Informatica Generale

Introduzione

Gli algoritmi

- Analisi e programmazione
- Gli algoritmi
  - Proprietà ed esempi
  - Costanti e variabili, assegnazione, istruzioni, proposizioni e predicati
  - Vettori e matrici
  - I diagrammi a blocchi
  - Analisi strutturata
  - Gli algoritmi iterativi
  - La pseudocodifica
  - Gli algoritmi ricorsivi

Analisi e programmazione

Analisi e programmazione - 1

- Tramite un elaboratore si possono risolvere problemi di varia natura: emissione di certificati anagrafici, gestione dei c/c di un istituto di credito, prenotazioni ferroviarie…
- Il problema deve essere formulato in modo opportuno, perché sia possibile utilizzare un elaboratore per la sua soluzione
- Per analisi e programmazione si intende l'insieme delle attività preliminari atte a risolvere problemi utilizzando un elaboratore, dalla formulazione del problema fino alla predisposizione dell'elaboratore
  - Scopo dell'analisi ⇨ definire un algoritmo
  - Scopo della programmazione ⇨ definire un programma

Analisi e programmazione - 2

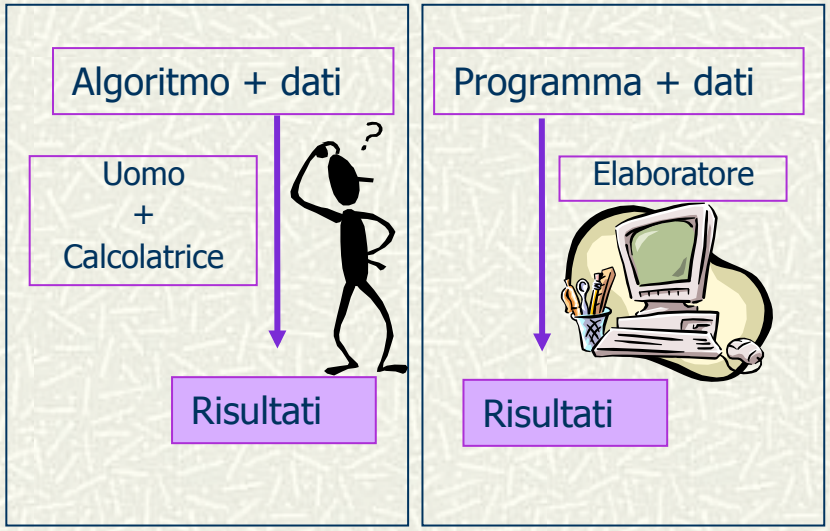
- Algoritmo — elenco finito di istruzioni, che specificano le operazioni eseguendo le quali si risolve una classe di problemi
  - Un problema della classe viene risolto con l'algoritmo sui dati che lo caratterizzano
  - **Un algoritmo non può essere eseguito direttamente dall'elaboratore**
- Programma — codifica dell'algoritmo in un linguaggio compilabile in linguaggio macchina
- Linguaggio di programmazione — linguaggio rigoroso che permette la formalizzazione di un algoritmo in un programma

Analisi e programmazione - 3

- **Esempio**

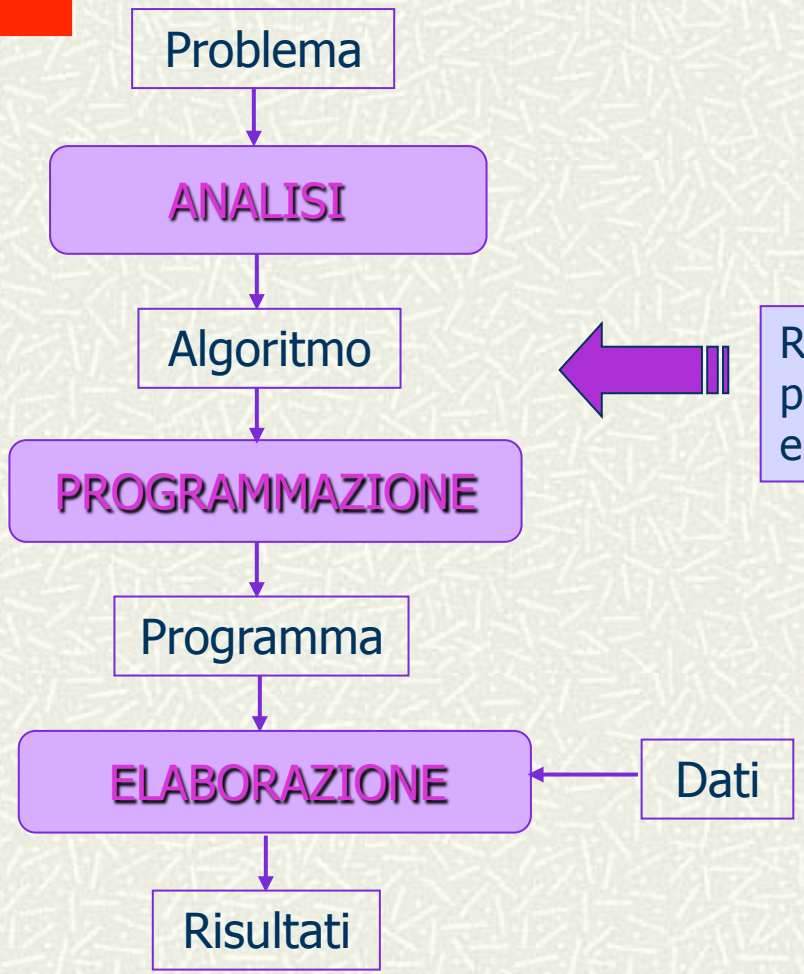
  Problema: Effettuare un accredito su un c/c bancario

  Soluzione: Utilizzare un programma che serva per predisporre il calcolatore all'accredito di una qualunque cifra su un qualunque c/c; cifra da accreditare e numero di c/c sono i dati caratteristici del problema



Analogie tra le azioni che devono essere eseguite da un operatore umano e, in modo automatico, tramite un elaboratore

Le fasi del procedimento di analisi e programmazione

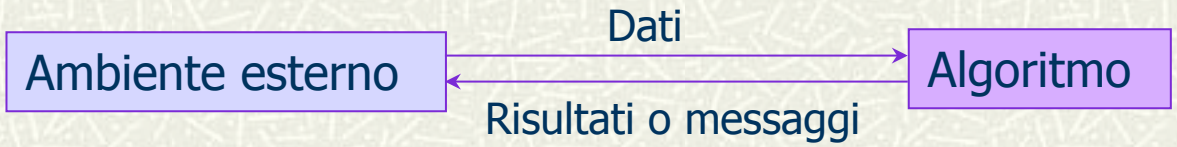


Relazioni tra problema, analisi, algoritmo, programmazione, programma, dati ed elaborazione

Gli algoritmi

Definizione di algoritmo

- ***Algoritmo*** deriva dal nome del matematico arabo *Al Khuwarizmi*, vissuto nel IX secolo d.C.
- Un algoritmo è una successione di **istruzioni** o **passi** che definiscono le operazioni da eseguire sui dati per ottenere i risultati; un algoritmo fornisce la soluzione ad una **classe di problemi**
- Lo **schema di esecuzione** di un algoritmo specifica che i passi devono essere eseguiti in sequenza, salvo diversa indicazione
- Ogni algoritmo è concepito per interagire con l'ambiente esterno per acquisire dati e comunicare messaggi o risultati; i dati su cui opera un'istruzione sono forniti dall'esterno o sono frutto di istruzioni eseguite in precedenza



Esempio: Il gioco dell'undici

- **Problema**: Undici fiammiferi sono disposti su un tavolo: il primo giocatore (A) può raccogliere da 1 a 3 fiammiferi, il secondo (B) ne raccoglie a sua volta 1, 2 o 3; i giocatori alternano le loro mosse finché sul tavolo non ci sono più fiammiferi; il giocatore che è costretto a raccogliere l'ultimo fiammifero è il perdente
- **Algoritmo**: Strategia vincente per il giocatore A che gioca per primo
  - prima mossa: A raccoglie 2 fiammiferi
  - mosse successive: se B raccoglie k fiammiferi ($k \leq 3$), allora A raccoglie 4-k fiammiferi

Esempio: Il gioco dell'undici

Topolino perde!

Esempio: Ordinamento di un mazzo di carte

- **Problema**: Sia dato un mazzo da 40 carte da ordinare in modo che le cuori precedano le quadri, che a loro volta precedono fiori e picche; le carte di uno stesso seme sono ordinate dall'asso al re
- **Algoritmo**:
  - Si suddivida il mazzo in 4 mazzetti, ciascuno costituito da tutte le carte dello stesso seme
  - Si ordinino le carte di ciascun mazzetto dall'asso al re
  - Si prendano nell'ordine i mazzetti delle cuori, quadri, fiori e picche

Esempio: Ricerca in un mazzo di chiavi

- **Problema**: Si vuole ricercare, all'interno di un mazzo di chiavi, quella che apre un dato lucchetto
- **Algoritmo**:

1) Si seleziona una chiave dal mazzo e la si marca con un pennarello
2) Si tenta di aprire il lucchetto con la chiave appena marcata; se funziona, si va al passo 4)
3) Altrimenti, si controlla la chiave successiva
   i. Se non è marcata, la si marca e si torna al passo 2)
   ii. Viceversa, si prende atto che nel mazzo non è presente la chiave che apre il lucchetto
4) Fine della ricerca

Esempio: Radici delle equazioni di 2° grado

- **Problema**: Calcolo delle radici reali di $ax^2+bx+c=0$
- **Algoritmo**:
  1) Acquisire i coefficienti a,b,c
  2) Calcolare $\Delta = b^2-4ac$
  3) Se $\Delta<0$ non esistono radici reali, eseguire l'istruzione 7)
  4) Se $\Delta=0$, $x_1=x_2=-b/2a$, poi eseguire l'istruzione 6)
  5) $x_1=(-b+\sqrt{\Delta})/2a$, $x_2=(-b-\sqrt{\Delta})/2a$
  6) Comunicare i valori x_1, x_2
  7) Fine

Esempio: Calcolo del M.C.D. - 1

- **Problema**: Calcolare il M.C.D. di due interi a,b, con a>b
- **Algoritmo**: Formalizzato da Euclide nel 300 a.C., si basa sul fatto che ogni divisore comune ad a e b è anche divisore del resto r della divisione intera di a per b, quando a>b e r≠0; se r=0, b è il M.C.D.

$$MCD(a,b) = MCD(b,r), \text{ se } r \neq 0$$

$$MCD(a,b)=b, \text{ se } r=0$$

- **Nota**

  L'algoritmo garantisce la determinazione del M.C.D. senza il calcolo di tutti i divisori di a e b

Esempio: Calcolo del M.C.D. - 2

1) Acquisire i valori di a e b
2) Se b>a, scambiare i valori di a e b
3) Calcolare il resto r della divisione intera di a per b
4) Se r=0, MCD(a,b)=b; comunicare il risultato all'esterno; eseguire l'istruzione 6)
5) Se r≠0, sostituire il valore di a con il valore di b ed il valore di b con il valore di r; tornare al passo 3)
6) Fine

Proprietà degli algoritmi

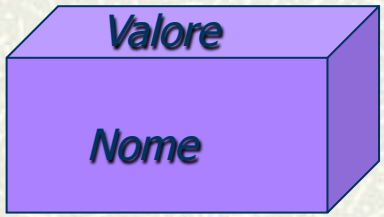
- Affinché una "*ricetta*", un elenco di istruzioni, possa essere considerato un algoritmo, devono essere soddisfatti i seguenti requisiti:
  - Finitezza: ogni algoritmo deve essere finito, cioè ogni singola istruzione deve poter essere eseguita in tempo finito ed un numero finito di volte
  - Generalità: ogni algoritmo deve fornire la soluzione per una classe di problemi; deve pertanto essere applicabile a qualsiasi insieme di dati appartenenti all'**insieme di definizione** o **dominio dell'algoritmo** e deve produrre risultati che appartengano all'**insieme di arrivo** o **codominio**
  - Non ambiguità: devono essere definiti in modo univoco i passi successivi da eseguire; devono essere evitati paradossi, contraddizioni ed ambiguità; il significato di ogni istruzione deve essere univoco per chiunque esegua l'algoritmo

Algoritmi

- Un algoritmo deve poter essere eseguito da chiunque, senza che l'esecutore sia stato necessariamente coinvolto nell'analisi del problema o nella descrizione dell'algoritmo
- Gli algoritmi devono essere formalizzati per mezzo di appositi linguaggi, dotati di strutture linguistiche che garantiscano precisione e sintesi
- I linguaggi naturali non soddisfano questi requisiti, infatti...
  - ...sono ambigui: la stessa parola può assumere significati diversi in contesti differenti (***pesca*** è un frutto o un'attività sportiva)
  - ...sono ridondanti: lo stesso concetto può essere espresso in molti modi diversi, ad esempio "somma 2 a 3", "calcola 2+3", "esegui l'addizione tra 2 e 3"

Costanti e variabili - 1

- I dati su cui opera un algoritmo sono costanti e variabili
  - Un dato è costante quando il suo valore non può essere aggiornato durante l'esecuzione dell'algoritmo o per esecuzioni successive
  - Una variabile è una coppia <*nome*, *valore*>: può essere immaginata come una scatola sulla quale è scritto un nome e che può contenere un valore



Rappresentazione di una variabile

Costanti e variabili - 2

- Il valore di una variabile deve appartenere all'**insieme di definizione**, su cui si opera mediante regole opportune, specifiche dell'insieme
- Data una variabile <x,v>, x è il **nome** della variabile e v è il suo **valore attuale**; le variabili sono indeterminate in fase di definizione dell'algoritmo, ma corrispondono a valori specifici durante ogni esecuzione
- **Esempio**: Nell'algoritmo di risoluzione delle equazioni di 2° grado, a, b, c non corrispondono a nessun valore finché non si esegue l'algoritmo per trovare le soluzioni di una data equazione, ad esempio $x^2-9x-4=0$; in fase di esecuzione, $a=1$, $b=-9$, $c=-4$; nell'istruzione $\Delta=b^2-4ac$, Δ è la variabile che contiene il valore del discriminante

Assegnazione - 1

- L'istruzione di assegnazione definisce il valore attuale di una variabile, che resta inalterato fino all'assegnazione successiva
- L'assegnazione si rappresenta con il simbolo "←":

  nome di variabile ← espressione

  che si legge *assegna alla variabile "nome di variabile" il valore di "espressione"*; l'espressione a destra di ← è costituita da variabili, costanti e operatori
- L'assegnazione viene così eseguita:
  a) si valuta l'espressione a destra di ←, sostituendo ai nomi di variabile i loro valori attuali; il risultato deve appartenere all'insieme di definizione della variabile a sinistra di ←
  b) il valore calcolato diventa il nuovo valore della variabile il cui nome appare a sinistra di ←

Assegnazione - 2

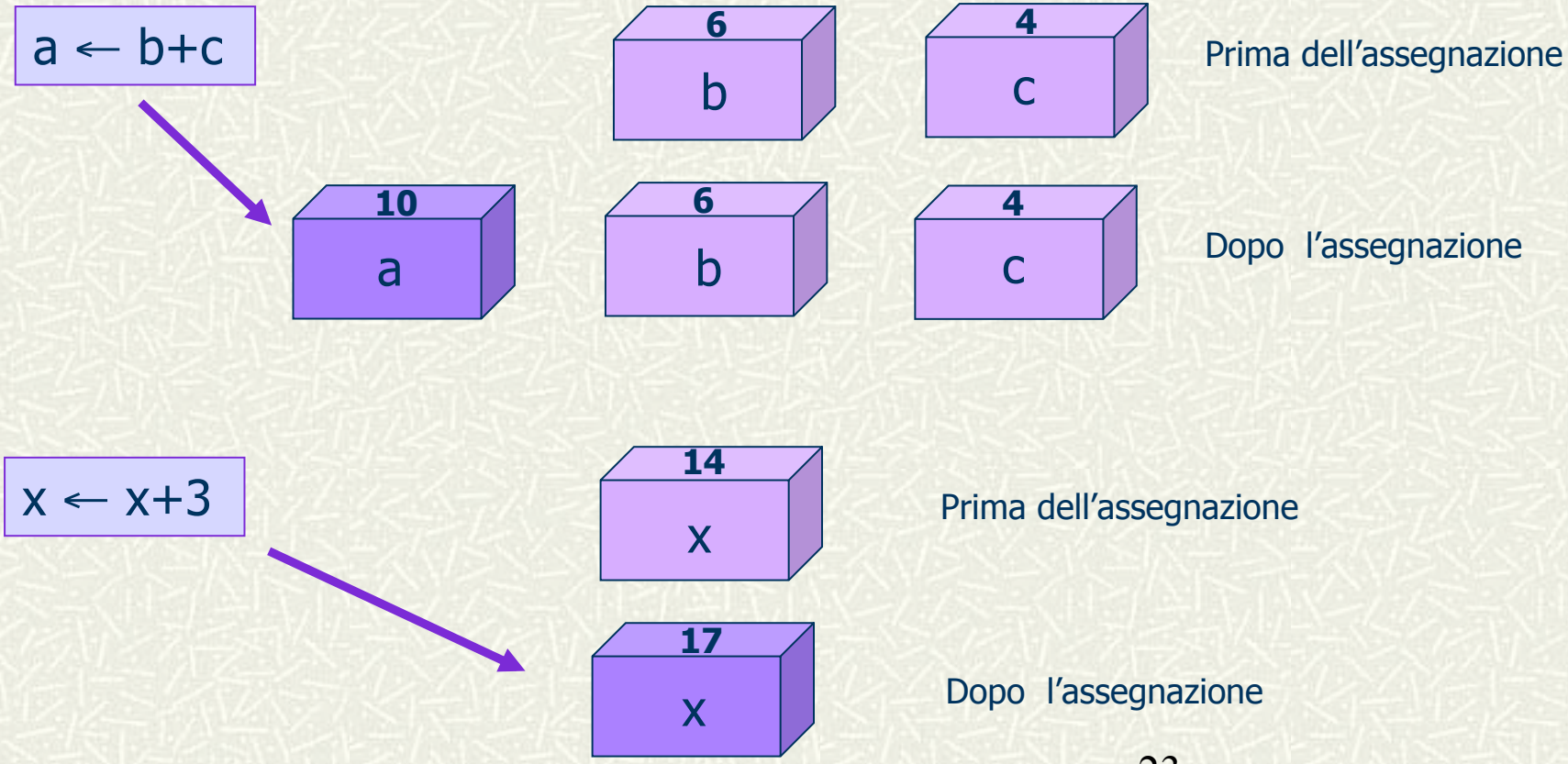
- **I nomi delle variabili possono essere scelti in modo arbitrario, ma è opportuno selezionare nomi significativi del contenuto della variabile**
- È necessario rispettare la regola dell'ordinamento:

  Quando una variabile appare a destra di ← in una assegnazione deve essere già istanziata

  cioè le deve essere già stato assegnato un valore

Assegnazione - 3

- Esempi



Le istruzioni - 1

- Istruzioni operative, che producono risultati
- Istruzioni di controllo, che controllano il verificarsi di condizioni specificate e, in base al risultato del controllo, determinano il flusso di istruzioni da eseguire
- Istruzioni di salto, che alterano il normale flusso di esecuzione sequenziale delle istruzioni di un algoritmo, specificando quale sia la successiva istruzione da eseguire
  - nelle istruzioni di salto condizionato, l'effettiva esecuzione del salto è legata al verificarsi di una condizione specificata
  - l'istruzione di salto incondizionato produce sempre un salto
- Istruzioni di ingresso/uscita, che specificano come debba essere effettuata una trasmissione di dati o messaggi fra l'algoritmo e l'ambiente esterno
- Istruzioni di inizio/fine esecuzione, che indicano l'inizio/la fine dell'algoritmo

Le istruzioni - 2

- **Esempio: Calcolo delle radici di equazioni di 2° grado**
  a) "acquisire i coefficienti a, b, c" è un'istruzione di lettura (ingresso)
  b) "calcolare $\Delta=b^2-4ac$" è un'istruzione operativa
  c) "se $\Delta=0$, $x_1=x_2=-b/2a$" è un'istruzione di controllo: l'istruzione di assegnazione $x_1=x_2=-b/2a$ viene eseguita solo se $\Delta=0$
  d) "comunicare i valori x_1, x_2" è un'istruzione di scrittura (uscita)
  e) "eseguire l'istruzione 6)" è un'istruzione di salto incondizionato
  f) "se $\Delta<0$ eseguire l'istruzione 7)" è un'istruzione di salto condizionato, perché l'istruzione 7) è la prossima istruzione da eseguire solo se $\Delta<0$

Proposizioni e predicati - 1

- Una **proposizione** è un costrutto linguistico del quale si può asserire o negare la veridicità
- **Esempi**
  1) *"Roma è la capitale della Gran Bretagna"* **falsa**
  2) *"3 è un numero intero"* **vera**
- Il **valore di verità** di una proposizione è il suo essere vera o falsa
- Una proposizione è un **predicato** se il suo valore di verità dipende dall'istanziazione di alcune variabili
- **Esempi**
  1) *"la variabile età è minore di 30"*
  2) *"la variabile base è maggiore della variabile altezza"*

Proposizioni e predicati - 2

- La valutazione di un predicato è l'operazione che permette di determinare se il predicato è vero o falso, sostituendo alle variabili i loro valori attuali
- I valori vero e falso sono detti valori logici o booleani
- Proposizioni e predicati possono essere espressi concisamente per mezzo degli operatori relazionali:

= (uguale)	≠ (diverso)
> (maggiore)	< (minore)
≥ (maggiore o uguale)	≤ (minore o uguale)

- I predicati che contengono un solo operatore relazionale sono detti semplici

Proposizioni e predicati - 3

- Dato un predicato **p**, il predicato **not p**, detto opposto o negazione logica di **p**, ha i valori di verità opposti rispetto a **p**
- Dati due predicati **p** e **q**, la congiunzione logica **p and q** è un predicato vero solo quando **p** e **q** sono entrambi veri, e falso in tutti gli altri casi
- Dati due predicati **p** e **q**, la disgiunzione logica **p or q** è un predicato falso solo quando **p** e **q** sono entrambi falsi, e vero in tutti gli altri casi
- I predicati nei quali compare almeno un operatore logico, **not**, **and**, **or**, sono detti composti
- La tavola di verità di un predicato composto specifica il valore del predicato per ognuna delle possibili combinazioni dei suoi argomenti

Proposizioni e predicati - 4

- Esempio

not (base > altezza)

è vero solo quando il valore di base è minore o uguale del valore di altezza

età > 30 **and** età < 50

è vero solo quando il valore di età è compreso tra 30 e 50 (esclusi)

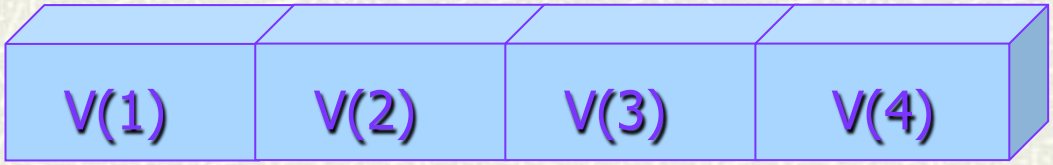
base > altezza **or** base > 100

è vero quando il valore di base è maggiore del valore di altezza, o quando il valore di base è maggiore di 100, o quando entrambe le condizioni sono verificate

Vettori e matrici - 1

- Le variabili definite come coppie <*nome*, *valore*> sono dette variabili **scalari**
- Una coppia <*nome, insieme di valori*> è una variabile **vettore** o **array** e può essere immaginata come un contenitore diviso in scomparti; ciascuno scomparto contiene un valore, detto **elemento** o **componente** del vettore
- Ciascuna componente è individuata dal nome del vettore, seguito dal relativo numero progressivo, racchiuso fra parentesi tonde: l'**indice** del vettore
- La **dimensione** di un vettore è il numero dei suoi elementi
- I vettori sono particolarmente utili per collezionare dati fra loro correlati, sui quali devono essere effettuate le stesse operazioni

Vettori e matrici - 2



Variabile vettoriale V, costituita dai 4 elementi V(1), V(2), V(3), V(4)

- L'utilizzo di variabili vettoriali, in un algoritmo, presuppone la dichiarazione esplicita della loro dimensione
- La dimensione del vettore costituisce un limite invalicabile per la selezione delle componenti del vettore
- Esempio: V(100) asserisce che il vettore V è costituito da 100 elementi; possono essere selezionati V(12),V(57),V(89), ma non V(121) o V(763), che non esistono

Vettori e matrici - 3

- Il concetto di **matrice** è un'estensione del concetto di vettore
- Una matrice è costituita da un insieme di valori, ciascuno dei quali viene individuato per mezzo della sua posizione, espressa da più indici
- Ad esempio, se una matrice M ha due dimensioni, i suoi elementi sono disposti su righe e colonne ed ogni suo elemento M(i,j) è individuato da due indici, con **i** indice di **riga** e **j** indice di **colonna**

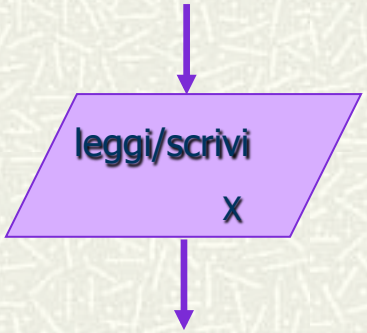
$$M = \begin{pmatrix} m_{11}\ m_{12} \ldots m_{1n} \\ \ldots\ \ldots\ \ldots \\ m_{q1}\ m_{q2} \ldots m_{qn} \end{pmatrix}$$

I diagrammi a blocchi - 1

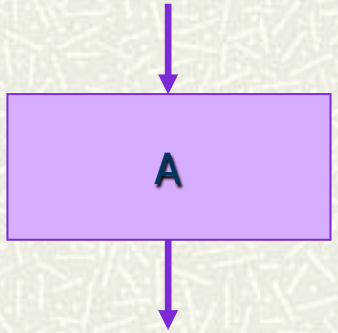
- Il linguaggio dei **diagrammi a blocchi** è un possibile formalismo per la descrizione di algoritmi
- Il diagramma a blocchi, o *flowchart*, è una rappresentazione grafica dell'algoritmo
- Un diagramma a blocchi descrive il **flusso** delle operazioni da eseguire per realizzare la trasformazione, definita nell'algoritmo, dai dati iniziali ai risultati
- Ogni istruzione dell'algoritmo viene rappresentata all'interno di un **blocco elementare**, la cui forma grafica è determinata dal tipo di istruzione
- I blocchi sono collegati tra loro da **linee di flusso**, munite di frecce, che indicano il susseguirsi di azioni elementari

I diagrammi a blocchi - 2

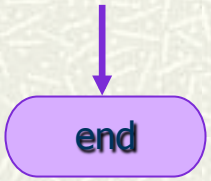
Blocco iniziale



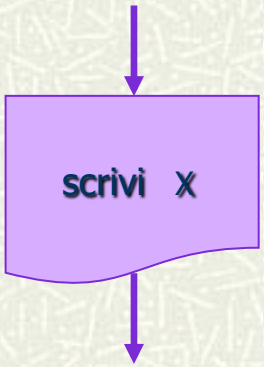
Blocco di lettura/scrittura



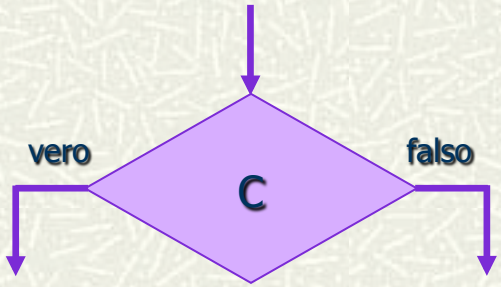
Blocco azione



Blocco finale



Blocco di scrittura



Blocco di controllo

Blocchi elementari

I diagrammi a blocchi - 3

- Un diagramma a blocchi è un insieme di blocchi elementari composto da:

  a) un blocco iniziale
  b) un blocco finale
  c) un numero finito n ($n \geq 1$) di blocchi di azione e/o di blocchi di lettura/scrittura
  d) un numero finito m ($m \geq 0$) di blocchi di controllo

I diagrammi a blocchi - 4

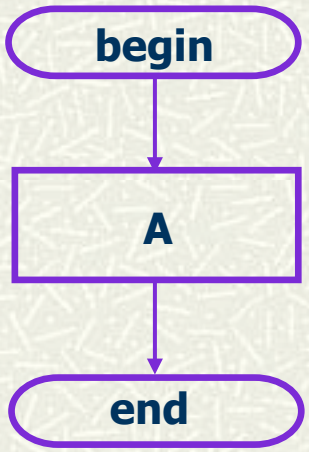
- L'insieme dei blocchi elementari che descrivono un algoritmo deve soddisfare le seguenti condizioni:
  - ciascun blocco di azione o di lettura/scrittura ha una sola freccia entrante ed una sola freccia uscente
  - ciascun blocco di controllo ha una sola freccia entrante e due frecce uscenti
  - ciascuna freccia entra in un blocco oppure si innesta in un'altra freccia
  - ciascun blocco è **raggiungibile** dal blocco iniziale
  - il blocco finale è **raggiungibile** da qualsiasi altro blocco
- Un blocco B è raggiungibile a partire da un blocco A se esiste una sequenza di blocchi $X_1, X_2, \ldots, X_n$, tali che $A=X_1$, $B=X_n$, e $\forall X_i$, $i=1, \ldots, n-1$, X_i è connesso con una freccia a X_{i+1}

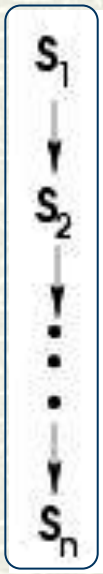
Analisi strutturata - 1

- I programmatori inesperti tendono ad "aggrovigliare" il programma introducendo numerosi salti privi di regole (***spaghetti programming***)
- L'analisi strutturata favorisce, viceversa, la descrizione di algoritmi facilmente documentabili e comprensibili
- I blocchi di un diagramma a blocchi strutturato sono collegati secondo i seguenti schemi di flusso:
  - Schema di sequenza – più schemi di flusso sono eseguiti in sequenza
  - Schema di selezione – un blocco di controllo subordina l'esecuzione di due possibili schemi di flusso al verificarsi di una condizione
  - Schema di iterazione – si itera l'esecuzione di un dato schema di flusso

Analisi strutturata - 2

- Ovvero: un **diagramma a blocchi strutturato** è un diagramma a blocchi nel quale gli schemi di flusso sono **strutturati**
- Uno schema di flusso è strutturato quando soddisfa una delle seguenti proprietà…
  1) …è uno schema elementare o uno schema di sequenza

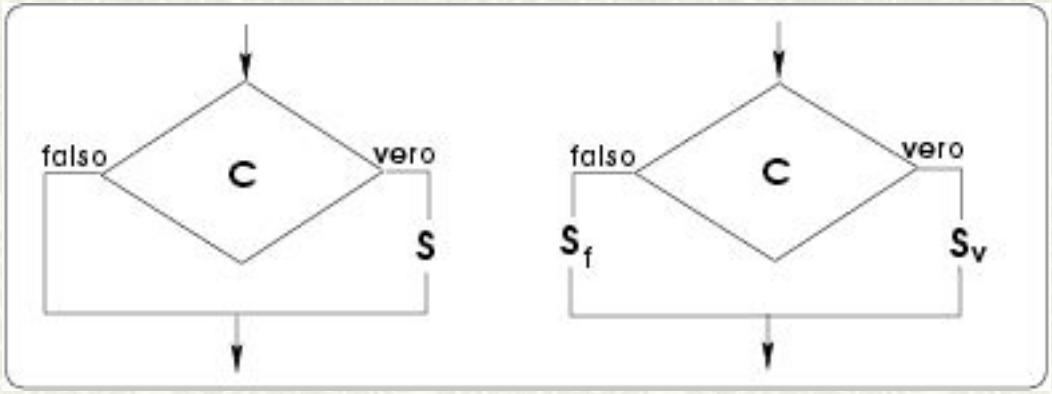




S_1, S_2,…, S_n schemi di flusso strutturati

Analisi strutturata - 3

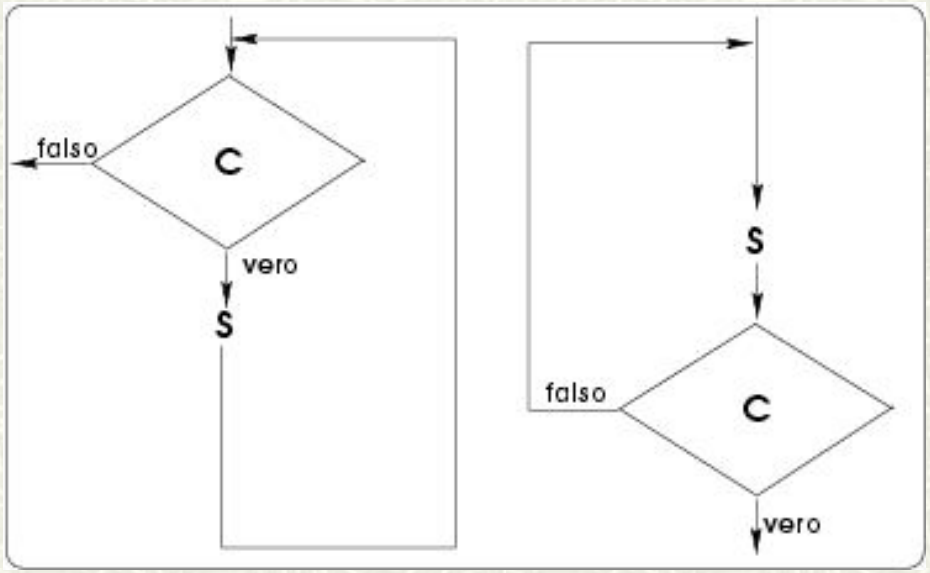
2) ...è uno schema di selezione



- Nel primo caso, lo schema S viene eseguito solo se la condizione C è vera; se C è falsa, non viene eseguita alcuna azione
- Nel secondo caso, viene eseguito solo uno dei due schemi S_v o S_f, in dipendenza del valore di verità della condizione

Analisi strutturata - 4

3) ...è uno schema di iterazione



- Nel primo caso, S può non venire mai eseguito, se la condizione C è subito falsa; nel secondo caso, S viene eseguito almeno una volta
- Quando lo schema S viene eseguito finché la condizione C si mantiene vera si parla di iterazione per vero; si ha un'iterazione per falso quando S viene eseguito finché C è falsa

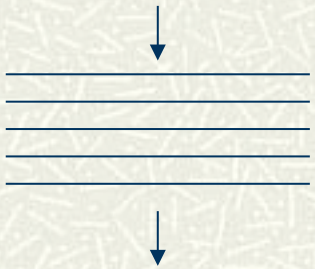
Analisi strutturata - 5

- Gli schemi di flusso sono **aperti** quando consentono una sola esecuzione di una sequenza di blocchi elementari, sono **chiusi** quando permettono più di un'esecuzione della sequenza di blocchi
- **Gli schemi di sequenza e di selezione sono aperti, lo schema di iterazione è chiuso**
- Ogni diagramma a blocchi non strutturato è trasformabile in un diagramma a blocchi strutturato equivalente
- Due diagrammi a blocchi sono **equivalenti** se, operando sugli stessi dati, producono gli stessi risultati
- L'uso dell'analisi strutturata garantisce:
  - facilità di comprensione e modifica dei diagrammi a blocchi
  - maggiore uniformità nella descrizione degli algoritmi

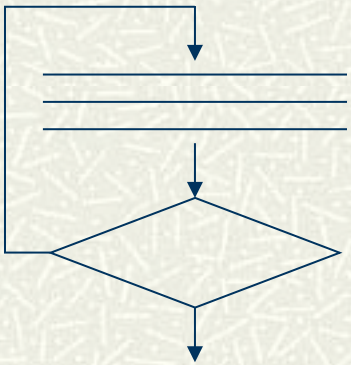
Analisi strutturata - 6

- Inoltre...
  - È stato dimostrato (teorema fondamentale della programmazione di Bohm-Jacopini, 1966) che ogni programma può essere codificato riferendosi esclusivamente ad un algoritmo strutturato e quindi attenendosi alle tre strutture fondamentali:

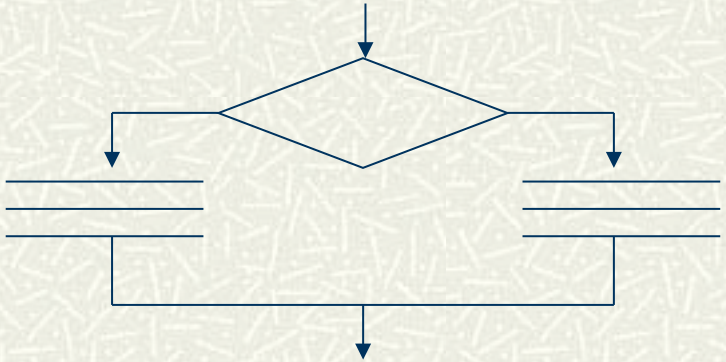
Sequenziale



Iterativa



Condizionale

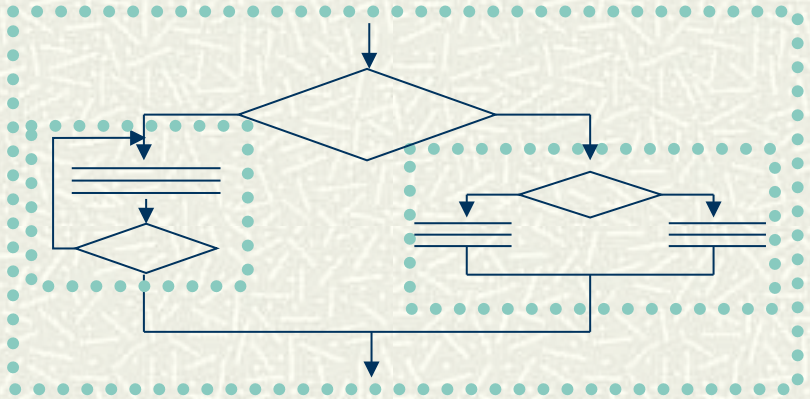


Analisi strutturata - 7

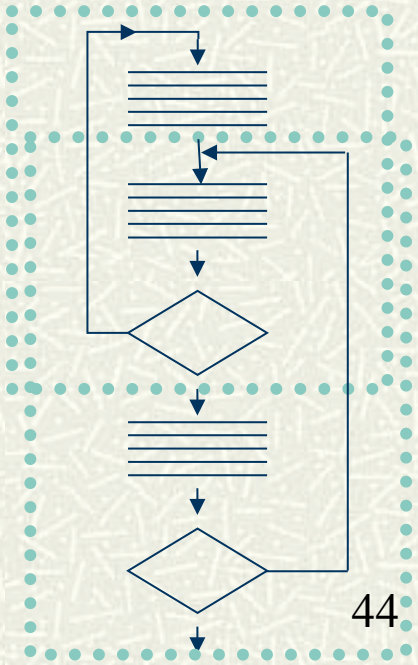
- Il teorema di Bohm-Jacopini ha un interesse soprattutto teorico, in quanto i linguaggi di programmazione tendono a dotarsi di più tipi di istruzioni, non sempre "rispettose" del teorema, ma utili per la realizzazione di programmi di facile scrittura e comprensione
- Il suo valore consiste nella capacità di fornire indicazioni generali per le attività di progettazione di nuovi linguaggi e di strategie di programmazione
- In effetti, esso ha contribuito alla critica dell'uso sconsiderato delle istruzioni *go to* e alla definizione delle linee guida della programmazione strutturata, sviluppate negli anni `70

Analisi strutturata - 8

- In un diagramma strutturato non apparirà mai una istruzione di salto incondizionato
- I tre schemi fondamentali possono essere ***concatenati,*** uno di seguito all'altro, o ***nidificati,*** uno dentro l'altro; non possono in nessun caso essere "intrecciati" o "accavallati"



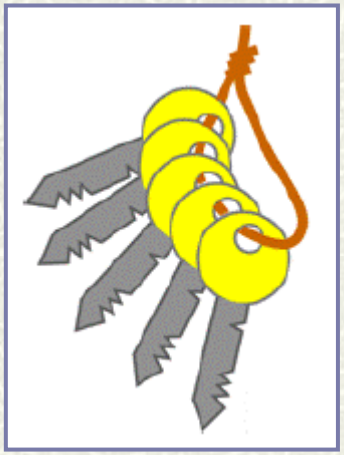
Corretto

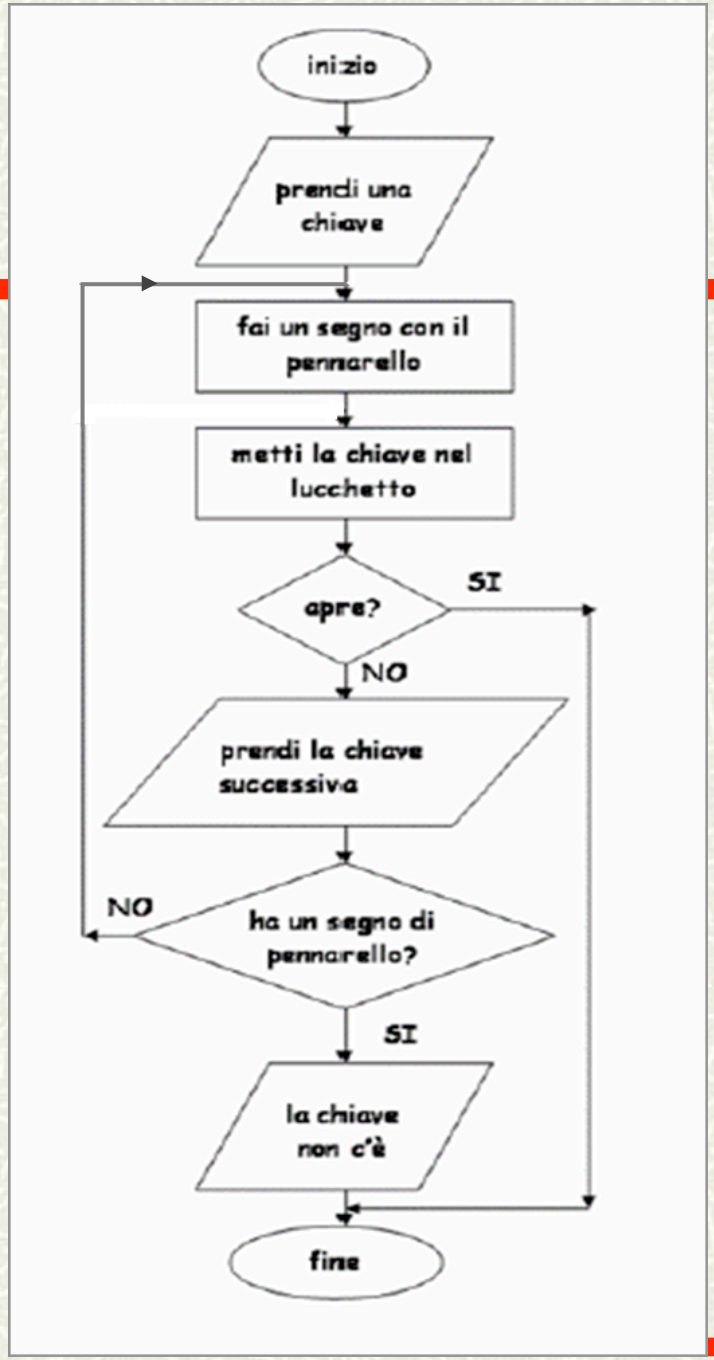


Sbagliato

Esempio

- Diagramma a blocchi per la selezione, in un mazzo di chiavi, di quella che apre un lucchetto



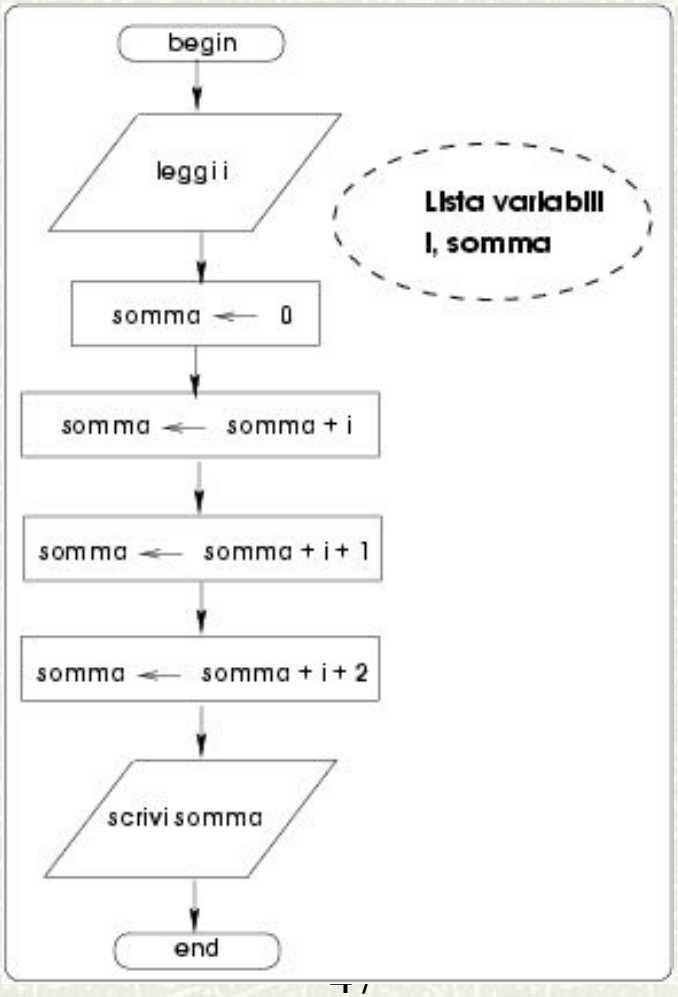


Esercizi

- Scrivere un algoritmo, e rappresentarlo tramite diagramma a blocchi, per la soluzione dei seguenti problemi:
  - calcolare l'area del triangolo
  - trovare il max di due numeri
  - moltiplicare due numeri (usando solo l'operazione di somma)
- Formalizzare, tramite diagramma a blocchi, l'algoritmo per…
  - …calcolare le radici reali di equazioni di 2° grado
  - …calcolare il M.C.D. di due numeri con il metodo di Euclide

Gli algoritmi iterativi - 1

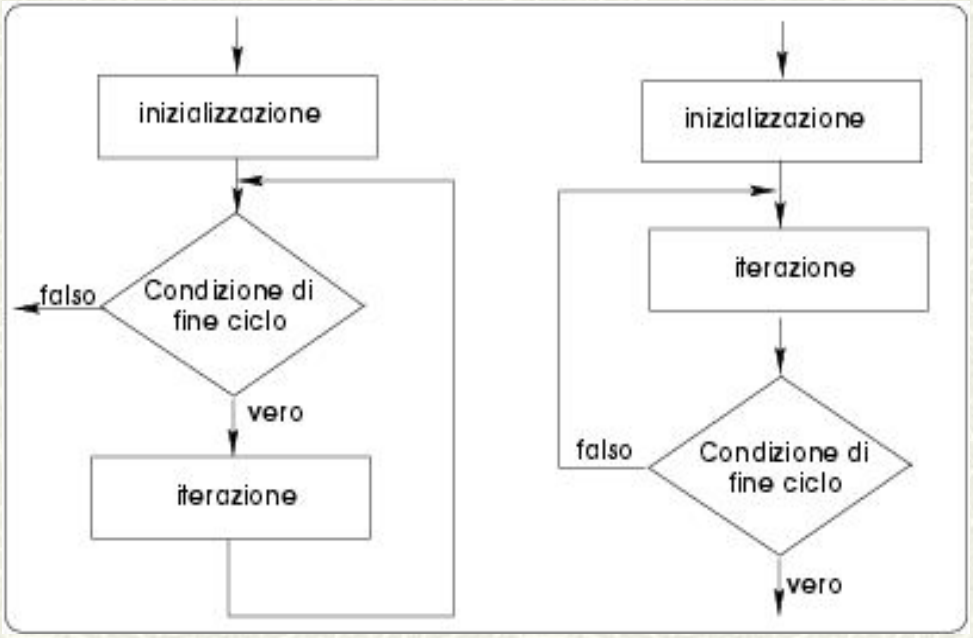
- **Problema**: Calcolare la somma di tre interi consecutivi



- **Note**:
  - La variabile **somma** è un contenitore di somme parziali, finché non si ottiene la somma totale richiesta
  - La soluzione del problema viene raggiunta eseguendo azioni simili per un numero opportuno di volte

Gli algoritmi iterativi - 2

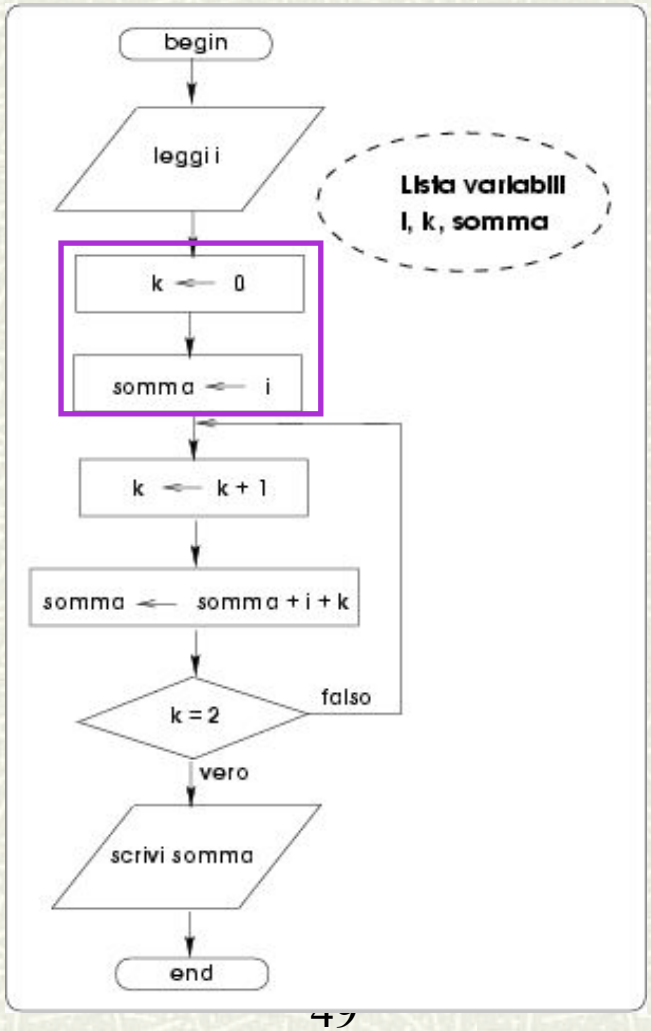
- Il ciclo o *loop* è uno schema di flusso per descrivere, in modo conciso, situazioni in cui un gruppo di operazioni deve essere ripetuto più volte
- La condizione di fine ciclo viene verificata ogni volta che si esegue il ciclo; se la condizione assume valore vero (falso), le istruzioni vengono reiterate, altrimenti si **esce dal ciclo**
- La condizione di fine ciclo può essere verificata prima o dopo l'esecuzione dell'iterazione
- Le istruzioni di inizializzazione assegnano valori iniziali ad alcune variabili (almeno a quella che controlla la condizione di fine ciclo)



Ciclo con controllo in testa Ciclo con controllo in coda

Gli algoritmi iterativi - 3

- **Problema**: Calcolare la somma di tre interi consecutivi



- **Note**:
  - La fase di inizializzazione riguarda la somma e l'indice del ciclo
  - Il controllo di fine ciclo viene effettuato in coda

Gli algoritmi iterativi - 4

- Un ciclo è definito quando è noto a priori quante volte deve essere eseguito; un ciclo definito è detto anche enumerativo
- Un contatore del ciclo tiene memoria di quante iterazioni sono state effettuate; può essere utilizzato in due modi:
  - **incremento del contatore**: il contatore viene inizializzato ad un valore minimo (ad es. 0 o 1) e incrementato ad ogni esecuzione del ciclo; si esce dal ciclo quando il valore del contatore eguaglia il numero di iterazioni richieste
  - **decremento del contatore**: il contatore viene inizializzato al numero di iterazioni richiesto e decrementato di uno ad ogni iterazione; si esce dal ciclo quando il valore del contatore raggiunge 0 (o 1)

Gli algoritmi iterativi - 5

- Un ciclo è **indefinito** quando non è possibile conoscere a priori quante volte verrà eseguito
- La condizione di fine ciclo controlla il valore di una o più variabili modificate da istruzioni che fanno parte dell'iterazione
- Comunque, un ciclo deve essere eseguito un numero finito di volte, cioè si deve sempre verificare la **terminazione** dell'esecuzione del ciclo

Gli algoritmi iterativi - 6

- Problema: Calcolo della media di un insieme di numeri; non è noto a priori quanti sono i numeri di cui deve essere calcolata la media
  - I numeri vengono letti uno alla volta fino a che non si incontra un x=0, che segnala la fine dell'insieme