Dall'Algoritmo al programma

Lezione n. 3bis

Algoritmo - definizione intuitiva

Elenco preciso di operazioni, comprensibile da un esecutore, che definisce una sequenza finita di passi i quali risolvono ogni problema di un dato tipo (classe di problemi).

Esempio: operazioni necessarie per compiere una telefonata, per prelevare denaro dal bancomat, per iscriversi ad un esame, etc.

Origini dell'algoritmo

- Il concetto di algoritmo è antico e non è strettamente legato al calcolatore: l'esecutore può essere diverso
- Sono stati ritrovati algoritmi in tavolette antiche in Mesopotamia risalenti al 1800-1600 a.c.
- Il termine algoritmo deriva dal nome di un matematico arabo - alKhuwarizmi vissuto nel nono secolo d.c.

Esempio

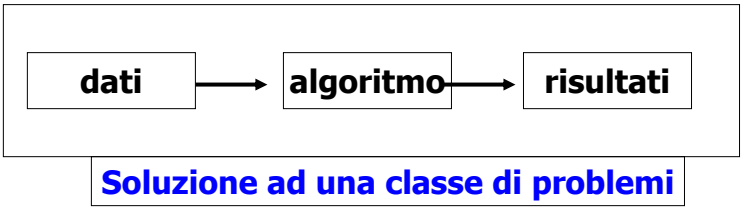
- Problema - Prelevare contanti in banca
- Analisi - si possono prelevare contanti in diversi modi:bancomat, mediante assegno, chiedendo un prestito, fare una rapina, etc.
- Algoritmo - si decide per il bancomat, si descrivono i passi operativi e le istruzioni che attivano il bancomat
- Elaborazione - esecuzione delle operazioni
- Risultati - i contanti

Esempio - bancomat macro operazioni

1. Estrazione della tessera bancomat
2. Inserimento della tessera nell'apposito macchinario
3. Scelta dell'operazione da eseguire
4. Inserimento del codice segreto
5. Conclusione operazione
6. Prelievo tessera
7. Prelievo contanti

Algoritmo - definizione rigorosa

Sequenza ordinata finita di passi, ripetibili e non ambigui, che se eseguita con determinati dati in ingresso (input) produce in uscita(output) dei risultati ovvero la soluzione di una classe di problemi



Caratteristiche dell'algoritmo

- Un algoritmo si suppone sempre che comunichi con l'ambiente acquisendo dati e restituendo risultati
- Le operazioni di cui è composto l'algoritmo si dicono istruzioni
- Ogni istruzione opera su dei dati o prodotti dall'algoritmo stesso o acquisiti dall'esterno

Proprietà di un algoritmo

- Finitezza - deve portare alla soluzione in un numero finito di passi
- Generalità - per classi di problemi
- Ripetitività - con gli stessi dati deve fornire gli stessi risultati
- Determinismo o Non ambiguità - dando gli stessi dati si devono ottenere gli stessi risultati anche se è eseguito da persone diverse

Una ricetta non è un algoritmo

- di solito fra gli ingredienti vi sono espressioni ambigue come “un pizzico di sale” o “quanto basta” quindi lasciati alla soggettività
- non è vero che ripetendo gli stessi passi si ottengono gli stessi risultati

Descrizione dell’algoritmo - I

- L’algoritmo deve essere eseguito da un esecutore (calcolatrice, uomo, meccanismo, ingranaggio meccanico, etc. non necessariamente dal computer)
- Occorre quindi fornire all’esecutore una serie di istruzioni comprensibili
- Occorre quindi descrivere le operazioni utilizzando un linguaggio preciso e generale

Descrizione dell'algoritmo - II

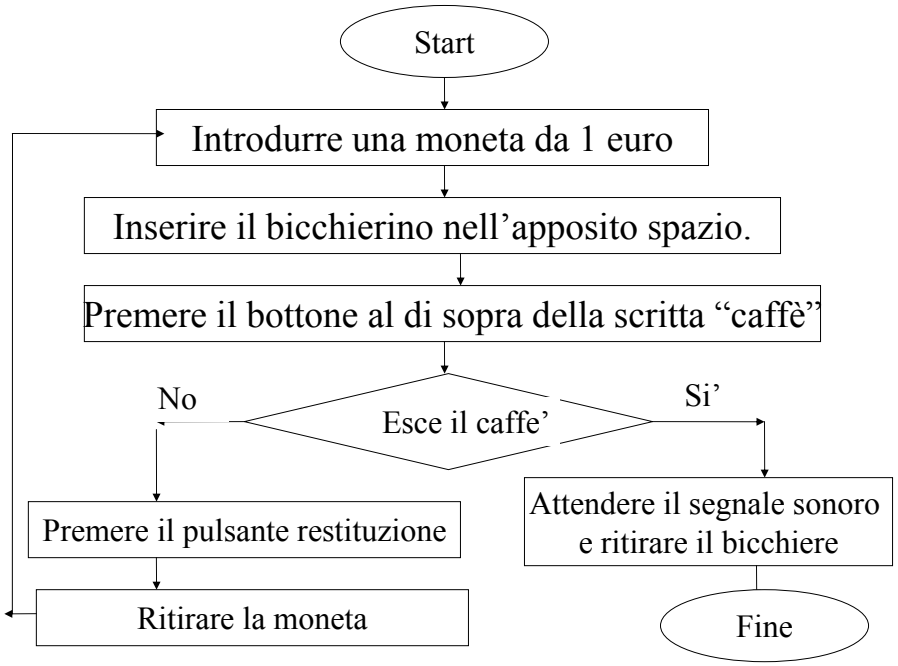
- Le asserzioni sono composte da due parti:
  - istruzioni - descrizione delle operazioni
  - dati - descrizione degli oggetti manipolati dalle operazioni
- esempio
  - inserisci la tessera - istruzione
  - digitare - istruzione
  - codice segreto - dato
  - importo - dato

Rappresentazione logica degli algoritmi

- pseudocodifica (esempio precedente)
  - professionale
    - verbi di "esecuzione" (effetto osservabile)
    - condizioni
    - iterazioni
- diagramma a blocchi
  - utile a scopi dimostrativi
    - indica un flusso di istruzioni ovvero la sequenza dei passi da eseguire
    - basato su simboli grafici
    - ogni simbolo corrisponde a un costrutto
- le due modalità sono semanticamente equivalenti

Esempio in pseudocodifica

1. Introdurre una moneta da un euro
2. Inserire il bicchierino nell'apposito spazio
3. Premere il bottone al di sopra della scritta "caffè"
4. Se esce il caffè, attendere il segnale sonoro e quindi ritirare il bicchiere
5. Altrimenti premere il bottone al di sopra della scritta "restituzione" e ritirare la moneta.
6. Ricominciare da 1



Pseudocodice - esempio

```
Inizio
Inizializza MAX a 0
Inizializza N1 a 0
Leggi N1
Finché N1 != 999
    Se N1 > MAX allora
        Assegna N1 a MAX
    Leggi N1
Stampa MAX
Fine
```


I diagrammi a blocchi - I

- la diagrammazione a blocchi o flow chart è un metodo per rappresentare l'algoritmo in modo sintetico e preciso
- un diagramma a blocchi indica un flusso di istruzioni ovvero la sequenza dei passi da eseguire
- è basato su simboli grafici
- ogni simbolo corrisponde ad un preciso costrutto o insieme di istruzioni

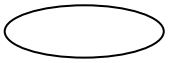
I diagrammi a blocchi - II

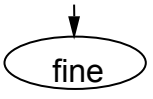
- Un diagramma a blocchi è un insieme di blocchi elementari costituito sempre dalle seguenti parti:
  - blocco di inizio
  - blocco di fine
  - numero finito di blocchi di lettura/scrittura o di azione
  - numero finito di blocchi di controllo (opzionale)

I diagrammi a blocchi - III

Rappresenta il flusso (l'ordine) del diagramma

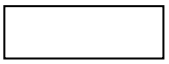
Istruzioni di inizio e di fine





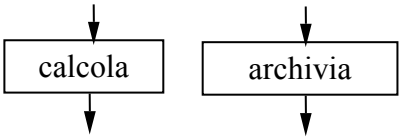
I diagrammi a blocchi - IV

istruzione operativa (*effettiva*)



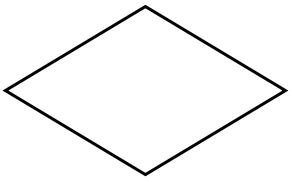
Rappresenta una elaborazione

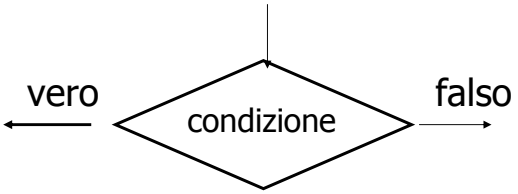
Esempi:



I diagrammi a blocchi - V

simbolo di decisione

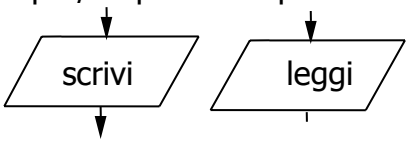




I diagrammi a blocchi - VI

Simbolo di input/output

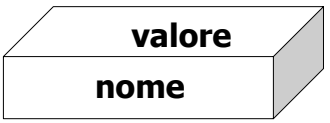
Rappresenta un'operazione di input/output. Esempi:



Dati: costanti e variabili

- Un dato può essere:
- costante - un dato che assume un valore all'inizio dell'algoritmo e lo mantiene per tutta l'esecuzione di questo

  importo x 1936,27 rate di conversione
- variabile - è una coppia <nome, valore>



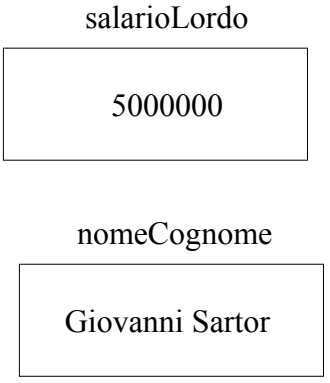
Variabili

- Il valore delle variabili deve appartenere a una categoria che solitamente sono:
  - numeri interi - 60
  - numeri reali - 220,56
  - sequenze di caratteri - Mario Rossi
  - valori booleani - vero, falso
- I valori una volta che appartengono ad una categoria di dati devono rispettare le regole della categoria
  - Esempio i numeri si possono moltiplicare fra loro i caratteri no

I dati in dettaglio

- Variabile:
  - locazione di memoria avente un nome, in cui possono essere caricati (copiati) contenuti o valori si un certo tipo
- Ogni variabile ha:
  - Un nome (salarioLordo)
  - Un tipo (integer)
  - Un valore (“5000000”)

Variabili



Istruzioni - I

- Istruzioni operative -
  - istruzioni che eseguono azioni che producono risultati anche parziali
  - moltiplica, conta, trasforma, etc.
- Istruzioni di controllo
  - istruzioni che guidano l'andamento dell'esecuzione
  - se piove allora resto a casa
- Istruzioni di salto
  - istruzioni che spostano il punto di esecuzione da una parte all'altra dell'algoritmo
  - salto condizionato, salto incondizionato

Istruzioni - II

- Istruzioni di inizio e di fine
  - sono le istruzione di apertura e di chiusura dell'algoritmo
- Istruzioni di ingresso e di uscita
  - sono le istruzioni che consentono l'acquisizione dei dati (ingresso) e l'emissione dei risultati (uscita)
  - lettura del codice, inserimento della carta, stampa del fogliettino, emissione dei soldi

Le istruzioni operative

- Dichiarazione di variabile
  - Crea le variabili (predispone le zone di memoria e attribuisce loro dei nomi)
    - var salarioLordo, salarioNetto;
- Assegnamento
  - Memorizza un contenuto (a destra) nella variabile (a sinistra)
    - Un valore particolare
      - salarioLordo = 1000000;
      - imposta = 350000;
    - Il risultato di un'operazione
      - salarioNetto = salarioLordo – imposta;
  - NB. In molti linguaggi "=" non indica l'eguaglianza ma l'assegnamento

Gli operatori

- Nel lato destro delle istruzioni di assegnamento vi possono essere operazioni.
- La variabile memorizza il risultato dell'operazione. Esempi:
  - Concatenamento

    nomeCognome = "Giovanni " + " Sartor";

    - Somma

    somma = 5 + 8;
- + Addizione 5 + 7
- -, / , *, %,

Istruzioni di input e output

- Istruzioni di input: acquisiscono informazioni dai disposiviti di input.
  - primoNumeroStringa = window.prompt("inserisci il tuo stipendio");
- Istruzioni di output
  - alert("stipendio = " + stipendio);
  - document.write ("stipendio = " + stipendio)

Istruzioni di controllo: i salti

- Istruzione di salto (*jump, goto*):
  - prescrive che l'ulteriore elaborazione continui a partire da un certo punto
  - trasferisce il controllo al punto indicato

Salti incondizionati e condizionati

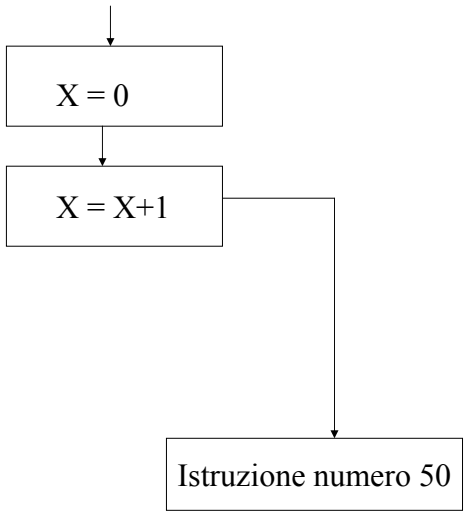
Salto incondizionato

1. X = 0;
2. X = X+1;
3. Goto 50

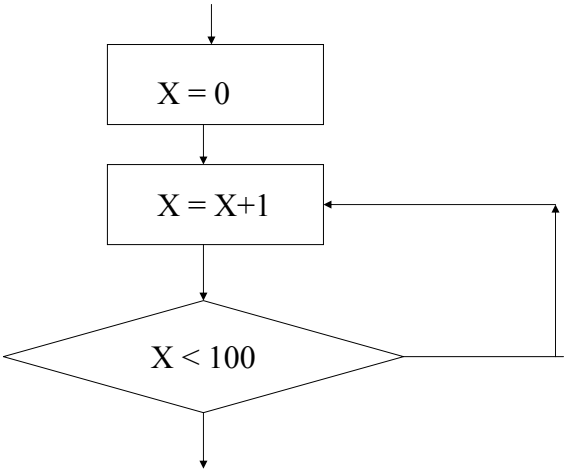
Soluzione: Salto condizionato

1. X = 0;
2. X = X+1;
3. If X < 10 go to 2

Salto incondizionato



Salto condizionato



Istruzioni di controllo

- Condizione
  - è una espressione che si verifica o vera o falsa
  - ogni tipo di dati ha i suoi operatori di confronto

Le condizioni

Operatori di confronto dei numeri:

- uguale = X==3
- maggiore > X>3
- maggiore uguale => X>=3
- minore < X<3
- minore uguale <= X<=3
- diverso <> X<>3

Operatori di confronto dei caratteri:

- uguale = = “monica”==Y
- diverso <> “monica” <>Y

Gli operatori booleani - I

Operatori booleani:

- **and** "canta" e "danza" ossia la condizione è vera quando il soggetto canta *e* danza contemporaneamente
- **or** "canta" o "danza" - in modo alternativo ossia sono valide le seguenti situazioni: "canta e non danza", oppure "danza e non canta", oppure "canta e danza"
- **not** non "canta" – negazione
- **xor** "canta" o "danza" - in modo esclusivo ovvero sono valide le seguenti situazioni "canta e non danza" oppure "danza e non canta"

Gli operatori booleani - II

A	B	A AND B	A OR B	NOT A	A XOR B
VERO	VERO	VERO	VERO	FALSO	FALSO
FALSO	VERO	FALSO	VERO	VERO	VERO
VERO	FALSO	FALSO	VERO	FALSO	VERO
FALSO	FALSO	FALSO	FALSO	VERO	FALSO

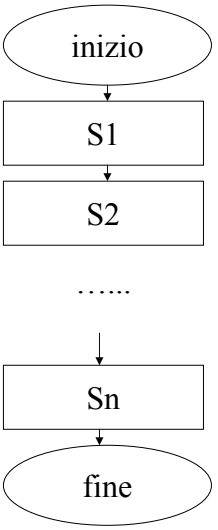
Programmazione strutturata

- la programmazione strutturata è quel procedimento che permette di ottenere algoritmi facilmente documentabili e comprensibili
- si basa su tre costrutti:
  - Sequenza
  - Selezione
  - Ripetizione
- regole base:
  - i salti sono rigorosamente proibiti
  - esiste una sola fine di tutto il programma
  - le selezioni si chiudono sempre

La sequenza

- Le istruzioni si susseguono rispetto ad un ordine e vengono eseguite nella sequenza indicata
- {<S1>; <S2>; … <Sn>}

La sequenza



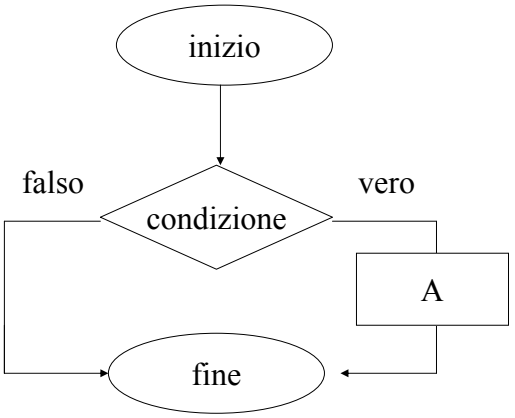
La selezione

- If (<condizione>)
  {
  <S1>…..
  <Sn>
  }

  BLOCCO

  - Se la “condizione” è vera allora vengono eseguite l’istruzioni del blocco
  - Se la “condizione” e’ falsa, allora il blocco non viene eseguito

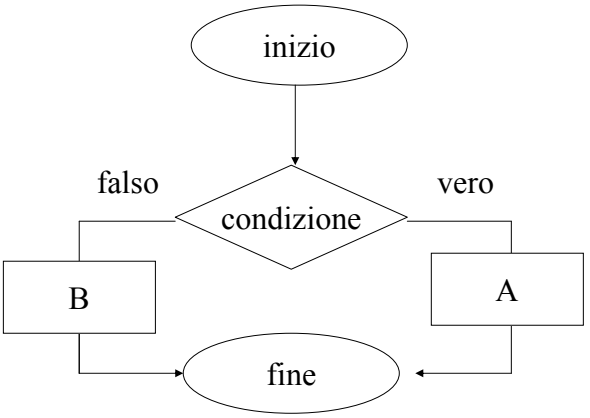
La selezione



La selezione con alternativa

- If (<condizione>)
  then
  {
  <blocco1>
  }
  else
  {
  <blocco2>
  }
  - Se la condizione è vera allora viene eseguito il blocco1
  - altrimenti il blocco2

La selezione con alternativa

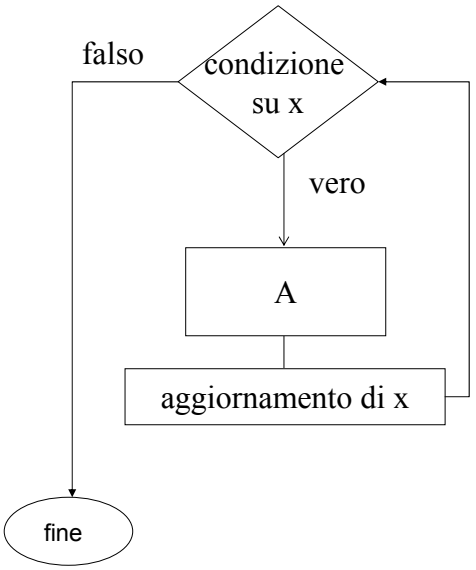


La ripetizione (while)

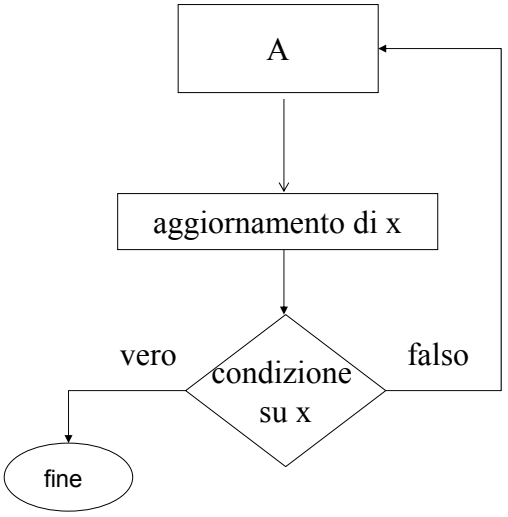
```
While (<condizione>)
  do
  {
  <blocco>
  }
```

- il blocco viene ripetuto finché la condizione risulta vera
- quando l'espressione risulta falsa la ripetizione cessa e si passa all'istruzione successiva

La ripetizione con la condizione vera in testa



La ripetizione con la condizione falsa in coda

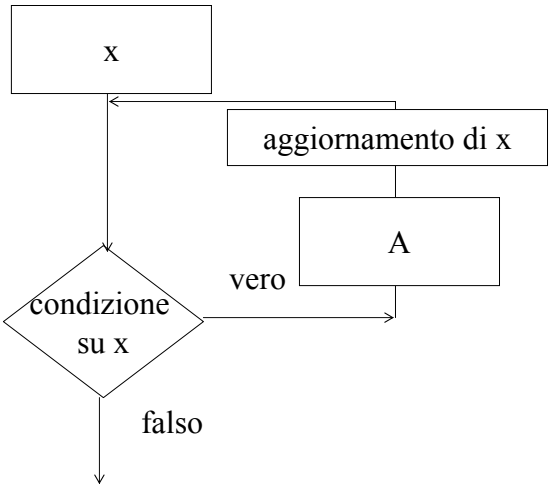


La ripetizione enumerativa (for)

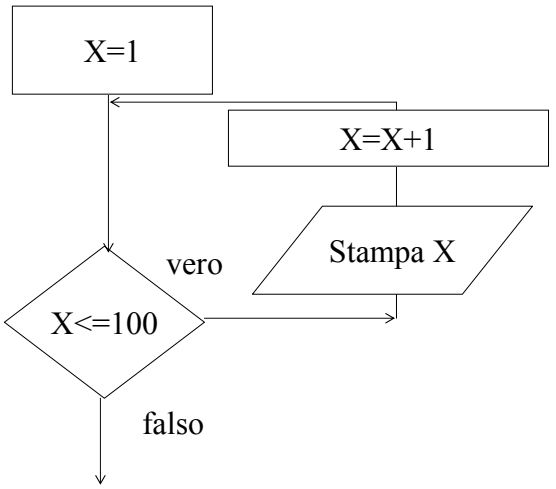
for (<espressione iniziale>, <condizione>, <aggiornameto espressione>)
{
<blocco>
}

- il blocco viene ripetuto finché la condizione risulta vera
- la formula di aggiornamento modifica i valori di confronto nell'espressione fino a quando la condizione non diviene vera

La ripetizione enumerativa (for)



Esempio di for



Cosa produce questo algoritmo ?

Potenza della programmazione strutturata

- Le tre strutture presentate consentono di esprimere qualsiasi algoritmo
- Teorema di Bohm-Jacopini:

"Ogni diagramma a blocchi non strutturato è sempre trasformabile in un diagramma a blocchi strutturato equivalente..."

(...con l'eventuale aggiunta di una variabile)

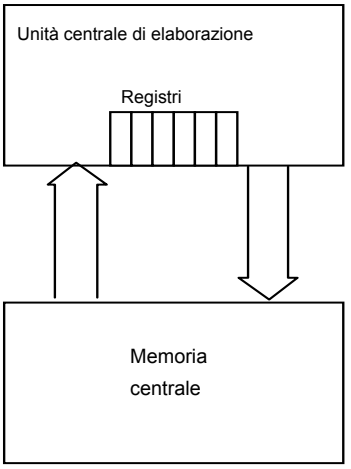
Il metodo top-down

- Metodo top-down: scomposizione progressiva del problema e delle azioni (istruzioni) che lo risolvono
- Favorisce la divisione del lavoro:
  - Analisi: definizione della struttura generale del programma, dei moduli principali che lo compongono, e delle operazioni astratte svolte da ciascuno di essi, le cosiddette "specifiche"
  - Programmazione: realizzazione dei programmi che svolgono le singole operazioni individuate nella fase di analisi

Scomposizione in sotto-algoritmi: sviluppo top-down

- Spesso per semplificare il flusso procedurale si rappresentano insieme di istruzioni in un solo blocco concettualmente omogeneo
- Successivamente si scompone il blocco in istruzioni sempre più "fini"
- Si procede dal generale al particolare
- Questo metodo di analizzare i problemi partendo da macro-blocchi per poi arrivare alle istruzioni più dettagliate è detto top-down

La macchina di von Neumann (1)



La macchina di von Neumann (2)

- L'elaborazione si svolge come segue:
  - un'istruzione e i dati che tale istruzione deve manipolare vengono trasferiti o "caricati" dalla memoria nei registri;
  - l'unità centrale esegue l'istruzione;
  - gli eventuali risultati vengono trasferiti dai registri alla memoria;
  - si passa all'istruzione successiva (o a quella specificata dall'istruzione di controllo, se l'istruzione eseguita era di questo tipo).
- NB: L'elaborazione è sequenziale: viene eseguita un'istruzione per volta

Macchina di von Neumann e programmazione

- Un programma per una macchina di Von Neumann consiste, pertanto:
  - nella descrizione/prescrizione di una sequenza di operazioni elementari sui dati,
  - combinate con istruzioni di controllo, che modificano l'ordine nel quale eseguire le operazioni sui dati

Analisi di algoritmi

- Ricerca di un elemento in un insieme ordinato di elementi (per esempio un numero telefonico in un elenco ordinato alfabeticamente o un numero intero in un insieme ordinato di numeri interi)
- Siano dati 100 numeri interi; la domanda è: il numero N (per es. 45) è contenuto nell'insieme?

Algoritmo:
Tre variabili: *N; Elemento dell'insieme, Risultato*
Acquisisci *N* e *Insieme di elementi*
Imposta *Risultato* a `Insuccesso`
Ripeti la scansione degli elementi dell'insieme finchè *Risultato* vale `Successo` OR fino alla fine dell'insieme
    Confronto l' elemento successivo dell'insieme con N
    Se sono uguali allora imposta *Risultato* a `Successo`

Comunica la mondo il valore di *Risultato*

Ricerca sequenziale

- Se gli elementi dell'insieme sono 100
- Caso migliore: 1
- Caso peggiore: 100
- Caso medio: (se N è nella prima posizione il numero dei tentativi è 1) + (se N è nella seconda posizione il numero dei tentativi è 2) + ecc..
  - Allora il caso medio è (1 + 2 + 3 + …+ 100) / 100
  - Più in generale (1 + 2 + 3 + …+ N) / N
  - Vale N * (N+1)/2
  - Diviso N dà (N+1)/2 che se N è 100 vale 50

Ricerca binaria

- Si riduce il problema della metà (circa) a ogni interazione limitando la ricerca alla parte precedente o alla parte successiva del valore di metà dell'insieme
- Ogni tentativo permette di ridurre l'ampiezza del problema del fattore 2
- Il primo tentativo riduce il numero delle possibilità a *N/2*; il secondo a *N/4*, ecc.
- Nel caso peggiore si dovrà proseguire finché rimane una sola possibilità
- Il numero di passi è allora *N/2/2/2/…/2 = 1* (con k divisioni; dove k è il numero dei tentativi necessari a trovare il valore cercato)
- Caso medio $N/2^k = 1$ cioè $K = log_2 N$

Misure nel caso peggiore

Valore di N	*Ricerca lineare*	*Ricerca binaria*
10	10	4
100	100	7
1 000	1 000	10
10 000	10 000	14
100 000	100 000	17
1 000 000	1 000 000	20

Classi di complessità

Quando *N* raddoppia il tempo di esecuzione …	**Si dice complessità …**
Non cambia	costante
Aumenta di una costante “piccola”	logaritmica
Raddoppia	lineare
Un pò più che raddoppia	N log N
Aumenta di un fattore 4	quadratica
Aumenta di un fattore esponente di 2	polinomiale
Aumenta di molto secondo la base di cui N è esponente	esponenziale