

Algoritmi

Andrea Passerini
passerini@disi.unitn.it

Informatica

Un elaboratore o *computer* è una macchina *digitale, elettronica, automatica* capace di effettuare trasformazioni o elaborazioni sui dati

digitale l'informazione è rappresentata in forma numerica discreta

elettronica la logica di manipolazione e la memorizzazione sono implementate con tecnologie di tipo elettronico (piuttosto che di tipo meccanico)

automatica è in grado di eseguire una successione di operazioni in modo autonomo (cioè senza intervento di un operatore umano)

- le operazioni sono descritte sotto forma di un *programma*
- il programma e i dati su cui deve operare sono registrati in un dispositivo di *memoria*
- un dispositivo detto *unità di controllo* legge il programma e lo esegue su i dati
- questo modo di operare è detto *architettura alla Von Neumann*

Macchina universale

- Un programma è una descrizione delle operazioni che devono essere eseguite per risolvere una certa classe di problemi (e.g. trovare il maggiore tra due numeri)
- Un programma eseguito su dei dati in ingresso risolve lo specifico problema definito dai dati, nella classe di problemi trattati dal programma stesso (e.g. il maggiore tra 3 e 7 è 7)
- Un programma deve essere in una forma che sia rappresentabile nella memoria dell'elaboratore e comprensibile dall'elaboratore stesso.
- L'elaboratore è una *macchina universale* poiché può essere usato per risolvere qualsiasi problema la cui soluzione può essere descritta sotto forma di programma.

- Il termine algoritmo deriva dal nome di un matematico arabo al-Khuwarizmi (IX sec d.C.)
- Definizione: un algoritmo è una successione ordinata di istruzioni (o passi) che definiscono le operazioni da eseguire su dei dati per risolvere una classe di problemi
- Un programma è la descrizione di un algoritmo in una forma comprensibile (ed eseguibile) dall'elaboratore

Esempio: calcolo del M.C.D. di due numeri interi a, b positivi

- Come calcolare il M.C.D. senza provare tutti i divisori interi di a e b ?
- Supponiamo che $a > b$.
- Calcoliamo la divisione intera di a con b , ossia $a \times q + r = b$ con q ed r quoziente e resto della divisione.
- Se a è divisibile per b ($r = 0$), il M.C.D. è b :

$$M.C.D.(a, b) = b \quad \text{se } r = 0$$

- Altrimenti, il M.C.D. di a e b è anche divisore di r (e.g. $a=15, b=6, r=3$ che è il M.C.D.)

$$M.C.D.(a, b) = M.C.D.(b, r) \quad \text{se } r \neq 0$$

Esempio: algoritmo di Euclide per M.C.D

- 1 inizio dell'algoritmo;
- 2 acquisisci i valori di a e b dall'esterno;
- 3 se $b > a$ scambia tra loro i valori di a e b ;
- 4 calcola il resto r della divisione intera di a con b ;
- 5 se $r = 0$ allora comunica b all'esterno e vai al passo 7;
- 6 se $r \neq 0$ sostituisci il valore di a con il valore di b ed il valore di b con il valore di r e vai al passo 4;
- 7 fine dell'algoritmo.

Proprietà degli algoritmi

Esistono dei precisi requisiti che devono essere soddisfatti affinché un determinato elenco di istruzioni possa essere considerato un algoritmo:

Finitezza Ogni algoritmo deve essere finito, ossia ogni istruzione deve essere eseguita in un intervallo finito di tempo ed un numero finito di volte.

Generalità Ogni algoritmo deve fornire soluzione per tutti i problemi appartenenti ad una data classe, ed essere applicabile a tutti i dati appartenenti al suo *insieme di definizione* o *dominio* producendo risultati che appartengono al suo *insieme di arrivo* o *codominio*.

Non ambiguità Devono essere definiti in modo univoco e non ambiguo i passi successivi da eseguire per ottenere i risultati voluti, evitando paradossi e contraddizioni.

- Il linguaggio naturale non è adatto a descrivere gli algoritmi in quanto *ambiguo* e *ridondante*.
- Le proposizioni contenute in un algoritmo sono costituite da due componenti fondamentali:
 - istruzioni la descrizione delle operazioni da eseguire
 - dati la descrizione degli oggetti su cui eseguire le operazioni

Variabili

sono costituite da coppie $\langle nome, valore \rangle$ (e.g. $\langle x, 10 \rangle$)

- Il nome della variabile non cambia e la identifica all'interno dell'algoritmo.
- Il valore deve appartenere ad un insieme detto *insieme di definizione*, che è l'insieme di tutti i possibili valori che la variabile può assumere.
- All'insieme di definizione sono associate delle *regole di manipolazione* per operare sui suoi elementi (e.g. le operazioni aritmetiche su interi positivi).

Vettori

sono variabili contenenti insiemi di valori (detti *elementi* del vettore)

- Il numero di elementi contenuti in un vettore è detto *dimensione* del vettore.
- Ogni elemento è indicato tramite un indice che ne identifica la posizione all'interno del vettore (e.g. $V(1)$ è il primo elemento del vettore V , $V(n)$ l'ultimo se n è la dimensione del vettore).

Matrici

sono generalizzazioni dei vettori ad un numero arbitrario di dimensioni.

- Nelle matrici a due dimensioni gli elementi vengono individuati tramite due indici, detti *riga* e *colonna*.
- Ad esempio $M(3,4)$ indica l'elemento contenuto nella terza riga e nella quarta colonna della matrice M .

Istruzione di assegnazione

imposta il valore corrente di una variabile. Si indica con

nome di variabile \leftarrow espressione

- espressione è una stringa di costanti, nomi di variabili ed operazioni.
- nome di variabile indica la variabile il cui valore deve essere impostato al risultato di espressione
- I valori delle variabili contenute nell'espressione rimangono inalterati.
- Esempi:

$x \leftarrow 10$

$x \leftarrow y \times 2$

Tipi di istruzione

istruzioni operative producono risultati una volta eseguite (e.g. l'istruzione di assegnazione)

istruzioni di controllo controllano il verificarsi o meno di condizioni specificate e determinano quali istruzioni eseguire in base al risultato del controllo.

istruzione di inizio esecuzione indica quale istruzione dell'algoritmo deve essere eseguita per prima

istruzione di fine esecuzione indica la fine dell'esecuzione dell'algoritmo

istruzioni di **ingresso/uscita** indicano una trasmissione di dati o messaggi tra l'algoritmo e l'ambiente esterno. Si dividono in:

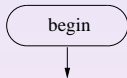
istruzioni di **ingresso (lettura)** l'algoritmo riceve dei dati dall'ambiente esterno

istruzioni di **uscita (scrittura)** l'algoritmo trasmette dei dati all'ambiente esterno

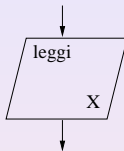
Diagrammi a blocchi (*flow chart*)

- E' un linguaggio formale di tipo grafico per rappresentare gli algoritmi
- Attraverso il diagramma a blocchi si può indicare l'ordine di esecuzione delle istruzioni.
- Un particolare simbolo grafico detto *blocco elementare* è associato ad ogni tipo di istruzione elementare.
- I blocchi sono collegati tra loro tramite frecce che indicano il susseguirsi delle istruzioni

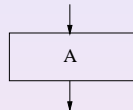
Blocchi elementari



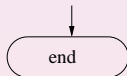
blocco iniziale



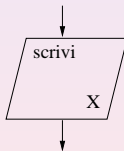
blocco di lettura



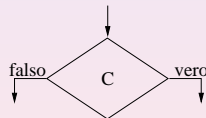
blocco azione



blocco finale



blocco di scrittura

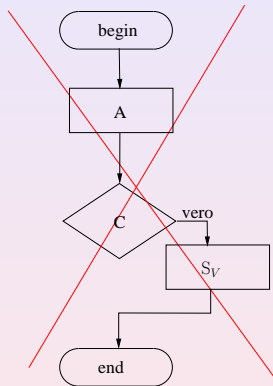
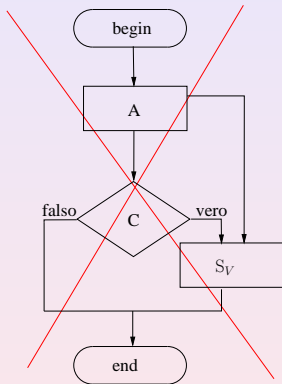
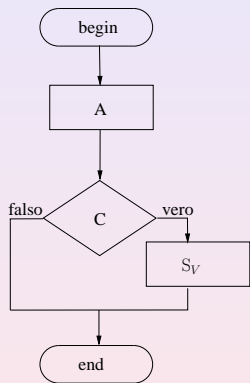


blocco di controllo

- Un diagramma a blocchi è un insieme di blocchi costituito da:
 - un blocco iniziale
 - un numero finito $n \geq 1$ di blocchi azione e/o blocchi di lettura/scrittura
 - un numero finito $m \geq 0$ di blocchi di controllo
 - un blocco finale
- ciascun blocco elementare soddisfa le *condizioni di validità*

- ciascun blocco azione, lettura/scrittura ha una sola freccia entrante e una sola freccia uscente
- ciascun blocco di controllo ha una sola freccia entrante e due uscenti
- ciascuna freccia entra in un blocco o si innesta su una altra freccia
- ciascun blocco è raggiungibile dal blocco iniziale
- il blocco finale è raggiungibile da qualsiasi altro blocco

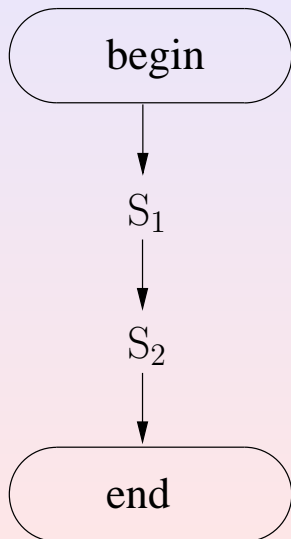
Esempi



- Analisi volta alla stesura di descrizioni di algoritmi tramite diagrammi a blocchi strutturati, più facilmente comprensibili e modificabili
- Un diagramma a blocchi è strutturato se:
 - è costituito da un blocco iniziale, un blocco di azione o lettura/scrittura ed un blocco finale
 - è costituito da blocchi collegati tramite i seguenti schemi di flusso strutturato:
 - schema di sequenza
 - schema di selezione
 - schema di iterazione

Schema di sequenza

Due o più schemi di flusso sono eseguiti uno di seguito all'altro

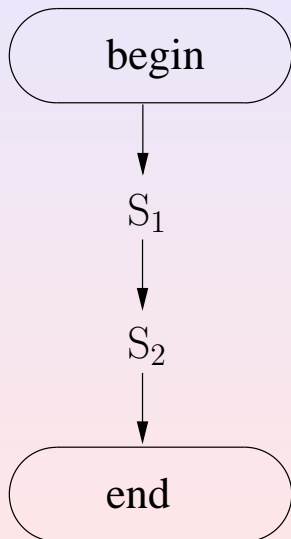


Nota

Lo schema è strutturato se e solo se sono strutturati gli schemi S_1 ed S_2 .

Schema di sequenza

Due o più schemi di flusso sono eseguiti uno di seguito all'altro



Nota

Lo schema è strutturato se e solo se sono strutturati gli schemi S_1 ed S_2 .

Contiene un blocco di controllo che opera in uno dei seguenti modi:

- Caso 1: subordina l'esecuzione di uno schema di flusso S_V alla soddisfazione della condizione
- Caso 2: permette di scegliere quale schema di flusso eseguire tra due schemi indicati S_V ed S_F .

Nota

Lo schema è strutturato se e solo se sono strutturati gli schemi S_V ed S_F .

Contiene un blocco di controllo che opera in uno dei seguenti modi:

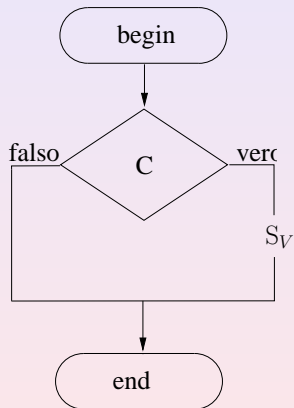
- Caso 1: subordina l'esecuzione di uno schema di flusso S_V alla soddisfazione della condizione
- Caso 2: permette di scegliere quale schema di flusso eseguire tra due schemi indicati S_V ed S_F .

Nota

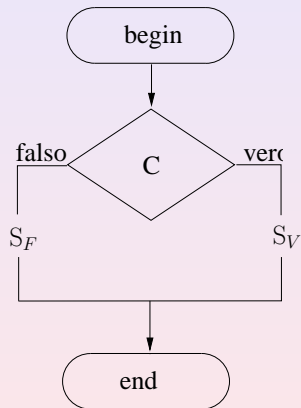
Lo schema è strutturato se e solo se sono strutturati gli schemi S_V ed S_F .

Schema di selezione

Caso 1



Caso 2



Schema di iterazione

Permette di eseguire più volte un determinato schema di flusso S .

Iterazione per vero lo schema S viene eseguito finché una condizione C è vera

Iterazione per falso lo schema S viene eseguito finché una condizione C è falsa

Nota

Lo schema è strutturato se e solo se è strutturato lo schema S .

Schema di iterazione

Permette di eseguire più volte un determinato schema di flusso S .

Iterazione per vero lo schema S viene eseguito finché una condizione C è vera

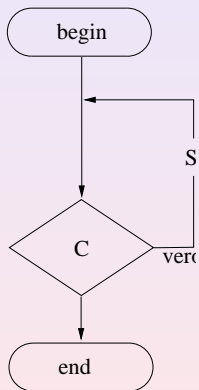
Iterazione per falso lo schema S viene eseguito finché una condizione C è falsa

Nota

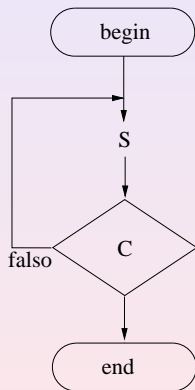
Lo schema è strutturato se e solo se è strutturato lo schema S .

Schema di iterazione

Iterazione per vero



Iterazione per falso



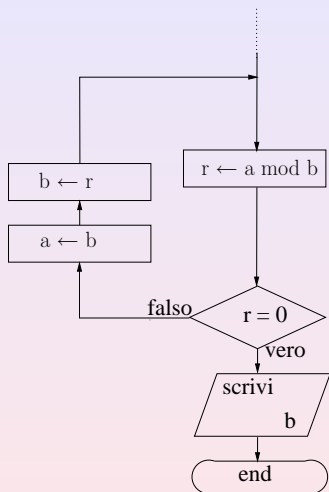
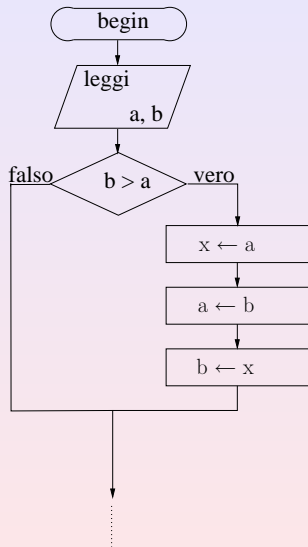
Schema di iterazione

- Detto anche *ciclo* o *loop*.
- Si distingue anche per la posizione della *condizione di fine ciclo*:
 - controllo in testa** la condizione viene verificata *prima* di eseguire le istruzioni della iterazione (che possono anche non essere mai eseguite)
 - controllo in coda** la condizione viene verificata *dopo* aver eseguito le istruzioni della iterazione (che vengono eseguite sempre almeno una volta).
- Viene in genere preceduto da una sequenza di *istruzioni di inializzazione* che assegnano un valore iniziale ad alcune variabili utilizzate nel ciclo.

Schema di iterazione

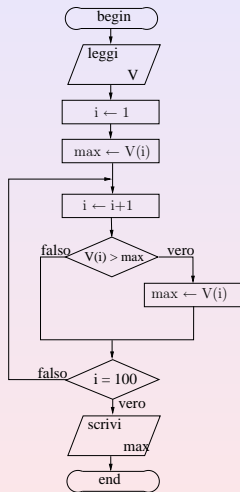
- Un ciclo è detto *enumerativo* quando è noto a priori il numero di volte che deve essere eseguito
 - si usa la tecnica del *contatore* per controllarne l'esecuzione: si usa cioè una variabile detta contatore del ciclo che viene incrementata (o decrementata) fino a raggiungere un valore prefessato
- Un ciclo è *indefinito* quando non è noto a priori il numero di volte che deve essere eseguito
 - questo accade quando la condizione di fine ciclo dipende dal valore di una o più variabili che o dipendono dall'interazione con l'esterno o vengono modificate all'interno dell'iterazione in modo complesso

Esempio: algoritmo di Euclide

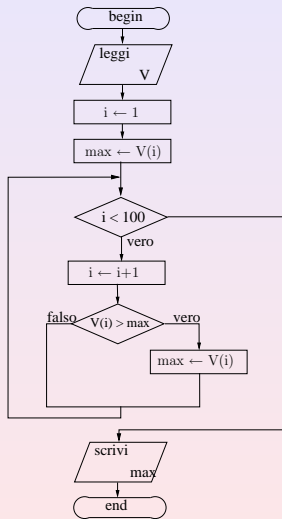


Esempio: massimo in un vettore di 100 numeri

iterazione per falso



iterazione per vero



Ogni diagramma a blocchi non strutturato è sempre trasformabile in un diagramma a blocchi strutturato ad esso equivalente

- Due diagrammi sono equivalenti se partendo dagli stessi dati iniziali producono gli stessi risultati