

Lezione 1:

Concetti introduttivi

Informatica
Algoritmo
Programma

Introduzione all'Informatica - corso G (L. Pontieri)

Cos'è l'informatica?

◆ Esistono varie definizioni:

- Scienza dell'informazione
- Informazione + automatica
- Scienza dei calcolatori ("Computer Science")
- *Scienza e tecnica dell'elaborazione dei dati e, genericamente, del trattamento automatico dell'informazione* [Zingarelli]
- *Scienza del trattamento razionale, specialmente per mezzo di macchine automatiche, dell'informazione, considerata come supporto alla conoscenza umana e alla comunicazione* [Académie Française]

Introduzione all'Informatica - corso G (L. Pontieri)

3

Informazione e comunicazione

◆ Informazione

- Notizia, dato o elemento che consente di avere conoscenza più o meno esatta di fatti, situazioni, modi di essere

◆ Comunicazione

- Scambio di informazione, mediante **messaggi**

Introduzione all'Informatica - corso G (L. Pontieri)

2

Cos'è l'informatica?

◆ Definizione "accettabile"

- **Scienza della rappresentazione e dell'elaborazione dell'informazione**
- Studia le caratteristiche dell'informazione e i modi di usarla, immagazzinarla, elaborarla e trasportarla in modo automatico

◆ L'informatica ha due anime:

- **metodologica**: studia i metodi per la soluzione di problemi e la gestione delle informazioni
- **tecnologica**: studia i calcolatori elettronici, le reti e i sistemi che li utilizzano

Introduzione all'Informatica - corso G (L. Pontieri)

4

Elaboratore elettronico

◆ Elaboratore elettronico

- è uno strumento **programmabile** per rappresentare, memorizzare e elaborare informazioni
- detto anche “computer” o “calcolatore”

◆ La prima decomposizione di un calcolatore è relativa alle seguenti macro-componenti

- **Hardware**: la struttura fisica del calcolatore (costituita da vari dispositivi)
- **Software**: l'insieme dei programmi che consentono all'hardware di svolgere dei compiti utili all'utente

5

Software e macchine virtuali

◆ Macchine virtuali

- Semplificano la comunicazione fra uomo e hardware
- Le diverse macchine e i relativi insiemi di operazioni sono via via più astratti: più vicini alla logica dell'utente e più lontani dalla logica del calcolatore come dispositivo elettronico
- Alla fine, comunque, l'unico responsabile dell'esecuzione del software è l'hardware disponibile

◆ Il software di base

- ➔ macchina virtuale più semplice da gestire e programmare

◆ Il software applicativo

- ➔ macchina virtuale utilizzabile per la risoluzione di problemi

7

Hardware e software

◆ Hardware: la macchina reale

- sistema di dispositivi di varia natura (elettronici, elettromagnetici, elettromeccanici, ottici ...)
- le operazioni (**istruzioni**) che l'hardware sa eseguire direttamente rappresentano le frasi del **linguaggio macchina** del calcolatore
- le istruzioni del linguaggio macchina sono molto semplici e il calcolatore può eseguirle in modo molto efficiente

◆ Software: insieme dei programmi eseguiti dal calcolatore

- mostra il calcolatore agli utenti come una **macchina virtuale** più semplice da usare rispetto all'hardware sottostante

6

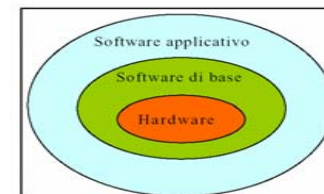
Classificazione del software

◆ Software di Base (es. il Sistema Operativo)

- permette una più semplice interazione con le componenti hardware (memorie, periferiche, ...)

◆ Software Applicativo

- consente agli utenti di utilizzare il calcolatore come una macchina virtuale utile per la **risoluzione di problemi**



8

La programmazione

Attività con cui si predispone l'elaboratore ad eseguire un **particolare insieme di azioni** su una **particolare tipologia di dati**, allo scopo di **risolvere un problema**.



9

I problemi

I problemi affrontati dalle applicazioni informatiche sono di natura e complessità molto varia, es.:

- Trovare il maggiore fra due numeri
- Dato un elenco di nomi e numeri di telefono, trovare il numero di una data persona
- Dati a e b , risolvere l'equazione $ax+b=0$
- Stabilire se una parola precede alfabeticamente un'altra
- Ordinare un elenco di nomi
- Creare, modificare e alterare suoni
- Analizzare, riconoscere e modificare immagini
- Gestione di un'organizzazione (private e pubbliche)
- Supportare operazioni di commercio elettronico

11

La programmazione

◆ Alcune domande fondamentali:

- Quali istruzioni esegue un elaboratore?
- Quali problemi può risolvere un elaboratore?
- Esistono problemi che un elaboratore non può risolvere?

◆ Il problema di fondo

- Come si costruisce la soluzione a un problema?
- Qual è il giusto "punto di partenza" per pensare la soluzione a un problema?
- Quali metodologie e tecniche usare?

10

I problemi

◆ Descrizione del problema

- La descrizione del problema non indica direttamente (in genere) un modo per risolverlo

specifica di un problema
≠
specifica del processo di risoluzione

◆ Risoluzione di un problema

- Comprensione
- Modellazione
- Individuazione di un opportuno metodo risolutivo (**algoritmo di risoluzione**)

12

Risoluzione di un problema

L'obiettivo fondamentale

Descrizione di un problema



Individuazione di un ALGORITMO

13

Algoritmo

- ◆ La parola **algoritmo** deriva dal nome di un autore scientifico persiano del IX secolo
 - **Abu Ja'far Mohammed ibn Musa al-Khowarizmi** scrisse, circa nell'825, il trattato "*Kitab al jabr w'al-muqabala*" (forse *regole di trasposto e semplificazione*) dove si descrivono regole per la semplificazione delle equazioni
 - **algebra** deriva da **al jabr** (parte del titolo del trattato)
 - **algoritmo** deriva da **al-Khowarizmi** (ultima parte del nome dell'autore, indicante la città di nascita)
- ◆ Il termine originario era *agorismo*, trasformato in *algoritmo* per analogia con *aritmetica*

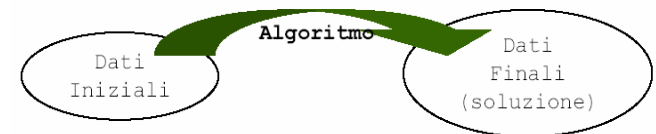
15

Algoritmo

- ◆ **Algoritmo** = sequenza finita di passi che risolve un problema in un tempo finito
- ◆ Esempi di "algoritmi":
 - Istruzioni di montaggio
 - Preparazione del caffè
 - Prelievo bancomat
 - Ricetta di cucina
 - Calcolo del *massimo comun divisore* tra due interi

14

Algoritmo



Si definisce **algoritmo** una **sequenza di azioni** che trasforma i dati iniziali in un numero finito di passi, elementari e non ambigui, per giungere al risultato finale.

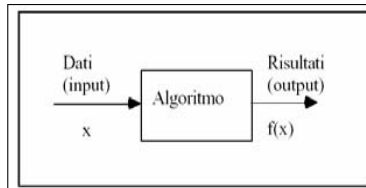
Questa sequenza di azioni è valida per un insieme di dati iniziali ben definito e può essere eseguita da un opportuno esecutore.

16

Algoritmo

In generale un algoritmo può essere visto come una **funzione**

- da un dominio di ingresso (**input**)
- ad un dominio di uscita (**output**)



17

Proprietà degli algoritmi

◆ Proprietà fondamentali

- **Generalità**: applicabile a ogni insieme di dati di ingresso appartenente al dominio di definizione del problema
- **Non-ambiguità**: ogni azione deve essere univocamente interpretabile dall'esecutore (persona o "macchina")
 - costituito da operazioni appartenenti ad un determinato insieme di operazioni fondamentali
- **Eseguibilità**: ogni azione deve essere eseguibile in un tempo finito da parte dell'esecutore dell'algoritmo
- **Finitezza**: per ogni insieme di dati di ingresso, il numero totale di azioni da eseguire deve essere finito

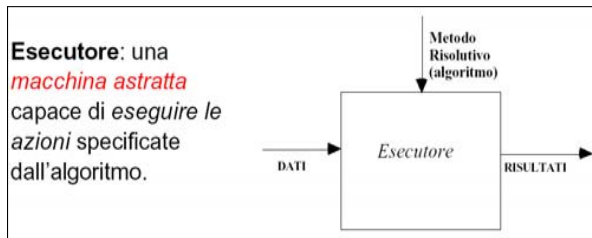
◆ Proprietà desiderabile

- **Efficienza**: deve risolvere il problema utilizzando al meglio le risorse a disposizione

19

Algoritmo: esecuzione

L'esecuzione delle azioni *nell'ordine specificato dall'algoritmo* consente di ottenere, a partire dai dati di ingresso, i risultati che risolvono la particolare **istanza** il problema



18

Rappresentazione degli algoritmi

➡ Linguaggio naturale

Linguaggi informali

2. Diagrammi di flusso

Linguaggi semi-formali

3. Pseudo-codice

4. Linguaggio di programmazione

Linguaggi formali

20

ESEMPIO: calcolo del MCD

Problema:

- Dati due interi M ed N (input)
- calcolare il Massimo Comun Divisore (MCD) fra M e N

Algoritmo 1* (descritto in linguaggio naturale):

1. Calcola l'insieme A dei divisori di M
2. Calcola l'insieme B dei divisori di N
3. Calcola l'insieme C dei divisori comuni = $A \cap B$
4. Il risultato è il massimo dell'insieme C

Presuppone l'esistenza di un esecutore in grado di interpretarlo ed eseguirlo!

21

Problemi non risolvibili

Non ammettono algoritmi di risoluzione con nessun modello di calcolo reale o astratto

Esempio:

data una funzione $f: \mathbb{N} \rightarrow \mathbb{N}$ (non nota), stabilire se $f(x)$ è costante per ogni valore di x

23

Complessità degli algoritmi e dei problemi

◆ Complessità (temporale) di un algoritmo

- indica l'andamento (in genere, asintotico) del tempo impiegato per risolvere il problema
- ... in funzione della dimensione dell'input (quantità di memoria necessaria per rappresentare i dati di input)

◆ Classificazione dei problemi

■ Trattabili

- Esistono algoritmi con complessità polinomiale (rispetto alla dimensione dell'input) che li risolvono (es., massimo)

■ Intrattabili

- non esistono algoritmi con complessità polinomiale che li risolvono

■ Non risolvibili

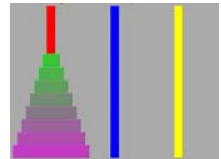
- non esistono algoritmi che li risolvono

22

Un problema intrattabile: le torri di Hanoi

■ Problema

- Dati 3 paletti e n dischi di raggio diverso, posizionati sul 1° paletto
- Spostare i dischi sul 3° paletto, uno per volta, evitando di porre un disco su un altro più piccolo



■ Complessità esponenziale

- Nr. mosse = $2^n - 1$

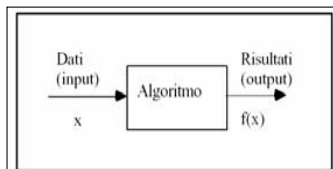
■ Per $n = 64$ (dischi)

- mosse = $2^n - 1 = 2^{64} - 1 = 18'446'744'073'709'551'615$
- 1 mossa al giorno = $5,0539025 \cdot 10^{16}$ anni
- ci restano 50 milioni di miliardi di anni !

24

Algoritmi equivalenti

- In generale un algoritmo può essere visto come una **funzione** da un dominio di ingresso (**input**) ad dominio di uscita (**output**)



- Due algoritmi si dicono equivalenti quando:
 - hanno stesso dominio di ingresso e stesso dominio di uscita
 - in corrispondenza degli stessi valori nel dominio di ingresso producono gli stessi valori nel dominio di uscita.

25

Variabili (cenni)

- Rappresentano dei “contenitori” per dati
- Ogni variabile possiede
 - nome
 - tipo
 - dominio dei valori che può assumere (es., numeri interi, caratteri,...)
 - valore
 - caratterizza lo stato della variabile, che può cambiare durante l'esecuzione
- Esempio:
 - $N = 2$ (il valore iniziale di N è 2)
 - $N = N + 3$ (il valore corrente di N è 5)

27

Calcolo del MCD: un altro algoritmo

Metodo di Euclide

$$\text{MCD}(M, N) = \begin{cases} M \text{ (oppure } N) & \text{se } M=N \\ \text{MCD}(M-N, N) & \text{se } M>N \\ \text{MCD}(M, N-M) & \text{se } M<N \end{cases}$$

Algoritmo 2

- Finché $M \neq N$
 - se $M > N$, sostituisci a M il valore $M-N$
 - altrimenti sostituisci a N il valore $N-M$
- Il Massimo Comun Divisore è M (o N), cioè il valore finale ottenuto quando M e N diventano uguali

I simboli M e N sono due **variabili**



26

Variabili: esempio

Calcoliamo il MCD di $M = 24$ e $N = 14$.

- $M=24, N=14$
 $24 > 14 \rightarrow M = 24 - 14 = 10$
- $M=10, N=14$
 $10 < 14 \rightarrow N = 14 - 10 = 4$
- $M=10, N=4$
 $10 > 4 \rightarrow M = 10 - 4 = 6$
- $M=6, N=4$
 $6 > 4 \rightarrow M = 6 - 4 = 2$
- $M=2, N=4$
 $2 < 4 \rightarrow N = 4 - 2 = 2$
- $M=2, N=2$
 $2 = 2 \rightarrow$ “il MCD di 24 e 14 è 2”

(Algoritmo)

- Finché $M \neq N$
 - se $M > N$, sostituisci a M il valore $M-N$
 - altrimenti sostituisci a N il valore $N-M$
- Il MCD è M ($=N$)

28

Calcolo del MCD (3)

Algoritmo n° 3

Dati due interi M e N ($M \geq N$)

1. Dividi M per N , e sia R il resto della divisione;
2. Se $R=0$ allora termina: N è il MCD;
3. Altrimenti assegna a M il valore di N ed a N il valore del resto R e torna al punto 1.

Osservazione

I tre algoritmi visti per il calcolo del MCD sono equivalenti, ma differiscono per efficienza

29

(Rappresentazione degli algoritmi)

1. Linguaggio naturale

Linguaggi informali



Diagrammi di flusso

Linguaggi semi-formali

3. Pseudo-codice

4. Linguaggio di programmazione

Linguaggi formali

31

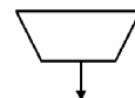
Calcolo del MCD (3): applicazione

Calcoliamo il MCD di $M = 24$ e $N = 14$.

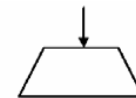
1. $M=24, N=14$
 $24/14 = 1, R=10 \rightarrow M=N=14, N=R=10$
2. $M=14, N=10$
 $14/10 = 1, R=4 \rightarrow M=N=10, N=R=4$
3. $M=10, N=4$
 $10/4 = 2, R=2 \rightarrow M=N=4, N=R=2$
4. $M=4, N=2$
 $4/2 = 2, R=0 \rightarrow$ "il MCD di 24 e 14 è 2"

30

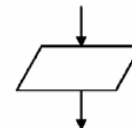
Diagrammi di flusso



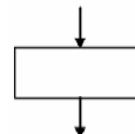
Inizio



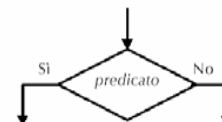
Fine



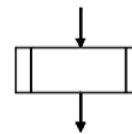
Operazioni
di ingresso/uscita



Elaborazione



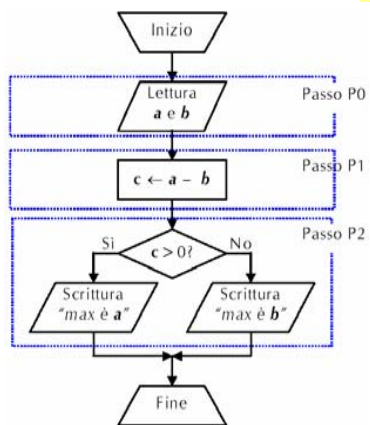
Selezione a due vie



Sottoprogramma

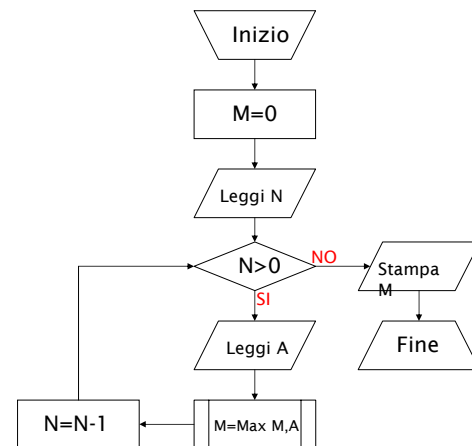
32

Esempio: massimo tra due numeri



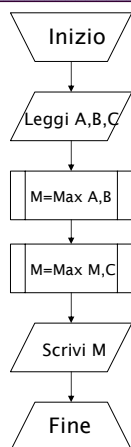
33

Esempio: massimo tra N numeri positivi



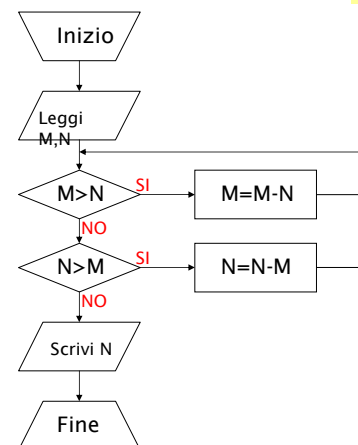
35

Esempio: massimo tra tre numeri



34

Esempio: MCD



36

Risoluzione di problemi con il calcolatore

- ◆ Un calcolatore (elaboratore) è una macchina (reale o virtuale) in grado di eseguire **azioni** elementari su **dati**
- ◆ L'esecuzione delle azioni elementari è richiesta all'elaboratore tramite comandi chiamati **istruzioni**
- ◆ Le istruzioni sono espresse attraverso **frasi** di un opportuno **linguaggio di programmazione**
 - **linguaggio macchina** nel caso dell'hardware
- ◆ Un **programma** è la formulazione testuale di un algoritmo in un linguaggio di programmazione
 - in accordo alla sintassi e alla semantica del linguaggio di programmazione scelto

37

(Rappresentazione degli algoritmi)

1. Linguaggio naturale

Linguaggi informali

2. Diagrammi di flusso

Linguaggi semi-formali

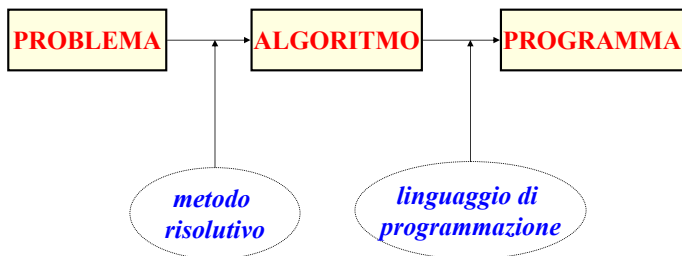
3. Pseudo-codice

➔ **Linguaggi di programmazione**

Linguaggi formali

39

Algoritmi e programmi



38

Elementi tipici di un linguaggio di programmazione

◆ Operazioni elementari

- Operazioni aritmetiche e assegnamento di valori a singole variabili
Es. $C = (A + B)$ ($C \leftarrow A + B$)
- Condizioni sul valore di singole variabili
Es. Se $A > B$ allora ...
- Lettura e scrittura di variabili
Es. **Leggi** A oppure **Stampa** B

◆ Istruzioni di controllo del flusso

- Specificano l'ordine di esecuzione delle varie operazioni di un programma

40

Strutture di controllo: SEQUENZA

- ◆ Le istruzioni devono semplicemente essere eseguite nell'**ordine** in cui sono presentate:
 - 1- solleva il ricevitore
 - 2- componi il numero
 - 3- ...
- ◆ Una sequenza di istruzioni può essere **raggruppata** in modo da diventare una nuova macro-istruzione:

INIZIO

solleva il ricevitore

componi il numero

...

FINE

41

Strutture di controllo: ITERAZIONE

- ◆ Le istruzioni devono essere eseguite ripetutamente fino a che non si verifica una determinata condizione
- ◆ Esempio:

RIPETI

componi il numero

FINO a che la linea è libera

43

Strutture di controllo: CONDIZIONE

- ◆ Le istruzioni da eseguire sono determinate dalla valutazione di una data **condizione**
- ◆ Esempio:

SE il numero è libero

ALLORA

attendi la risposta

conduci la conversazione

deponi il ricevitore

ALTRIMENTI

deponi il ricevitore

42

Esempio: Calcolo della potenza (algoritmo)

- ◆ **Problema:**
Dati due interi **a** e **n** calcolare la potenza **a^n**
- ◆ **Algoritmo (in pseudo-codice):**
 1. inizializza le **variabili** $K = n$, $Ris = 1$
 2. fino a che $K > 0$
 - 2.1 calcola $Ris \cdot a$ e memorizzalo in Ris
 - 2.2 decrementa K

Correttezza: al termine $Ris = a^n$

44

Linguaggi ad alto livello

- ◆ Conviene impostare la soluzione di un problema a partire dalle “mosse elementari” del linguaggio macchina?
 - **SI**, per risolvere il problema con **efficienza**
 - **NO**, se la macchina di partenza ha mosse di **livello troppo basso** (difficile progettare un algoritmo)



- ◆ **Linguaggi di Programmazione ad Alto Livello**

- Alto livello di **astrazione**: le istruzioni corrispondono ad operazioni più complesse
- esempi: Pascal, Basic, C, C++, Java
- E' necessario tradurre il programma nel linguaggio macchina mediante opportuni programmi (**interprete** o **compilatore**)

45

Meccanismi di astrazione

ASTRAZIONE: processo di aggregazione di informazioni e dati e di sintesi di modelli concettuali che ne enunciano le proprietà rilevanti escludendo i dettagli inessenziali

- ◆ **Tipi di dati astratti**

- **astrazione sui dati**: tipi complessi definiti dal programmatore (in base a tipi elementari o altri tipi astratti)



- ◆ **Sottoprogrammi**

- funzioni, procedure, metodi
- **astrazione sulle operazioni**: operazioni ad alto livello definite dal programmatore (in base a operazioni elementari o altri sottoprogrammi)

47

Esempio: calcolo della potenza (programma)

Programma (in pseudo-Pascal):

```
PROGRAM potenza;
INTEGER Ris,N,A;
BEGIN
  READ(N);
  READ(A);
  Ris=1;
  WHILE (N>0) DO
  BEGIN
    Ris=Ris*A;
    N=N-1;
  END;
  PRINT(Ris);
END.
```

L'esecutore deve:

1. leggere i valori iniziali dei parametri (*N* e *A*) dall'**input** (es., tastiera)
2. stampare il risultato (valore finale di *Ris*) sull'**output** (es, video)



Il programma (in linguaggio ad alto livello) deve essere tradotto nel **linguaggio macchina** del calcolatore

46

Esempio (sottoprogrammi)

Programma:

```
PROGRAM MaxTreNumeri;
INTEGER X,Y,Z, M;
BEGIN
  READ(X);
  READ(Y);
  READ(Z);
  M=Max(X,Y);
  M=Max(X,Y);
  PRINT(M);
END.
```

SottoProgramma (funzione):

```
FUNCTION Max (INTEGER X,Y): INTEGER;
BEGIN
  IF (X>=Y) THEN
    Max=X;
  ELSE
    Max=Y;
  END
```

N.B.: L'esempio è poco significativo dal punto di vista pratico

48

Meccanismi di astrazione

◆ Meccanismi più evoluti

- Classi (incapsulamento):
 - modellano variabili complesse (oggetti) con funzioni (metodi) che indicano le principali operazioni richiamabili su di esse (comportamento)
- *Information hiding*, Ereditarietà, Polimorfismo

◆ Vantaggi dell'astrazione

- Modularità dei programmi: componenti distinte e indipendenti con interfacce per l'interazione
- Progettazione *divide et impera*
- Facilità verifica e manutenzione
- Riuso