in maiuscolo.
- Usa operatori quali $\leftarrow$, $+$, -, /
- Usa identazione, ossia il rientro dei gruppi di azioni riferite a cicli o a strutture a scelta.

Esempio di pseudocodice

Dati: $m \geq 1$ e due numeri positivi ognuno contenente m cifre, $a_{m-1}\ a_{m-2}, \ldots, a_0$ e $b_{m-1}\ b_{m-2}, \ldots, b_0$

Trovare: $c_m c_{m-1}\ c_{m-2}, \ldots, c_0$, dove $c_m c_{m-1}\ c_{m-2}, \ldots, c_0 = (a_{m-1}\ a_{m-2}, \ldots, a_0) + (b_{m-1}\ b_{m-2}, \ldots, b_0)$

Algoritmo:

Passo 1 Imposta il valore di *riporto* a 0.

Passo 2 Imposta il valore di i a 0.

Passo 3 Finché il valore di i è minore o uguale a $m-1$, ripeti le istruzioni dei passi da 4 a 6.

Passo 4 Somma le due cifre a_i e b_i al valore corrente di *riporto* per ottenere c_i.

Passo 5 Se $c_i \geq 10$, allora riporta c_i a $(c_i - 10)$ e imposta il valore di *riporto* a 1; altrimenti, imposta il nuovo valore di *riporto* a 0.

Passo 6 Somma 1 a i, spostandoti di una colonna a sinistra.

Passo 7 Imposta c_m al valore di *riporto*.

Passo 8 Stampa la soluzione finale, $c_m c_{m-1}\ c_{m-2}, \ldots, c_0$.

Passo 9 Stop.

Argomenti

Categorie

Algoritmi di ricerca verificano la presenza di un valore in un insieme di dati.

Algoritmi di ordinamento scansionano i valori all'interno di un insieme in modo crescente o decrescente.

Algoritmo di ricerca

- Ha come scopo quello di trovare un elemento avente determinate caratteristiche all'interno di un insieme di dati.
- L'insieme puó essere ordinato o non ordinato.
- L'insieme puó essere ordinato in modo crescente o decrescente.
- La ricerca puó essere binaria o sequenziale.

Ricerca sequenziale o completa

- Consiste nella scansione sequenziale degli elementi di un insieme dal primo all'ultimo.
- Si interrompe quando:
  - il valore cercato é stato trovato;
  - non vi sono piú elementi da cercare;
  - oppure quando siamo sicuri che il valore non possa essere presente.
- Ha il vantaggio di poter essere applicata anche ad insiemi di dati non ordinati.
- Durante la ricerca, si effettuano confronti tra il valore da ricercare e gli elementi dell'insieme.

Ricerca sequenziale o completa

- Supponendo di voler cercare il valore x in un insieme di interi di nome v con n elementi.
- La funzione, che effettua la ricerca, puó restituire:
  - l'indice dell'elemento dell'insieme con valore x;
  - o -1 in caso di valore non trovato.

Ricerca binaria o logaritmica

- É simile al metodo usato per trovare una parola sul dizionario.
- Puó essere applicata solo ad insiemi di dati ordinati:
  - Anche il vocabolario é ordinato alfabeticamente.
- Inizia dall'elemento centrale e non dal primo:
  - Nel caso del vocabolario inizia a metá dizionario.
- Il valore ricercato viene confrontato con il valore dell'elemento dell'insieme preso in esame:
  - Se corrisponde, la ricerca termina indicando che l'elemento é stato trovato;
  - Se é inferiore, la ricerca viene ripetuta sugli elementi precedenti (ovvero sulla prima metá del dizionario), scartando quelli successivi;
  - Se invece é superiore, la ricerca viene ripetuta sugli elementi successivi (ovvero sulla seconda met del dizionario), scartando quelli precedenti;
  - Se tutti gli elementi sono stati scartati, la ricerca termina indicando che il valore non stato trovato.

Ricerca binaria o logaritmica

- Si presta ad una definizione ricorsiva.
- Ad ogni chiamata della funzione, si verifica se l'elemento ricercato si trova al centro dell'intervallo:
  - in tal caso la funzione termina con successo.
- In caso contrario, si modifica l'intervallo di ricerca e si effettua una nuova chiamata della funzione.
- Nel caso in cui l'intervallo di ricerca sia nullo, la ricorsione termina con insuccesso.

Confronto tra i due algoritmi di ricerca

- L'efficienza di un algoritmo si misura in base al numero di confronti effettuati che dipende da n, ossia la lunghezza dell'insieme.
- In generale interessa il caso medio.

Algoritmo	Caso peggiore	Caso medio	Caso migliore
Di Ricerca Sequenziale (RS)	n	$\frac{n}{2}$	n
Di Ricerca Binaria (RB)	$log_2(n)$	$log_2(n)$	1

Caso medio con $n = 1000$.
Efficienza: 500 con RS, 10 con RB.

Algoritmo di ordinamento

- Si basa su operazioni di scambio e confronto.
- Puó essere necessario scandire piú volte gli elementi dell'insieme.
- Puó seguire una delle seguenti classi:
  - Stupid Sort;
  - Selection Sort;
  - Bubble Sort.

Stupid Sort

- É particolarmente inefficiente.
- Consiste nel mischiare gli elementi dell'insieme, come si puó fare con un mazzo di carte.
- Controlla se é ordinato e, se non lo é, ricomincia da capo.
- É probabilistico.
- Arriva quasi sicuramente ad una conclusione per il teorema della scimmia instancabile:
  - ad ogni tentativo c'é una probabilitá di ottenere l'ordinamento giusto;
  - dato un numero illimitato di tentativi, infine dovrebbe avere successo.

Teorema della scimmia instancabile

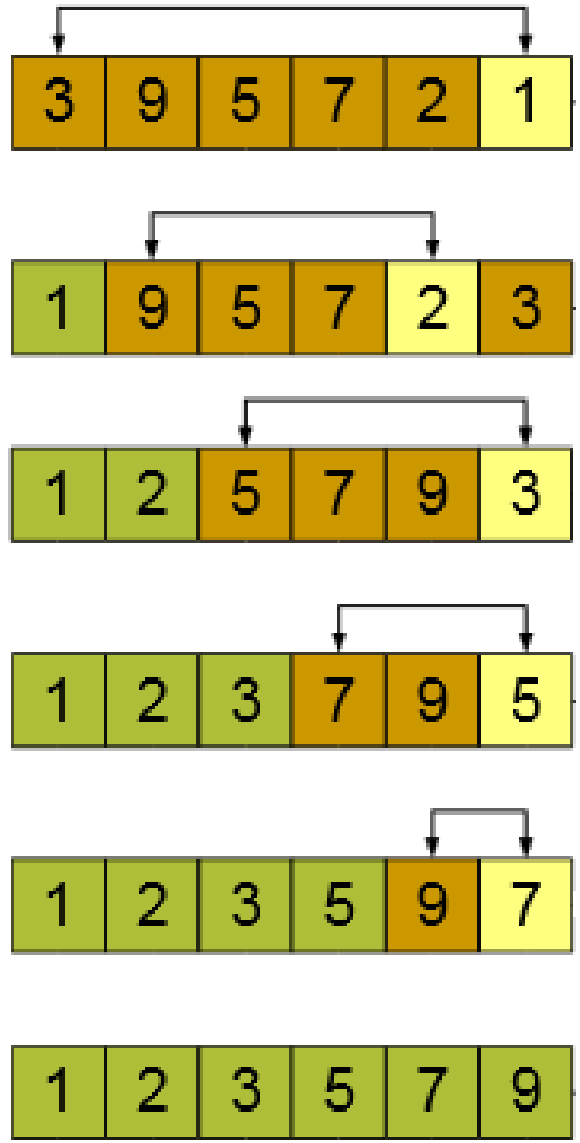
Il teorema della scimmia instancabile o teorema delle scimmie infinite afferma che una scimmia che prema a caso i tasti di una tastiera per un tempo infinitamente lungo quasi certamente riuscir a comporre qualsiasi opera letteraria conservata nella Biblioteca Nazionale di Francia [Borel, 1914].
Il termine scimmia é una metafora per una macchina teorica che produce una sequenza casuale di lettere ad infinitum.

Bobo Sort

- Variante meno efficace dello Stupid Sort.
- Consiste nel controllare se l'insieme é ordinato.
- Se non lo é, prende due elementi casualmente e li scambia (indipendentemente dal fatto che lo scambio aiuti l'ordinamento o meno).

Selection Sort

- É un algoritmo semplice e di selezione.
- Effettua piú scansioni dell'insieme.
- Al termine della prima scansione, il primo elemento dell'insieme contiene il valore minore (ottenuto selezionando l'elemento di valore piú basso nell'insieme).
- Tra i rimanenti $n-1$ elementi, l'algoritmo ricerca l'elemento minore e lo scambia con il secondo.
- Ad ogni scansione viene scambiato un elemento dell'insieme nella posizione dell'elemento piú grande considerato.
- Si procede fino allo scambio degli ultimi due elementi.



Bubble Sort

- É un algoritmo semplice e di scambio.
- Effettua la scansione dell'insieme elemento per elemento.
- Scambia i valori dei due elementi consecutivi o adiacenti, quando il primo é maggiore del secondo.
- Al termine della prima scansione, il primo elemento dell'insieme contiene il valore minore.
- L'algoritmo prosegue ordinando la parte successiva dell'insieme.
- L'insieme risulta ordinato dopo aver effettuato $n - 1$ scansioni con n numero degli elementi dell'insieme.

Bubble Sort

- Al termine della prima scansione, in genere l'insieme non risulta ordinato.
- É necessario procedere ad una nuova scansione e alla conseguente serie di eventuali scambi tra i valori di due elementi consecutivi.
- La ripetizione degli scambi tra elementi adiacenti fa salire verso l'alto gli elementi piú grandi o piú piccoli in base all'ordine con cui viene fatto l'ordinamento.
- É detto bubblesort (ordinamento a bolle) per analogia con le bolle d'aria nell'acqua che, essendo leggere, tendono a spostarsi verso l'alto.
- Risulta poco efficiente quando ci sono molti elementi disordinati.

Per ulteriori letture

 Émil Borel, *Le Hazard*, Alcan (1914), Paris.