Laboratorio di Programmazione

Laurea in Bioinformatica

Docente: *Carlo Drioli*

Web: `www.scienze.univr.it/fol/main?ent=oi&id=39990`

Programmazione:

Dati
Strutture di controllo
Algoritmi

Materiale tratto dai lucidi ufficiali a corredo del testo:

D. Sciuto, G. Buonanno e L. Mari

“Introduzione ai sistemi informatici ”

2005 - McGrawHill

Problemi e algoritmi

Definire il problema

- **Eliminare le ambiguità nella formulazione del problema**
- **Individuare il risultato che si vuole ottenere, gli obiettivi da raggiungere**
- **Evidenziare**
  - le regole da rispettare
  - i vincoli interni ed esterni
  - i dati espliciti ed impliciti
- **Eliminare i dettagli inutili ed ambigui**

Procedura effettiva

- **Si dice *procedura effettiva per un esecutore* una successione di azioni tale che:**
  - tutte le azioni della successione sono *elementari* per l'esecutore, che è in grado di eseguire ciascuna di esse in un tempo finito e in modo deterministico, cioè ottenendo sempre gli stessi risultati a parità di input;
  - è fissato l'ordine di esecuzione delle azioni;
  - è esplicitamente specificato il modo in cui un'azione utilizza i risultati delle azioni che la precedono.

Algoritmo (definizione informale)

- Sequenza **finita** di istruzioni,
- **comprensibili** da un esecutore (si può trattare di uno strumento automatico),
- che descrive come **realizzare un compito** (come risolvere un "problema").

- Alcuni esempi
  - Istruzioni di montaggio di un elettrodomestico
  - Uso di un terminale Bancomat
  - Calcolo del massimo comune divisore di numeri naturali

Esecutori e linguaggi

- **Un esecutore è definito in base a tre elementi:**
  - l'insieme delle **operazioni** che è capace di compiere;
  - l'insieme delle **istruzioni** che capisce (**sintassi**);
  - quali **operazioni** associa ad ogni **istruzione** che riconosce (**semantica**).
- **Il calcolatore "capisce" le istruzioni che fanno parte del linguaggio macchina**
  - istruzioni primitive semplici (e.g. max 2 operandi)
  - attenzione all'efficienza (costi, complessità, velocità)
  - difficile e noioso da utilizzare per un programmatore
- **La soluzione si dice *effettiva* se l'esecutore è in grado di:**
  - interpretarla
  - compiere le azioni (in un tempo finito!)

Dal problema alla soluzione automatica

- Specifiche dei requisiti:
  descrizione precisa e corretta dei requisiti (verificabilità) ---> cosa?

- Progetto:
  procedimento con cui si individua la soluzione ---> come?

- Soluzione: algoritmo

Proprietà degli algoritmi

- **Correttezza**
  - L'algoritmo perviene alla soluzione del compito cui è preposto, senza difettare di alcun passo fondamentale
- **Efficienza**
  - L'algoritmo perviene alla soluzione del problema usando la minima quantità di risorse fisiche
    - tempo di esecuzione, memoria, …

Alcuni concetti

- ***Algoritmo*** = descrizione di come si risolve un problema
- ***Programma*** = algoritmo scritto in modo che possa essere eseguito da un calcolatore (linguaggio di programmazione)
- ***Linguaggio macchina*** = linguaggio effettivamente "compreso" da un calcolatore, caratterizzato da
  - istruzioni primitive semplici (e.g. max 2 operandi)
  - attenzione all'efficienza (costi, complessità, velocità)
  - difficile e noioso da utilizzare per un programmatore

- **Due aspetti rilevanti:**
  - **produrre algoritmi** (cioè capire la sequenza di passi che portano alla soluzione di un problema)
  - **codificarli in programmi** (cioè renderli comprensibili al calcolatore)

Algoritmi

Formalizzazione

Codifica degli algoritmi

- **Algoritmo formulato per essere comunicato tra esseri umani**
  - sintetico e intuitivo
  - codificato in linguaggi informali o semi-formali (linguaggio naturale, diagrammi di flusso, …)
- **Algoritmo formulato per essere eseguito automaticamente**
  - preciso ed eseguibile
  - codificato in linguaggi comprensibili dagli esecutori automatici (linguaggio macchina o linguaggio di programmazione di alto livello)

Algoritmi e variabili

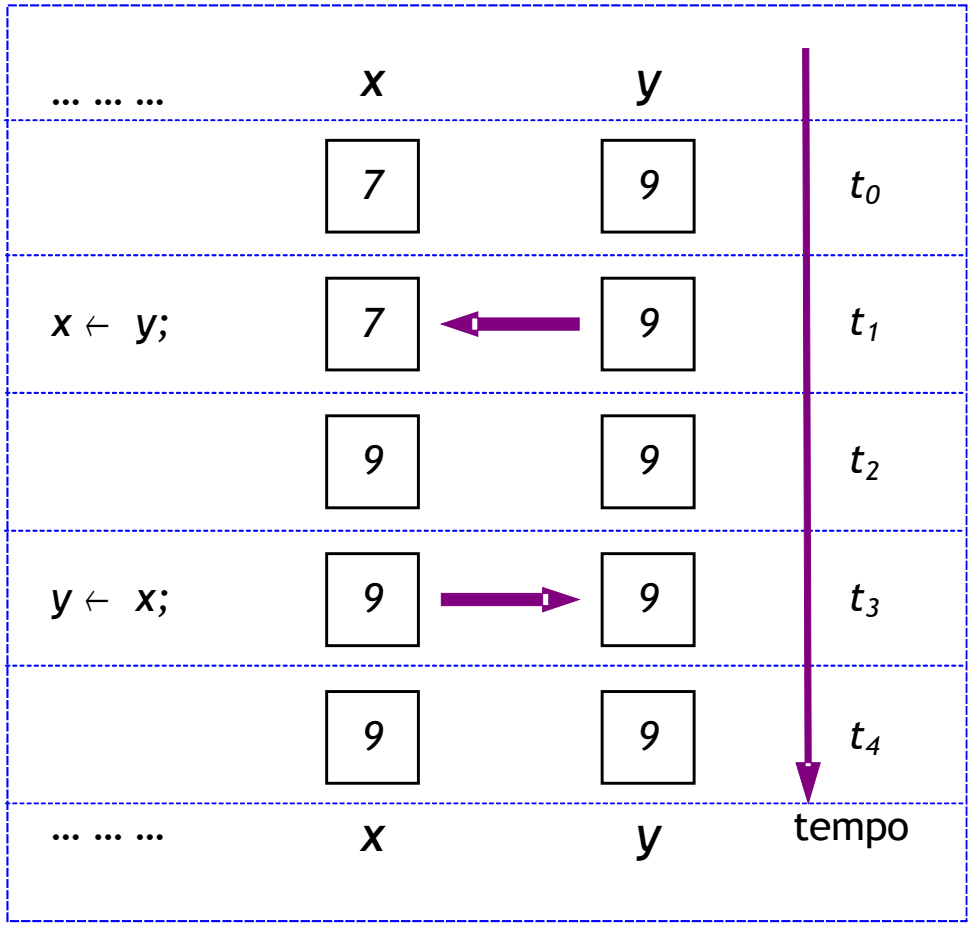
- **Gli algoritmi sono parametrici:**
  - producono un risultato che dipende da un insieme di dati di partenza;
  - descrivono la soluzione non di un singolo problema, ma di una intera **classe di problemi** strutturalmente equivalenti.
  - Esempi:
    - l'algoritmo per la moltiplicazione di due numeri specifica come effettuare il prodotto di *tutte* le possibili coppie di numeri;
    - l'algoritmo per la ricerca di un libro nello schedario della biblioteca vale per tutti i possibili libri;
    - …
- **Le istruzioni dell'algoritmo fanno riferimento a *variabili*, il cui valore non è fissato a priori ma cambia a seconda della situazione elaborativa in cui l'esecutore si trova.**

Uso delle variabili

- **All'interno di espressioni,**
  - l'esecutore usa il valore contenuto nelle variabili per calcolare il risultato dell'espressione,
  - per esempio **op1 + op2 × op3** oppure **op1 / op2 - op3**, …
- **in istruzioni di assegnamento**
  - introdurre nel contenitore identificato dal nome della variabile il valore specificato a destra dell'assegnamento;
  - per esempio **r ← 35** (assegna 35 alla variabile il cui nome è **r**), **pi ← 3,14**, …
- **in istruzioni di assegnamento combinate con espressioni**
  - assegna a una variabile il risultato ottenuto dalla valutazione di un'espressione, per esempio in "**circ ← 2 × r × pi**" il risultato dell'espressione **2 × r × pi** viene calcolato utilizzando i valori contenuti nelle variabili **r** e **pi** e il risultato viene poi assegnato alla variabile **circ**;
  - la stessa variabile può comparire in entrambi i lati dell'istruzione di assegnamento, per esempio in "**k ← k + 1**" il valore contenuto in **k** viene utilizzato per trovare il valore dell'espressione **k + 1** che viene memorizzato come nuovo valore di **k**.

Assegnamento di valori a variabili

- Il valore assegnato a una variabile si **sostituisce** a quello che era presente in precedenza: il vecchio valore non potrà più essere recuperato.
- Esempio: si ipotizzi di voler **scambiare** i valori contenuti in due variabili ***x*** e ***y***.
- Soluzione proposta:
  doppio assegnamento del tipo
  x ← ***y***
  y ← ***x***
  per indicare che il valore di ***y*** deve essere copiato in ***x*** e che, nello stesso tempo, il valore di ***x*** sia trasferito in ***y***.
- Le istruzioni però vengono eseguite in **sequenza**! Quindi l'assegnamento ***x*** ← ***y*** viene completato prima di iniziare ***y*** ← ***x***.

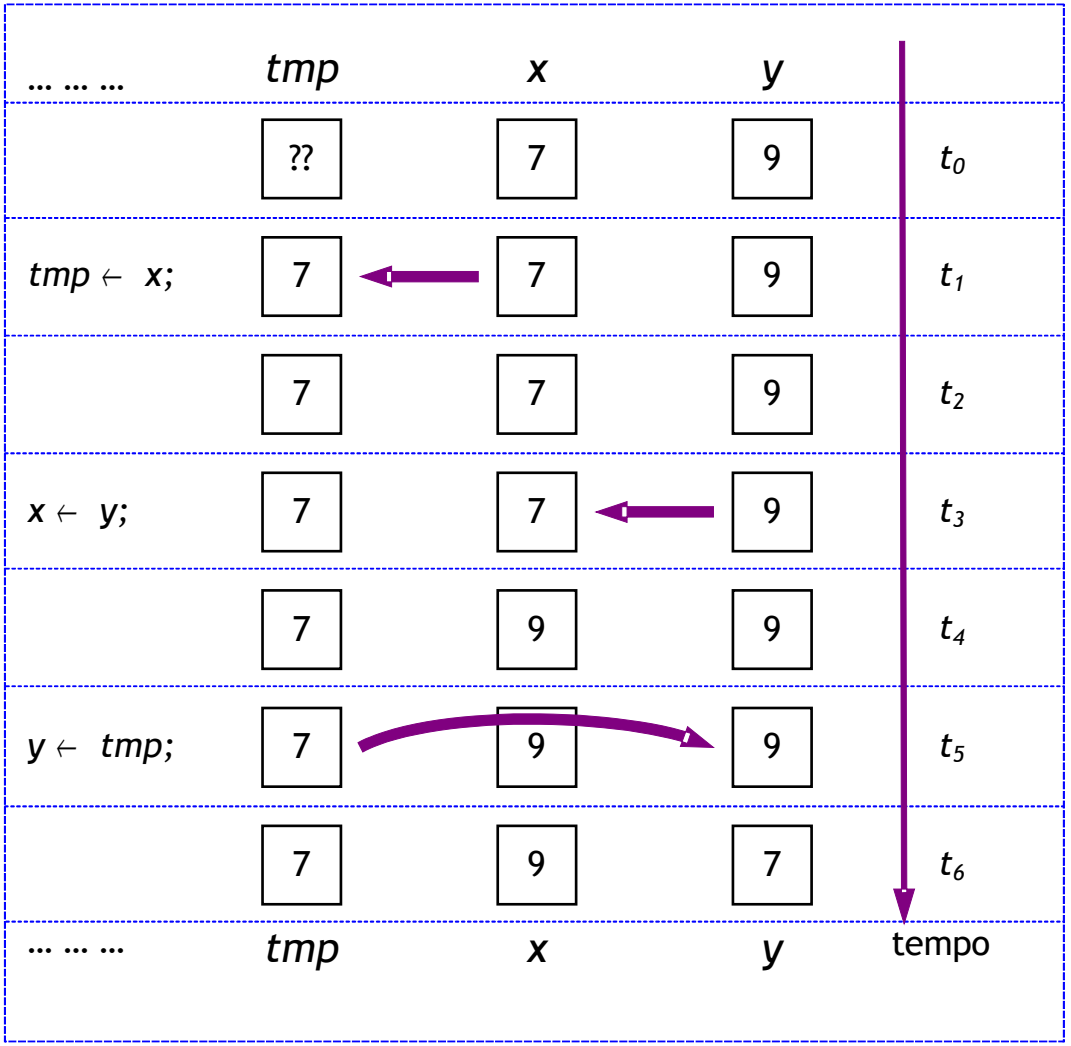


Assegnamento di valori a variabili

Soluzione corretta:
uso di una variabile aggiuntiva (*tmp*), come strumento di memorizzazione temporanea ("buffer") del valore originariamente contenuto in *x*

tmp ← *x*
x ← *y*
y ← *tmp*

In questo modo lo scambio avviene senza perdere i valori originari



Dati e istruzioni

- Tipi di dati
  - Numeri naturali o interi o reali (1, -2, 0.34)
  - Caratteri alfanumerici (A, B, ..)
  - Dati logici o booleani (Vero, Falso)
  - Array o vettore di n elementi ({1,2,3})
- Istruzioni
  - Operazioni di Input/Output (es. *leggi*, *scrivi*)
  - Operazioni Aritmetico-logiche (es. *max* = *A* + *B*)
  - Strutture di controllo (es. *SE*, *RIPETI*)

Operazioni elementari

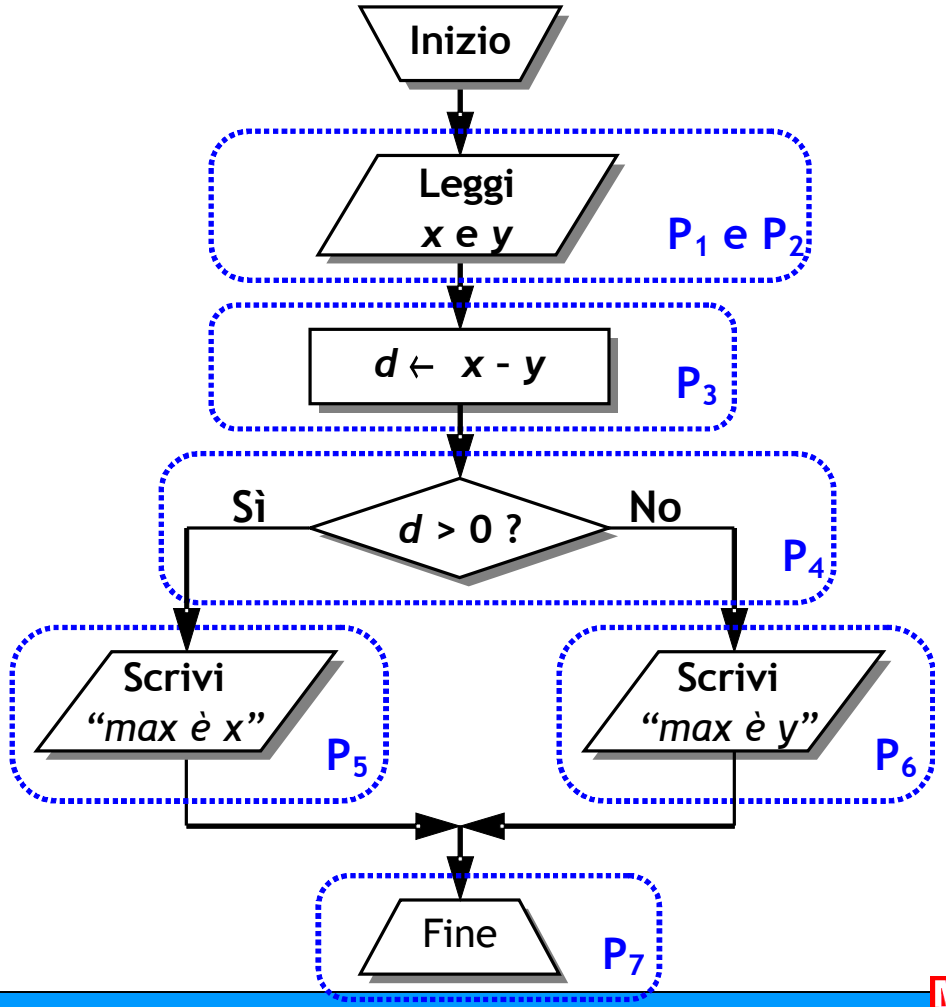
- **Operazioni aritmetiche e assegnamenti di valori a singole variabili**
  - Es. C ← (A + B)
- **Condizioni sul valore di singole variabili**
  - se (A > B) allora … altrimenti …
- **Lettura e scrittura di variabili**
  - “Leggi A” oppure “Stampa B”

Rappresentazione degli algoritmi

- Linguaggio naturale
- Diagramma a blocchi
- Pseudo codice
- Linguaggio di programmazione

Rappresentazione degli algoritmi

- **Linguaggio Naturale**
  - Leggi due numeri
  - Confrontali
  - Stampa un messaggio che dica quale dei due numeri e' il massimo
  - …
- **Diagramma di flusso**



Rappresentazione degli algoritmi

Pseudo Codice

```
leggi x, y
d ← x - y
SE d > 0
    ALLORA
        stampa "il max e' x"
    ALTRIMENTI
        stampa "il max e' y"
FINESE
```

Operazioni Logiche (Algebra Booleana)

Algebra di Boole

- L'algebra di Boole (dal suo inventore G. Boole) serve a descrivere le operazioni logiche.
- Componenti dell'algebra di Boole:
  - Operatori booleani
  - Regole di trasformazione ed equivalenza tra operatori booleani
- Gli operandi booleani assumono solo due valori:

  Vero/Falso True/False 1/0 Sì/No …

Operatori e tavole di verità

A	not A
0	1
1	0

A	B	A and B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A or B
0	0	0
0	1	1
1	0	1
1	1	1

A	B	A xor B
0	0	0
0	1	1
1	0	1
1	1	0

A	B	A ≡ B
0	0	1
0	1	0
1	0	0
1	1	1

A	B	A nand B
0	0	1
0	1	1
1	0	1
1	1	0

A	B	A nor B
0	0	1
0	1	0
1	0	0
1	1	0

Notazione

- Esistono convenzioni diverse:
  - **Negazione** not A ¬ A A ! - A
  - **Congiunzione** A and B A ∧ B A & B A × B
  - **Disgiunzione** A or B A ∨ B A | B A + B
  - **Disgiunzione esclusiva** A xor B A ^ B A ⊕ B
    [equivale a (A and (not B)) or ((not A) and B)]

  - **Implicazione** A → B A ⊃ B A ⇒ B
    (se ... allora)
  - **Doppia implicazione** A ↔ B A ≡ B A ⇔ B
    (se e solo se)

I programmi

Alcuni concetti

- ***Algoritmo*** = descrizione di come si risolve un problema
- ***Programma*** = algoritmo scritto in modo che possa essere eseguito da un calcolatore (linguaggio di programmazione)
- ***Linguaggio macchina*** = linguaggio effettivamente "compreso" da un calcolatore, caratterizzato da
  - istruzioni primitive semplici (e.g. max 2 operandi)
  - attenzione all'efficienza (costi, complessità, velocità)
  - difficile e noioso da utilizzare per un programmatore

- **Due aspetti rilevanti:**
  - **produrre algoritmi** (cioè capire la sequenza di passi che portano alla soluzione di un problema)
  - **codificarli in programmi** (cioè renderli comprensibili al calcolatore)

Parti fondamentali di un programma

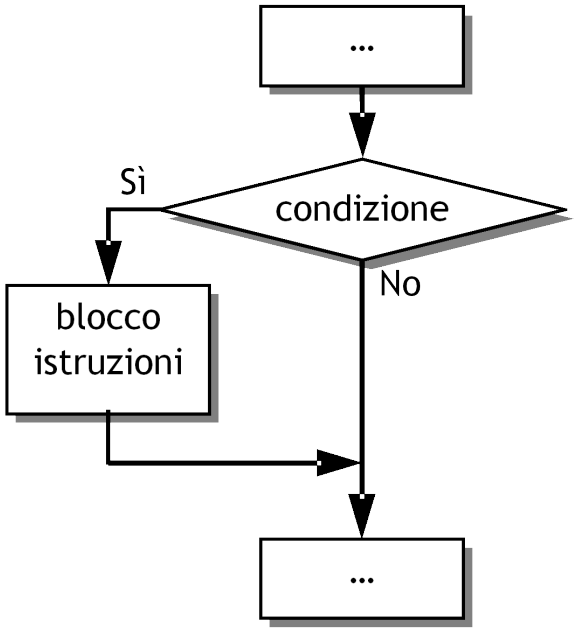
- *Identificazione* del programma
- *Dichiarazione* delle variabili utilizzate, di cui sono indicati tipo e nome
- Specificazione della parte *esecutiva* del programma, detta anche *corpo del programma*

Le istruzioni

- Istruzioni di ingresso/uscita
- Istruzioni aritmetico-logiche
- Istruzioni di controllo

Le strutture di controllo

Selezione semplice



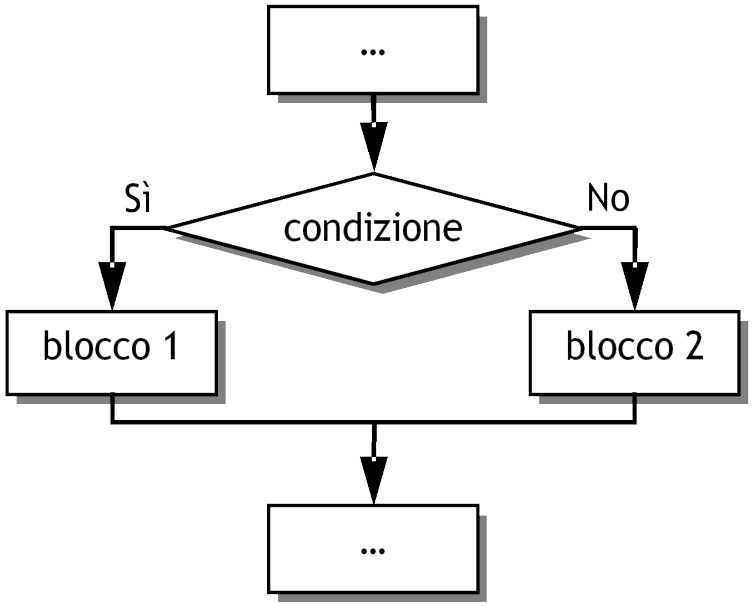
Pseudocodice

```
....
SE condizione
    ALLORA
        blocco istruzioni
FINESE
...
```

Java

```
....
if(condizione)
{
    blocco istruzioni
}
...
```

Selezione a due vie



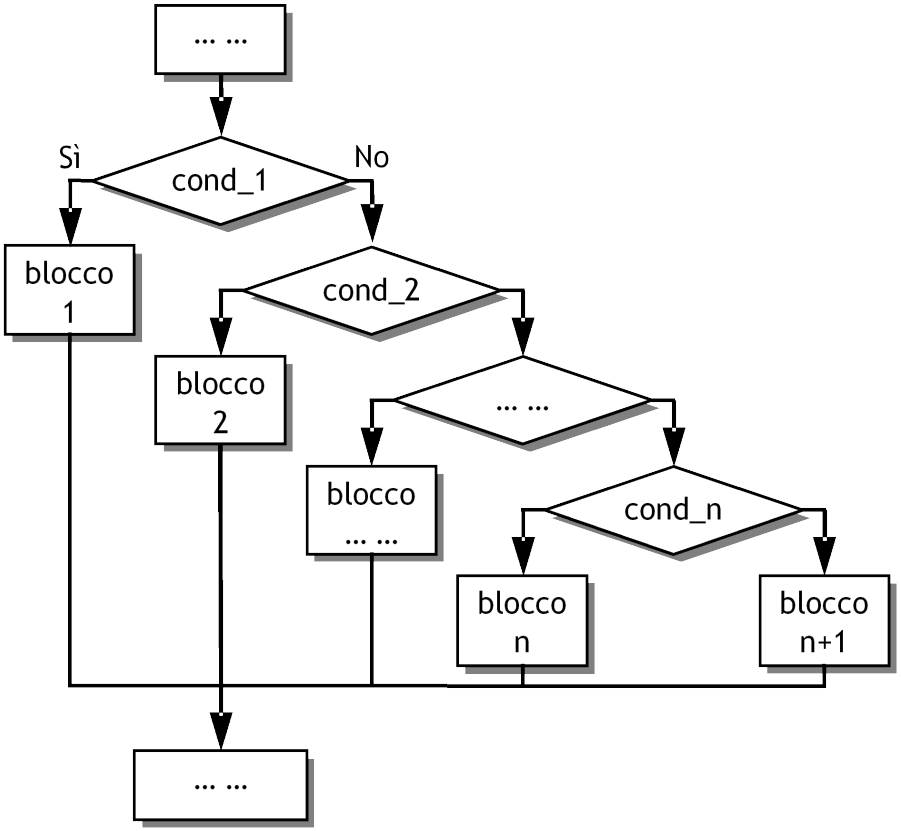
Pseudocodice

```
....
SE condizione
    ALLORA
        bbcco 1
    ALTRIMENTI
        bbcco 2
FINESE
...
```

Java

```
....
if(condizione){
    bbcco 1
}
else{
    bbcco 2
}
...
```

Selezione a più vie



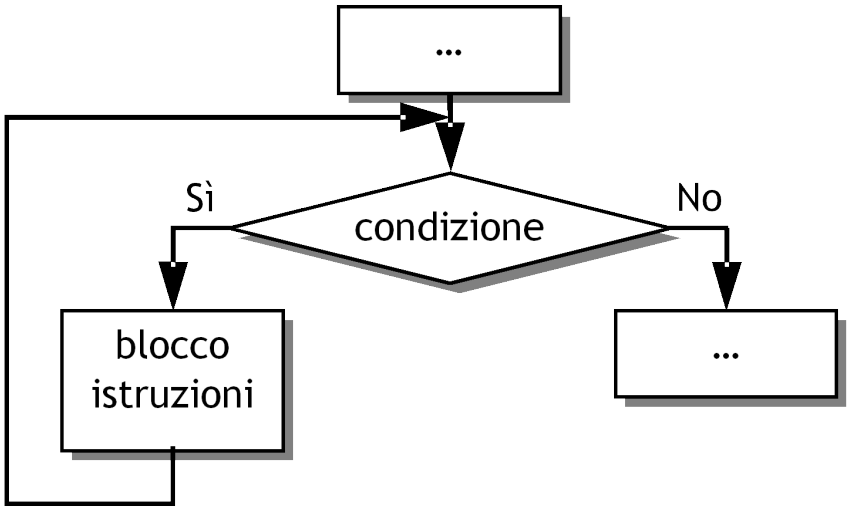
Pseudocodice

```
....
SE cond 1
    ALLORA
        bbcco 1
    ALTRIMENTI
    SE cond 2
        ALLORA
            bbcco 2
        ALTRIMENTI
        ...
    FINESE
FINESE
...
```

Java

```
....
if(cond1 ){
    bbcco 1
}
else {
    if(cond2){
        bbcco 2
    }
    else
    ...
}
...
```

Ciclo a condizione iniziale



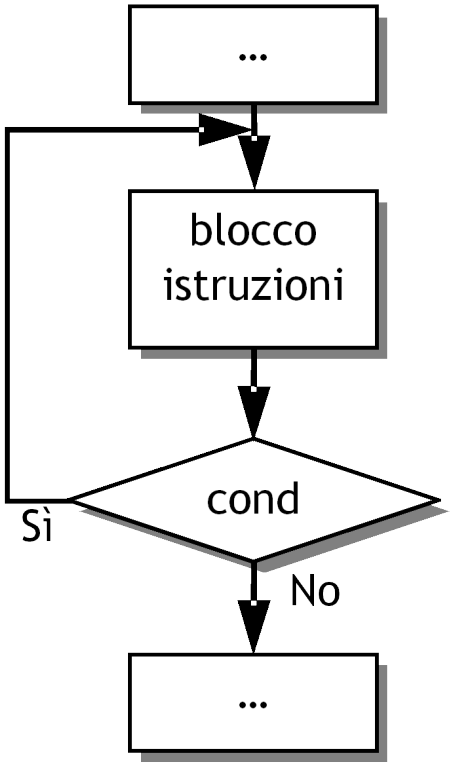
Pseudocodice

```
....
QUANDO condizione ESEGUI
    blocco istruzioni
RIPETI
....
```

Java

```
....
while (condizione ){
    blocco istruzioni
}
....
```

Ciclo a condizione finale



Pseudocodice

```
....
ESEGUI
    blocco istruzioni
QUANDO condizione
....
```

Java

```
....
do {
    blocco istruzioni
}while (condizione)
....
```

Esempi di algoritmo (in pseudocodice)

Numeri pari e dispari

Pseudocodice

```
variabili n, resto: interi

leggi n
resto = n % 2
SE resto == 0
    ALLORA
        scrivi "pari"
    ALTRIMENTI
        scrivi "dispari"
FINESE
```

% : operatore "modulo"

Somma di una sequenza di numeri

Pseudocodice

```
variabili somma, numero, quanti, cont: interi

leggi quanti
somma = 0
cont = 0
QUANDO cont < quanti ESEGUI
        leggi numero
        somma = somma + numero
        cont = cont + 1
RIPETI
scrivi somma
```